

TECHNICAL COMMUNICATION

III SEM.

INTRODUCTION TO TECHNICAL COMMUNICATION

1

PREVIOUS YEARS QUESTIONS

PART-A

Q.1 Define Technical Communication. [R.T.U. 2019, 2018]

Ans. Technical Communication : Technical communication is a broad field and includes any form of communication that exhibits one or more of the following characteristics:

- Communicating about technical or specialized topics, such as computer applications, medical procedures, or environmental regulations.
- Communicating by using technology, such as web pages, help files or social media sites.
- Providing instructions about how to do something, regardless of how technical the task is or even if technology is used to create or distribute that communication.

Q.2 What is linguistic ability?

OR

Explain methods to improve vocabulary and linguistic abilities. [R.T.U. 2018]

Ans. Linguistic intelligence/ability is a part of Howard Gardner's multiple intelligence theory that deals with individuals ability to understand both spoken and written language as well as their ability to speak and write themselves. In a practical sense linguistic intelligence is the extent to which an individual can use language, both written and verbal to achieve goals. In addition to this, high linguistic intelligence has been linked to improved problem solving, as well as to increased abstract reasoning.

In many cases, only the verbal aspects are taken into consideration. This is usually referred to as verbal intelligence or verbal fluency, and is commonly a reflection of an individual's overall linguistic intelligence.

PART-B

Q.3 Discuss the importance and characteristics of technical communication. [R.T.U. 2019]

Ans. Importance of Technical Communication : The functional importance of technical communication for an individual or an organisation cannot be overemphasised. Whether you are an executive working in a multinational company, and engineer working at the shop floor level, a scientist working in a premier scientific lab, or a technical student studying in a professional institution, you need effective technical communication skills in order to be successful.

Technical communication is the essence of organisational life and a prerequisite to effective management. The list of its multipurpose functions is very long. It not only makes professional interaction possible but also directs the flow of technical information and knowledge for the guidance of technocrats, engineers, and others in their professional activities. It stimulates scientists and researchers to act to achieve individual as well as social and organisational objectives and develops information and understanding essential or effective group functioning. Moreover, it leads to unification between the activities of individuals as a work team and helps to foster positive attitude required for motivation, cooperation, and other important organisational processes. Finally, it ensure free exchange of information and ideas and promotes scientific temper and maintenance of professional realtions.

Q.7 *Discuss different forms of technical communications.*

Ans. Forms of Technical Communications : The fast growth of technical knowledge coupled with the development of sophisticated information technology has changed the way we communicate in professional situations. We prefer fast, interactive, and result-oriented forms of communication such as voice-mail, email, video transmission, teleconferencing, videoconferencing, intranet transmission, and so on to the traditional and slow forms of communication such as letters, memos, newsletters, and so forth. Today, one may find a range of interactive technologies for communication. However, technical communication still depends on the two basic forms of communication : oral and written.

1. Oral and Written Communication : Each form of communication has its merits and demerits. While oral communication provides immediate feedback and promotes better understanding, written communication provides a permanent record and facilitates the creation of organised messages. Oral communication gives a personal touch to the communication process whereas written

10.4
communication is impersonal as the two communicators cannot see each other. The choice of an appropriate form of communication may depend on the need and purpose of the communication.

2. External and Internal Communication : There are two widely recognised categories of technical communication : external communication and internal communication. External communication consists primarily of describing the areas of expertise technical people or professionals provide. This may include any communication that an organisation does with people belonging to other organisations. Internal communication includes memos describing problems or requesting additional resources, different kinds of reports and proposals, internal presentations, company meetings, policy statements, office instructions, and office descriptions to be used for public releases or company websites.

3. General and Technical Communication : Technical communication is the process of communicating a specific message to a specific audience with a specific purpose. The repetition of the word 'specific' indicates the special nature of technical communication. It is this need-based specific characteristic of technical communication that makes it different from general or literary communication.

There are several characteristic features of technical communication that makes it different from general communication. Technical communication differs from general communication in content, style and approach of presentation, attitude, organisation and language.

Difference between General and Technical Communication

General Communication	Technical Communication
• General content • General vocabulary • Usually no formal elements • Both, factual and non-factual • Both, objective and subjective • Not always structured • Usually no specific exposition techniques • Not always for a specific audience • May or may not involve graphics	• Technical content • Specialised vocabulary • Format elements • Always formal in style • Always factual • Objective • Logically organised and structured • Complex and important exposition techniques • Specific audience • Usually, involves graphics

or the day after tomorrow, or the next month? The audience will definitely be confused. A better version will be: He will meet you by March 15.

You must understand the need for using specific language. While vague and abstract words create ambiguity and lead to incomplete communication or miscommunication, the use of specific and concrete words ensures limited interpretation, reducing the risk of confusion and misunderstanding.

Checklist for Achieving Clarity

- Do not use indirect expressions.
- Use specific words and phrases.
- Avoid exaggeration, artificial eloquence, and ornamentation.
- Use direct sentence.

PART-C

Q.10 What are the Technical communication skills? Discuss in detail.

OR

Describe four basic technical communication skills and how to develop these four essential skills of technical communication? [R.T.U. 2019]

OR

Describe technical communication skills (Listening, speaking, writing and reading) and their importance for engineers. [R.T.U. 2018]

Ans. Technical Communication Skills : Refer to Q.5.

1. Listening : Listening skills, as summarised in table 1 are probably one of the most important language skills that you need in order to be successful in your academic and professional pursuits. As student, you have to listen to lectures, explanations during tutorials and practical sessions, seminars and workshops, technical presentations, academic discussion, academic interactions, viva voce, and so on. Once you join a profession, you may be required to listen to discussions in meeting and conferences, seminars, business presentations, news bulletins, pre-recorded talks, telephonic conversations, teleconferences and videoconferences, and so forth. In fact, it is hard to imagine any academic, professional or business that does not require efficient listening skills.

Table-1: Listening Skills

General	Academic Listening to/during	Professional
Casual conversation	Lectures	Professional interaction
Formal conversation	Tutorials and practices	Meetings
Social interaction	Seminars and workshops	Conferences
Public speeches	Technical presentations	Interviews
Announcements	Academic discussions	Professional discussion
Radio	Academic interaction	Professional presentations
Television programmers	Viva voce	Teleconferencing
News reports		Videoconferencing

Moreover, one needs effective listening skills as it is a state of receptivity that permits understanding of what is heard and grants the listener full partnership in the communication process. One cannot be an effective communicator unless one is an effective listener. In any oral communicative situation, listening is of special significance because oral communication situation largely depends on one's ability to understand and appreciate the speaker's perspective. A student requires quick, efficient and imaginative listening techniques in order to achieve academic success because his/her academic performance partly depends on the quality and quality of listening. By listening to classroom lectures, academic discussions in seminars and workshops, and academic speeches the student acquires the professional knowledge and expertise needed to excel in his/her profession.

The importance of listening also lies in its multipurpose functions. As a purposeful communicative activity, listening serves individual needs and aids in implementing oral communication goals. There are many ways by which decisions are made in an organisation-by discussion in meetings and conferences, by negotiation in business dealings, by voting in democratic processes, and so on. In each instance, listening plays an important role as it aids in analysing a problem, understanding the possible solutions, and making a decision. In brief, listening:

- Makes oral interaction possible
- Aids in decision making

- Stimulates others to achieve individual as well as professional objectives.
- Develops information and understanding essential for decision-making in meetings and conferences.
- Leads to unification between the activities of individuals as a work team, towards achievement of common business, professional, or organisational goals.
- Promotes maintenance of professional relations.

2. **Speaking** : Speaking skills, as summarised in Table 2, are also very important for a person's professional survival and growth. One needs them to be successful in one's academic and professional pursuits. In fact, every academic, professional, or business work requires effective speaking skills. Whether you are a scientist engaged in technical research, a business executive involved in business activities, or a student working for a professional degree, you need effective speaking skills in order to take an active part in oral communicative processes.

Table-2 : Speaking Skills

General	Academic	Professional
	Listening to/during	
Casual conversation	Lectures	Professional interaction
Formal conversation	Tutorials and practicals	Oral reports
Social interaction	Seminars and workshops	Oral presentations
Public speeches	Technical presentations	Group discussions/Meeting
Small talks on local topics	Academic discussions	Conferences
Negotiating meetings in social situations	Viva voce	Interviews
Conversation talks in a given social milieu	Oral interaction	Professional discussions
	Oral reports	Teleconferencing
		Videoconferencing

A person requires effective oral communication skills in several formal communicative situations, i.e. meetings, conferences, group discussions, panel discussions, interviews, teleconferencing, videoconferencing, seminars, workshops, and symposiums. Some of the sub-skills of oral communication that one may need include the following:

Asking and Answering Questions

- Asking questions in meetings, talks, business presentations, lectures, seminars, conferences, training sessions, fieldwork, and laboratory work.
- Asking questions related to non-verbal data, formulae, equations, measures, numbers, fractions, and cardinal and ordinal numbers.
- Answering questions, providing clarification and giving explanations in the above situations.

Expressing opinions and comments

- Asking for opinions
- Expressing criticism, objections and doubt
- Expressing general comments
- Agreeing and disagreeing
- Seeking suggestions
- Giving suggestions
- Stating points of view

Academic and Professional Oral Interaction

- Interacting during interviews and tests
- Interacting orally with experts or professionals

Academic/Professional Discussions

- Discussing academic matter with fellow students
- Inviting to discuss
- Initiating discussion
- Initiating topic shift
- Giving information
- Coordinating in discussions

Meetings/Conferences

- Coordinating in meetings
- Conducting meetings
- Participating in meetings
- Summarizing and concluding in meetings

Oral Presentations

- Making oral presentations
- Orally describing and interpreting non-verbal data, e.g., graphs, tables, diagrams, charts, plans, maps, and so on
- Persuading
- Using visual aids effectively

Several factors have contributed to the increased importance of speaking skills today. The information revolution along with globalisation and other social and economic changes in the new millennium have increased the importance of speaking skills. Look around and you will find people involved in oral communication: Conversations, meeting, discussion, presentations, telephone calls, voice mails, teleconferencing, videoconferencing, gossiping, and so on. Speaking pervades the entire range of social and professional relationships, and plays a key role in our life. Without oral communication, social as well as professional interactions may not be possible.

The growth and expansion of service oriented industry has also increased the significance of oral competence for professionals in different fields. As the economy becomes less dependent on manufacturing and more service oriented, the need for students with good speaking skills has become more acute. Media reports frequently highlight employers' complaints that graduates' oral skills require considerable improvement. Students, thus, need to develop speaking skills if they are to be successful in their careers. Speaking skills are the single most important criterion in hiring professionals. Most of the professionals are hired through a selection process, which involves speaking skills in the form of group discussion/case discussion/personal interview/oral presentation or some other form of oral communication.

Moreover, increasing reliance on oral communication forms such as meetings, group discussions, conferences, seminars, workshops, and other forms of oral interaction in democratic decision making today has given increased significance to the need for effective speaking skills. Oral communication not only makes professional interaction possible, it develops information and understanding essential for decision making in meetings, conferences, and other similar oral communicative situations. Moreover, it leads to unification between the activities of individuals as a work team, towards achievement of common business, professional, or organisational goals; and promotes maintenance of business and professional relations.

As oral communication is a tool of professional and business interaction, one should be able to use it fluently, effectively, and confidently. A person should know how to speak using appropriate grammatical and lexical forms and how to sound using appropriate phonological and graphical

forms. One should have the ability to be an understanding interlocutor, create coherent discourse, and take appropriate turns in conversations, as well as have the ability to negotiate meanings with others by creating reciprocity of perspective on the topic of discussion.

3. **Reading** : Like listening and speaking, reading is crucial to effective communication. Your need effective reading skills as you may be required to read several kinds of reading materials for different general, academic, and professional purposes (summarised in table 3). Both professionals and students need excellent reading skills to be successful in their pursuits as it is hard to imagine any academic, professional, or business work that does not require efficient reading skills.

Table-3 : Reading Skills

General	Academic Reading	Professional
Newspapers	Text books	Business reports
Magazines	Journals	Proposals
Journals	Research papers	Business letters
Novels	Scientific articles	Memos
Stories	Classroom notes	Email messages
Articles	Lecture notes	Notes
Personal letters	Thesis	Notices
Emails	Dissertations	Circulars
General books	Abstracts	Promotional bulletins
Entertainment literature		Catalogues
		Instruction manuals
		Corporate brochures

The changes in business and industrial fields, coupled with the development of advanced information technology have greatly changed the way we receive and interpret information. Reading is not merely reading of letters and memos; it includes receiving and interpreting sophisticated technical material, email messages, intranet information, information of websites, and so on. Reading has new dimensions in industrial and technical organisations.

It is important that you are able to read and analyse scientific texts, instruction manuals, and technical materials in different forms. Comprehension of technical materials requires basic understanding of the technical subject, familiarity with scientific/technical terms, words and phrases, familiarity with scientific formulae/equations/abbreviations and with the discourse features of technical writing. It also includes the ability to understand and interpret graphic information.

Writing : The significance of writing skills (summarised in Table 4) cannot be overemphasised because writing is so important for students of all kinds and professionals in all fields. As a student, a person needs effective writing skills because one has to write examination answers, project reports, lab reports, summaries, synopsis, abstracts, and subject notes. Professionals are required to write business letters, memos, email messages, reports, proposals, minutes, notes, professional summaries, and so on. Both professionals and students need excellent writing skills to survive and excel in their pursuits as there is hardly any academic or professional activity that does not require writing skills.

Table-4 : Writing Skills

General	Academic Writing	Professional
Personal letter	Examination answers	Technical reports
Personal e-mail messages	Project reports	Industrial reports
Notes	Lab reports	Project proposals
Comments	Synopsis	Business proposals
General articles	Thesis	Business letters
	Dissertations	E mails
	Abstracts	Memos
	Research papers	Notices
	Scientific articles	Agenda
	Classroom notes	Minutes of meeting
	Lecture notes	Technical abstracts
		Job applications
		Resumes

COMPREHENSION OF TECHNICAL MATERIALS/TEXTS AND INFORMATION DESIGN & DEVELOPMENT

2

PREVIOUS YEARS QUESTIONS

PART-A

Q.1 What do you understand by print media?

[R.T.U. 2019]

Ans. Print Media : Print media advertising is a form of advertising that uses physically printed media, such as magazines and newspapers, to reach consumers, business customers and prospects.

Q.2 Mention various kinds of technical documents.

[R.T.U. 2019]

Ans. There are total nine types of technical documents;

1. Reports
2. Presentation
3. User manual
4. Executive summaries
5. Abstracts
6. Spec sheets
7. Proposals
8. Employment documents
9. Questionnaires

Q.3 What are technical manuals?

[R.T.U. 2018]

Ans. Document with instructions for operation, installation, use, maintenance, list of parts, support and any requirements for training for effectively using a machine or equipment are called technical manuals. 'Technical manual' is a term for any manual with a technical character. You could think of a

manual (or set of manuals) for a complicated machine like an airplane. You could also think of technical manuals to run a complete industrial installation.

- (i) Know who you are talking to
- (ii) Be aware of all directives
- (iii) Follow the guidelines
- (iv) Collect all the information that you can
- (v) Write effectively
- (vi) Use pictures
- (vii) Use animations

Q.4 Discuss different types of technical documents.

Ans. For the final report in some technical-writing courses, you can write one of (or even a combination of) several different types of reports. If there is some other type of report that you know about and want to write, get with your instructor to discuss it.

All the reports are briefly defined here, with cross-references to where their presentation occur:

1. Standard operating policies and procedures:

These are the operating documents for organization; they contain rules and regulations on how the organization and its members are expected to perform. Policies and procedures are like instructions, but they go much further. Standard operating procedures (SOPs) are more for procedures in which a process is performed-for example, taking a dental impression.

2. **Recommendation, feasibility, evaluation reports:** This group of similar reports does things like compare several options against a set of requirements and recommend one; consider an idea (plan, project) in terms of its "feasibility" for example, some combination of its technical, economical, social practicality or possibility; passes judgement on the worth or value of a thing by comparing it to a set of requirements, or criteria.
3. **Technical background reports:** This type is the hardest one to define but the one that most people write. It focuses on a technical topic, provides background on that topic for a specific set of readers who have specific needs for it. This report does not supply instructions, nor does it supply recommendations in any systematic way, nor does it report new and original data.
4. **Technical guides and handbooks:** Closely related to technical report but differing somewhat in purpose and audience are technical guides and handbooks.
5. **Business plans:** This type is a proposal to start a new business.
6. **Technical specifications:** This type presents descriptive and operational details on a new or updated product.

PART-B

Q.5 Describe the techniques to interpret and summarize technical texts. [R.T.U. 2019]

Ans. Techniques to Interpret Technical Texts :

1. **Skimming:** Skim the paper quickly, carefully review for headings, figures and the purpose of the reading.
2. **Vocabulary:** Go through the paper word by word and line by line, underlining or highlighting every word and phrase that don't understand. Look for underlined words and phrases. A technical English dictionary is a good place to look for definitions. Get an understanding from the context in which it is used. Often words that are used to describe the procedures used in engineering.
3. **Comprehension, section by section:** Try to deal with all the words and phrases, probably there will be a few technical terms that won't be understand. Now

go back and read the whole paper, section by section, for comprehension.

Techniques to Summarize Technical Texts :

1. Read the information source several times, in fact, read so many times as needed to fully understand the material.
2. Look up unfamiliar words and then if they're too difficult for understanding then rephrase them with simple words.
3. Construct the sentences clearly, don't forget to include the main points that want to deliver: for this jot down on the piece of paper to express the main ideas.
4. Keep it brief: don't lose the original essence but try to make it look more laconic, reduce and delete ruthlessly all the unessential sentences.

Q.6 What do you know about information design? Discuss its hierarchical levels.

OR

Describe strategies for organization of information and information design. [R.T.U. 2018]

Ans. Information design is the detailed planning of specific information that is to be provided to a particular audience to meet specific objectives. The information designer may or may not have available (or may create) an information architecture that defines the overall pattern or structure that is imposed on the information design and an information plan that defines information units and how they are to be completed. The output of an information design is sometimes expressed in written instructions, plans, sketches, drawings, or formal specifications. However, on very small projects, information design is likely to be much less formal.

Information design can be distinguished from information architecture and information planning. In one view, there are three hierarchical levels of activity:

1. **Information architecture:** Which is the general set of ideas about how all information in a given context should be organized. For example, one might say that "All of our product information should serve customer needs as expressed by tasks they have to do with our products," and then develop a pattern that organizes all product information in modules related to customer tasks. The output is an information architecture document.

2. **Information planning:** Which focuses on all aspects required to prepare and support the information of a specific set of products, single product, or event over the product life or other time span. This generally includes

understanding the product or event goals, studying the audience and their needs, considering possible information media, defining specific information units (books, chapters, web pages, visualizations, and so forth), specifying the people who will work on them, what the schedule is, and how this work will relate to the work of others. The output is an information plan.

3. **Information design:** Which focuses more narrowly on the information itself in one or more information units, and may encompass the information aspects of industrial design (labels, knobs, and the physical interface), information content design, page design, web site design, illustration design, typography decisions, and so forth. Information design can be applied to a single work, such as a city map, or to a corporation's entire set of customer information. The output may be part of an information plan, a separate information design document, or simply the designed object or set of objects.

Information design ideas can often be tested in a usability laboratory by observing surrogate users trying to use the designed information and getting their feedback. The practice of information design invites questions into how people learn or prefer to learn and how they use information. It also raises questions about how to design information for different cultural and other contextual differences in the audience.

Since information is now commonly delivered using electronic media with new possibilities for user interaction and as product designers have become more aware of the importance of usability, a new term, interaction design, has arisen as a corollary of information design. A number of universities now offer courses or degree programs centered on information design.

Q.7 What is the procedure for reading technical texts? Discuss role of first reading and second reading.

Ans. Introduction: Practitioners entering scientific and technical fields are confronted with an ocean of information and the tide will constantly rise. As a reader, you will struggle to stay current with the literature in order to maintain and advance your position in your field. It is a truism that the task of reading all the information published in leading journals, in all but the most narrowly defined specialities, is impossible. The problem for you as a reader will not be reading faster, but learning how to select what is worthwhile to read slowly. As a layperson reading in areas outside your field of expertise, you will be confronted by a language not your own. The

Q.9 *What is note-making? Discuss its importance and mechanism?*

Ans. Note-making is a brief record of writing something to assist the memory or for future important references. Notes are usually being taken to record the speeches or dictations after listening to it or after reading any materials like a book, magazine or an article. It is initiated to refer whenever needed and may be reproduced in the desired manner.

Importance of Note-Making : The process of grasping knowledge is vast and unlimited. Our memory is not able to store so much information altogether, hence notes making is quite handy and helpful. With the assistance of note making, we may recall the entire information of past events. For the students, it makes their preparation less strenuous.

Thus note making helps in keeping information handy whenever we require. Recollecting and recalling past events said or heard. Delivery of speeches for debates/discussion, writing an essay and revising lessons before examinations.

How to make notes: Reading of passage or the extract provided. While reading the extract, underline the important key sentences. It helps in the formation of the titles and subtitles. Make a rough note of all important and main points to be included and then give them a logical sequence. Use a format which is applicable to the extract and write down the summarization of the paragraph. They should be precise and to the point. They should have all the relevant and important information of the extract. All information should be systematically divided and subdivided. Use universally relevant symbols and abbreviations necessary and provide key to the abbreviations.

The main title should be short and apt to the extract. Try to avoid any long sentence as a title unless it is a common proverb. It must reflect the essence of the extract. Notes must be written in points and must be listed one after another and must be numbered properly.

Points to be avoided while making notes

1. Avoid the usage of full sentences.
2. Do not lift portions of the paragraph to form the notes.

3. Do not include irrelevant parts of the extract while taking down notes.

Mechanism- Note Making

Usage of abbreviation:

Capital letters to be used for first letter of the words for (eg: UNO, NCERT etc.).

Usage of abbreviations and symbols (for e.g. >, < @, % etc.)

Commonly used in newspapers and magazines. (gov., sec., Bio.)

Key to be used for own short forms

(For e.g. Trans.=Transparent).

Proper Indentation

Title (reflects the main theme of the extract and carries 3 marks, it should not be more than 1/3 of the passage length)

Subheadings (grasp of the passage by the students)

Try to use only phrases and avoid using full sentences.

Q.10 *What are the differences between writing for online and print media?*

Ans. People do not read how they used to. Thanks to the internet, people encounter so much more content than they once did - articles and informative pieces are available anytime, instantly. This means writing for the web differs greatly from writing for traditional media. Professors now teach writing students how to keep online readers engaged, so someone earning a bachelor of Arts in communication studies will be well-versed in internet writing. For other writers making the leap from paper to screen, there are several things to keep in mind.

1. Length

The average amount of time an online reader spends reading anything on the internet is two minutes. Printed pieces rarely even reach their main focus in that amount of time. Therefore, online articles tend to be much shorter than their printed counterparts and that goes for paragraphs, too. Instead of long segments of prose, like you would see in a newspaper or magazine, the paragraphs in online articles should end after a few lines.

but they are not always the most appropriate. If you decide to choose more innovative, visual or participatory tools, make sure that your evaluation users agree these ways of collecting information will provide sufficiently credible evidence.

Appropriateness to the Service and Its Values:

It's important to consider which tools are most appropriate to your organisation and its values. In some situations, more informal methods or anecdotal data may be acceptable, and in others you may need to use methods that are tried and tested, such as a validated questionnaire recognised and used in a particular field, for example, mental health.

User Views and Rights : Before you collect information from people, you should make sure they know why you are doing it, and how you will use the information. Make sure that they feel comfortable about the way that you are collecting the information and that you will treat information confidentially. For example, it might be useful to tape the interview, but you must always check whether your interviewee is happy with this. Think about other needs that they might have, which could relate to their learning, sight or hearing abilities, cultural background or other circumstances.

Reliability : A reliable method is one that can be applied consistently each time you use it, in different situations and with different people. It is essential when you are comparing information over time or between different participants and/or within different situations. If the question you ask can be interpreted differently by different people, your data may not be reliable.

Validity : Are you measuring what you intend to measure? To what extent, for example, are you relying on selective perception, rather than cross-checking through a number of data collection methods? How far do the question you ask through your data collection tools provide valid evidence?

Q.13 *What is organizational strategy? Discuss various types of strategy.*

Ans. Where do you want your company to go? What are your goals—more profits, more sales, bigger buildings? What do you envision for your business? To accomplish any of these objectives, you must have a plan—a road map of where you're

going and how you intend to get there. Who will be responsible for which parts of the journey?

These plans form the basis for an organizational strategy. It is the big picture of where you want the business to go and the activities needed to get there. An organizational strategy is the sum of the actions you must take to achieve your long-term goals.

Strategies are dynamic. They should be reviewed continuously to find out what's working and what's not. Adjustments can be made to get back on track. It's like playing chess. If a competition makes a move to counter your objectives, you adjust and counterattack. It's an ongoing process.

The time frame for a strategy is long term. Generally, this means one year, but it could be three or five years. The further ahead you plan, the fuzzier it gets. Constructing a strategic plan takes a lot of time and effort. More than likely, you'll have to begin working on the next strategic plan at least six months before the preceding plan ends. So, adjust the timeline to your needs and activities.

A strategy is a road map. It begins with a clear understanding of where you are now and where you want to go. The strategic journey defines the twists and turns in the road on the way to achieving your goals. It identifies the tasks along the way, who will be responsible for specific results, the mileposts and key performance indicators to measure progress and when results are expected.

Types of Strategies

A successful organizational strategy has a core purpose that defines the business. It is determined by the owner's vision of the company in the marketplace. For example, the owner may decide that the company should offer the best prices and be a cost leader in the market. This strategy may increase sales, but will it be profitable?

The business owner's vision of the company's position in the marketplace determines the type of organizational strategy to pursue. The overall objective is to gain a competitive advantage and maintain long-term profitability. A business has three choices of strategies to gain this advantage: low-cost production, differentiation or focus.

Low-cost production : The goal for this strategy is for the company to operate at the lowest cost possible and

offer lower prices to the consumer. This means the business has to minimize its cost of labor, find the cheapest source of materials and operate at minimal expense. The low-cost method increases profits by reducing the costs of services and products and charging lower prices to gain market share. If at thinner profit margins, the increased market share will lead to better overall profitability.

A no-frills airline is an example of the low-cost strategy. They do not offer any benefits other than a low-priced ticket. You'll have to pay for baggage, and they may even charge for the peanuts. The airline buys the cheapest fuel it can find, pays the least amount of wages that employees will tolerate and operates with a bare-bones overhead.

Differentiation : With a differentiation strategy, the company makes its products distinct from those of the competition. This means identifying unique features or functions of the products and promoting them to buyers. The success of this strategy relies on in-depth market research and finding innovative ways to attract customers. It depends on convincing consumers that the highlighted features are worth the price.

Sales and marketing programs must be designed to promote the differentiated benefits offered by the products

and send messages that are understood and believed by consumers.

Airlines that offer better service, more comfortable seats and better-quality food and snacks are examples of a differentiated strategy. These airlines try to convince passengers that even though ticket prices may be higher compared to the low-cost airline, they will have a more pleasant trip and receive more perks.

Focus : A company using the focus strategy concentrates on a specific niche market, which has a smaller number of potential buyers. The objective is to give exceptional product quality and customer service that results in strong brand loyalty. Competitors will have difficulty breaking into this niche market.

However, a focus strategy is not complete by itself. The company must still decide whether to compete as a low-cost producer or with differentiated product lines. Either way, the business has to offer something extra, such as a lower price or a better feature.

A small airline that serves markets with fewer flights is an example of a focus strategy.

□□□

Q.4

Spot the error and correct the following sentences :

- (a) We have received no information.**
- (b) I and he are brothers.**
- (c) I have no any friends.**
- (d) We should not make noise.**
- (e) He has finished his letter last night.**
- (f) If I were rich I will help you.**
- (g) She came yesterday, doesn't she?**
- (h) The last bus had gone until we had to walk home.**
- (i) Will you male this letter immediately?**
- (j) A person who can neither read or write is illiterate.**

[R.T.U. 2019]

Ans.(a) We have not received any information.

Ans.(b) He and I are brothers.

Ans.(c) I don't have any friend.

Ans.(d) We should not make a noise.

Ans.(e) He had finished his letter last night.

Ans.(f) If I were rich, I would help you.

Ans.(g) She came yesterday, didn't she?

Ans.(h) The last bus had gone so we had to walk home.

Ans.(i) Will you mail this letter immediately?

Ans.(j) A person who can neither read nor write is an illiterate.

- (7) *Not only Ram but only Sham is my friend.*
 (8) *A bus runs passed our house.*
 (9) *Neither Peter or James can sing well.*
 (10) *He drove fast so he might arrive early.*

[R.T.U. 2018]

Ans. Rewritten Sentences are as Below :

- (1) We all swam at the beach yesterday.
 (2) They had scarcely enough feed for their cattle.
 (3) The animals have damaged my plants.
 (4) The perfume is for your sister and you.
 (5) My father has slept all day.
 (6) Mother has cooked new dishes.
 (7) Not only Ram but also Sham is my friend.
 (8) A bus runs past our house.
 (9) Neither Peter nor James can sing well.
 (10) He drove fast so that he might arrive early.

Q.7 Describe the technical writing process.

OR

Describe the technical writing process in detail.

[R.T.U. 2018]

Ans. Beginning to write may be a hard task for most people as ideas come mixed up in a disorderly manner. In trying to get started, many different aspects come into mind: content, style, grammar, etc. and it may be difficult to cope with them all at the same time. In order to seek guidance and to acquire confidence, the writer may find it useful to resort to some kind of systematic and integrative approach which takes into consideration the most important aspects of writing.

The integrative approach adopted draws on knowledge of different approaches to writing (see figure 1). On the one hand, it takes into account the linguistic knowledge about texts, namely, grammar and text structure. Mastering syntax, an appropriate use of vocabulary and cohesive devices as well as patterns of information organization become essential to produce well-written texts. This is known as product approach. On the other hand, the integrative approach also pays attention to the writing skills or stages involved in writing.

Novice writers should be made aware of writing as a process consisting of different stages (planning, drafting, revising, etc.) when creating a text (process approach).



Fig. 1

Finally, this approach also heeds the social context, mainly the purpose and audience the document is addressed to, as well as the writing conventions established by the technical and scientific community (genre approach).

At this point it is useful to clarify that the three approaches mentioned above will be combined into one by subsuming the product and the genre approach under the process approach which, in turn, will serve as the guide to organizing the information to the next steps. However, the writing approach presented below is by no means intended to be prescriptive. Instead it has been designed to provide guidance allowing enough room for manoeuvre so that writers can adapt these guidelines to their own writing preferences and style. We view writing as a non-linear and recursive process composed of three main stages :

1. Pre-writing : Before beginning to write, invest some time to plan what to write and how to transmit the information. In order to do this, consider (a) audience and purpose (whom to write and why), (b) tone and style (how to transmit the information), (c) gathering of information (brainstorming, analysing sources of information, etc.) and (d) outlining (organization of information).

2. Writing : Once the information has been gathered and organized, start writing a first draft. At this stage, it is important to consider the main parts of the text, paragraph development and coherence as well as genre conventions. Revise and consider all these aspects, it may be helpful to use representative models as a reference.

3. Post-writing : The final stage of the writing process involves (a) revising content and organization, (b) checking

for grammatical accuracy (c) editing for style and (d) proofreading and peer review. These steps will help you spot any inconsistencies in document so as to produce a flawless final version.

The three main stages of the writing process together with their corresponding sub stages are shown in figure 2.

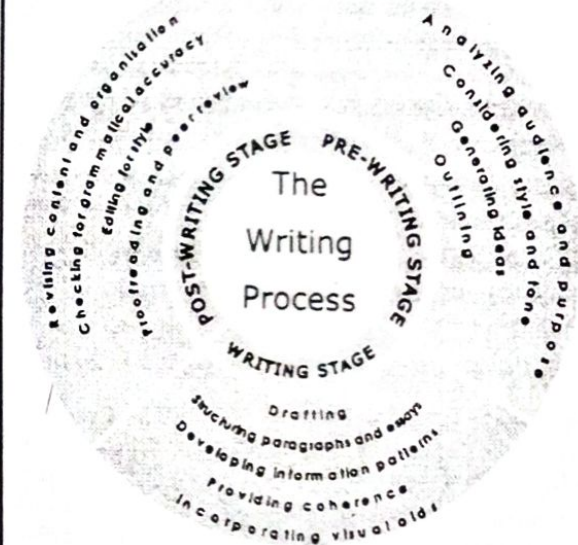


Fig. 2

Some of the benefits that can be obtained from adopting this process approach are outlined below:

- It helps the writer overcome the blank page syndrome and therefore get started.
- It serves the writer as a guide to writing since it suggests possible steps to follow in the writing process.
- It makes the writer aware of contextual considerations such as audience and purpose.
- It promotes awareness of the writing process.
- It accounts for individual variation, that is, it encompasses different learning styles and preferences.

Q.8 Discuss the three main steps of technical writing.

Ans. The process of writing is no easy feat. Whether the writing is for academic or professional ambitions, writers of all ages face the same challenges: coming up with ideas,

Q.12 *List any six most common grammatical errors found in technical writing.*

Ans. Some mistakes are more severe in nature when we write. These mistakes either compromise the clarity of your intended meaning or degrade the overall quality of your writing. Maintaining quality is important, so let's discuss some of the common grammar mistakes that must avoid to maintain a high writing standard in writing.

1. Subject-Verb Agreement Errors

The subject and verb of a sentence must agree with one another in number whether they are singular or plural. If the subject of the sentence is singular, its verb must also be singular; and if the subject is plural, the verb must also be plural.

Example:

Incorrect : An important part of my life have been the people who stood by me.

Correct : An important part of my life has been the people who stood by me.

2. No Comma in a Compound Sentence

A comma separates two or more independent clauses in a compound sentence separated by a conjunction. The comma goes after the first clause and before the coordinating conjunction that separates the clauses.

Example :

Incorrect : She was beautiful and she was happy and she was full of life.

Correct : She was beautiful, and she was happy, and she was full of life.

3. Vague Pronoun Reference

A pronoun can replace a noun, and its antecedent should be the person, place, or thing to which the pronoun refers. A vague pronoun reference (including words such as it, that, this, and which) can leave the reader confused about what or to whom the pronoun refers.

Example :

Incorrect : When Jonathan finally found his dog, he was so happy. (The dog or Jonathan?)

Correct : Jonathan was so happy when he finally found his dog.

4. Present vs. Future Tense

In technical writing, we should use the present tense as much as possible. Present tense indicates established facts and knowledge, which all writers should have available when creating public-facing information about a product. Future tense is rarely acceptable in technical writing because customers expect that companies are certain about "what happens," and that they are not simply predicting "what will happen."

Example :

Incorrect : A broken antenna will distort signals.

Correct : A broken antenna distorts signals.

5. Passive Voice

Active voice is always preferable for technical writing because it sounds more natural than passive voice. Passive voice usually leads to longer sentences with confusing structures. More importantly, it stresses the object rather than the subject, so unless that is your intention, avoid using passive voice and stick to a natural active style.

Example :

Incorrect : The report was presented by Prof. Howard.

Correct : Prof. Howard presented the report.

6. Using too many Nouns

Technical writing is very different from creative writing. In the latter, use of modifiers such as strong adjectives or abstract nouns is acceptable, if not desirable. In the case of technical writing, using too many nouns can create confusion.

Example :

Incorrect : Pushing the button causes the termination of the program.

Instead of using the abstract noun termination, it is better to write

Correct : Pushing the button terminates the program.

Q.13 How to design an effective resume. Explain.

Ans. A resume, from the French word for "summary," is a concise, standardized document that introduces you as a professional, most often for the purposes of seeking employment, but it also useful in other situations, such as applying for awards or seeking a promotion. It is likely a document you are already somewhat familiar with and you might even already have a resume of your own, but learning how to strategically and confidently build a resume for a particular audience and purpose is a vital professional communication skill. It is more than a list of jobs - it is a prioritized, condensed introduction to you, the job applicant, and it demands close attention.

Note : The practices outlined here are focused on business culture and norms in the United States, so keep in mind that you will need to research and follow the appropriate standards if you're applying for positions internationally.

To understand everything else that follows in this section, it is important to first consider the rhetorical situation for this particular type of communication. What is the intended effect? What are you trying to accomplish? Who is your audience? How will they be accessing and reading your document?

Audience Purpose

- Employer or representative of an employer
- Seeking potential employees
- Evaluating a pool of applicants to see if they are qualified, looking for a reason to say yes or no

Addresses

1. Your Address

The return address should be written in the top right-hand corner of the letter.

2. The Address of the person you are writing to

The inside address should be written on the left, starting below your address.

Date

Different people put the date on different sides of the page. You can write this on the right or the left on the line after the address you are writing to. Write the month as a word.

Salutation or greeting

1. Dear Sir or Madam,

If you do not know the name of the person you are writing to, use this. It is always advisable to try to find out a name.

2. Dear Mr Jenkins,

If you know the name, use the title (Mr, Mrs, Miss or Ms, Dr, etc.) and the surname only. If you are writing to a woman and do not know if she uses Mrs or Miss, you can use Ms, which is for married and single women.

Ending a letter

1. Yours Faithfully

If you do not know the name of the person, end the letter this way.

2. Yours Sincerely

If you know the name of the person, end the letter this way.

3. Your signature

Sign your name, then print it underneath the signature. If you think the person you are writing to might not know whether you are male or female, put your title in brackets after your name.

PART-C

Q.16 You and Anand/Arti of 18 Model Town, New Delhi.

You have seen an advertisement in the Hindu for the post of Engineer in an MNC. Apply for the job with complete resume.

[R.T.U. 2018]

OR

Imagine yourself as Mehul Kumar of Kanpur, write a job application with resume in response to the advertisement for the post of Technical Executive in Tata Motors Ltd. Andheri East, Mumbai.

[R.T.U. 2019]

Ans. 18, Model Town

New Delhi

June 15, 2018

The General Manager

Genus Power Ltd.

Gurgaon

Subject: Job application for the position of engineer.

Dear Sir/Madam,

I'm writing to express my interest in your company for the post of engineer advertised in The Hindu on 14 June, 2018. Your organization is a leading and well reputed company in producing electronic goods all over the country.

I am a B.tech graduate with electronics and communication stream from SKIT, Jaipur. I have been working with Enertrek Pvt. Ltd. for the last three years and possess enough knowledge in electronics, which render me quite suitable for the position. I wish to resign my present post as there is almost no scope for promotion.

Your organization will provide me time to promote my career. I think it could be mutually beneficial to both if I switch over to your reputed organization in mentioned vacancy. I enclose attested copies of my certificate, testimonials and updated C.V. for your consideration. Originals, in case required, shall be produced during interview.

I hope you will give me a chance to serve your company. I am waiting for your consideration.

Yours Faithfully,

Arti Singh

(16)

Arti Singh
18, Model Town
New Delhi
artisingh4@gmail.com

Professional Summary

Manager with three years of engineering experience in industrial settings, especially adept at developing and implementing business development strategies to improve operations efficiency. Committed to motivating staff to achieve goals while ensuring efficient project progress.

Core Qualifications

- Substantial project development and resource allocation experience
- Sound familiarity with Kanban scheduling systems MS Word and Excel
- Proficient in the use of engineering software
- Strong management and leadership qualities
- Comprehensive written and verbal communications abilities
- Good organizational skills

Experience

Engineering Manager (2015-2018)

Enertrek Pvt. Ltd.

Jaipur

- Implemented quality control standards in accordance with client requests.
- Managed and addressed HR issues including payroll and time off requests.
- Effectively managed a substantial engineering and production staff.
- Worked with clients to prepare project specifications.
- Identified and worked to resolve procedural issues.
- Performed internal audits and documented results.
- Ensured that all project requirements met.
- Maintained lean production methods.

Education

Bachelor of Technology – Engineering 2008

SKIT

Jaipur

PART-A

Q.1 Distinguish between technical proposal and technical report.

[R.T.U. 2019]

Ans. Technical Proposal : Document that lists and defines the technical requirements of a contract or project, and explains the approach and plan formulated to address them.

Technical Report : A technical report is a document that describes the process, progress, or results of technical or scientific research. It might also include recommendations and conclusions of the research.

Q.2 Define a technical project proposal.

OR

Describe what is a technical project proposal.

[R.T.U. 2018]

Ans. A technical proposal is a written official document to carry out some activity to specified conditions on either side with some financial consideration. One agency requires some material, some development or construction, some services, some software, or research. It advertises in a newspaper or a website or by any other mode, along with the nature of the product or services required, with detailed specification or scope of activity, conditions and time frame. The concerned, interested parties submit their techno-commercial proposals.

So, technical proposal is a written/printed document, indicating their interest in providing the service with specified conditions; procure material or equipment of given specifications; develop software for required application with given constrain; construct a building, road, bridge, railway line/station, dam, power house of given specifications; carry out research on a given problem or to solve some problem. The proposal also indicates financial expectations and other conditions like payment terms and time frame or schedule of completing the assignment.

Q.3 Explain the objectives of the technical reports.

Ans. In the world of science and technology, technical reports are extensively used. The report may be prepared individually or by a group, usually appointed for the purpose. It basically compiles the description about some process, it may be progress of some activity or an organization or a scheme. It may be in the form of results of a research or a survey. For journal or conference publication, the format is standardized. The technical papers or articles are reviewed and edited by the editorial board or referees.

Technical Reports : Reports are usually for the purpose of review. Sometimes it is mandatory by the funding agency. Sometimes, big organizations and government departments publish the reports for the benefit of the public and stake holders.

Depending on the objectives, there may be different types of reports. Laboratory reports, health and safety reports, research reports, progress reports, case study reports, field survey reports, cost analysis reports, proposal reports, technical reports and financial reports are the different types of reports. Reports are also aimed at finding advantages and

disadvantages of the project. Reports may be related to physical description, process analysis, examination, laboratory and design. Technical reports are also classified as technical background report for information on a technical topic, construction report for the activity or equipment, feasibility report for viability of a project, primary research report about experimentation and survey. Technical report may cover construction, working, special features and market potential of a new product, proposal reports are very detailed one, including feasibility analysis, literature review, etc. Business prospectus is also a report targeted at investors:

she may adopt strict norms and select only a few of them. In contrast, the organiser of a professional seminar may not have the option of rejecting many conference papers.

Review and Research Articles : A review article is an evaluation and analysis of published work on a particular topic. The main purpose of a review article is to evaluate a published work in order to determine its academic value and research potential. While reviewing a particular piece of published research work, the review writer tries to answer the question, "How does the work under discussion increase the sum of knowledge in a specific field?". The review writer may also comment on the necessity for future research in the concerned area and propose certain directions.

A research article, on the other hand, is based on original research carried out by the author. It may be the outcome of a particular research project carried out by the author or the result of a small scale study. The research might have been carried out in the laboratory or in the field. It might be theory based, or a part of action research to develop certain methods, equipment, procedures, systems, and so forth. The main purpose of a research article is to add to the existing knowledge, understanding, and scope of a particular subject. A research article could be either published in a journal or presented in a conference in order to achieve its objective.

Q.5 Write strategies, structure and formats of technical articles.

OR

Discuss the types, structure and written formats of technical articles. [R.T.U. 2018]

Ans. Elements of Technical Articles: Technical articles are formed technical documents, and are highly formalised in structure. They usually follow fixed-formula pattern to ensure objectivity. The components of a technical article include title, author by line, abstract, introduction methodology, results, discussion, conclusion, appendices, and references.

Title : All technical articles begin with a title. The title of a technical article is usually a long phrase that contains keywords and indicates the content of the article. It gives the readers a clear idea of the topic that is discussed in the article. A vague, abstract, or very general title will fail to indicate the content and will not be communicative. It is, therefore, important that the title of a technical article is informative, specific and comprehensive.

While writing the title of a technical article, sufficient information should be included for the reader to be able to understand the content of the article. This may be the title lengthy, but is better to use a lengthy title that is clear than to use a

short but unclear title. It is important to have a title that is specific and contains keywords that will guide the reader to the article.

Author By-line : This is the second element of a technical article. It includes the name of the author followed by institutional affiliations. The following are some examples:

1. LIZA SMITH
Nicholas Parker UK Ltd.
London EC3N 4HJ, UK
2. R. Ravi Kumar
Department of Organisation Behaviour,
Indian Institute of Management
Bangalore, India
3. V. Kumar and Dinesh Chandara
Department of Electronics and Instrumentation,
Indian School of Mines,
Dhanbad-826004, India

Abstract : The abstract is the most important element of a technical article. There are two kinds of abstract, that is, descriptive and informative. The descriptive abstract talks about the article and briefly states what the article contains while the informative abstract summarises the essential information in the article, focusing on key facts, findings, observation, results, conclusions, and recommendations. As informative abstracts are more comprehensive and self-explanatory than descriptive abstracts, most professional journals prefer the former to the latter.

Introduction : The main part of a technical article usually begins with an introduction that introduces the reader to the topic or research work under discussion. This section helps the reader to understand the article as it includes facts that the reader must know in order to comprehend the discussion and analysis that follow. A good article introduction may include some or all the following elements:

- **Background :** An objective description of the background of the problem, or events and conditions that led to the problem under discussion.
- **Research Status :** The status of research related to the problem, and the need of the present research.
- **Purpose :** Aims and objectives of the research or investigation.
- **Significance :** The significance of the work under discussion.
- **Methods :** Methods or procedures used to carry out the research work, and the rationale for using specific methods of investigation.
- **Scope :** The scope and limitation of the problem investigated.

Methodology : While writing a technical article, certain methods and materials need to be used to carry out the investigation. The professional value of an investigation largely depends on using appropriate methods and materials. In order to establish the validity of the findings, the materials and methods used need to be described with the rationale for using them. This may involve an objective description of materials that were used to conduct the investigation, the conditions under which the work was done, the methods of data collection, the procedures for investigating the situation/problem, and the criteria of survey.

Results/Findings: This is the main part of the article as it presents the results or findings of the investigation. This section is concerned with 'what was done', 'what happened' or 'what was looked into'. It focuses on facts and objectively presents the outcome, results, and findings of the research. It is usually divided into sections and sub-sections with well-structured and clear headings and sub-headings. This section should be written as carefully as possible because the value of the article depends on the relevance of its results.

Discussion: This section presents an analytical discussion of the results and findings of the investigation. It is an extension of the earlier section. As it conveys the significance meaning of the findings to the reader, it analysis the causes, implications, and significance of these findings. Moreover, it establishes a link between the findings and the existing research in the concerned area.

Conclusion: This section concludes the article by summarising the important highlights of the article. It may contain recommendations that are based on results and discussion. It may also indicate the need and nature for further work in the concerned area.

Appendices : A technical article may contain appendices. An appendix contains supporting material or data that is kept separate from the main body of the article to avoid interruptings the line of development of the article.

References: This section may contain references to books, journals, reports, dissertations, or/and published government documents, and so on used in the article. It may used the referencing method approved in the particular discipline to which the article belongs.

Budget Justification : When the project cost is very high, the justification of each major component of cost is desirable.

References : The foundation of new research is laid on the research work done by other people earlier. For some fundamental concepts some very old references can be quoted otherwise, recent references are considered relevant. Nowadays, the technology, particularly in electronics and computer is developing so fast that some products and procedure are outdated before routine introduction in the market. So, recent references are desirable.

Appendix: To maintain continuity of flow, some additional information is covered separately in appendix.

Q.13 Explain technical articles and also write their significance.

Ans. Refer to Q.5.

PART-C

Q.14 Discuss the kinds of Technical Proposals. Draft the format of technical proposal. [R.T.U. 2019]

Ans. There are four types of technical proposals :

1. Business Proposals
2. Research Proposals
3. Educational Proposals
4. Accreditation Proposals

1. Business Proposals : Supply of material, construction of roads, buildings, railway, a garden, a bridge, a dam, a party plot, a parking lot, etc; providing services for transportation, washing, catering, safety, security, insurance, fire fighting, training, consultation, recruitment, maintenance, etc; procurement of equipment, hardware, software, stationary, etc; all these can be considered as business proposals. One side has the requirement of material, machinery or man power or requires some type of service or requires development of some infrastructure or utilities. This first party has the funds but, it has no facility for in-house manufacture of the said item, or the temporary requirement is more than its manufacturing capacity, has no competent manpower to develop the said hardware or software, or to provide in house training, or the requirement is urgent and the organization cannot spend time for this activity or cannot wait, or in some cases the in-house solution is not cost effective or the

organizer is not able to diversify its activity. Under this condition, there is no option but, to take help of external persons or party. For the material, the type of material, its quality and quantity, schedule of requirement, affordable price, etc. may be identified. For the building or any other infrastructure or equipment, the specifications may be analyzed. For the services, the nature of services required, duration of the services, urgency or priority of the services, quality of services, quantum of services, liability on either side, etc. are fixed up. For the equipment, the specifications are decided. For the hardware and software, technical details are decided. In case of a problem or a project, it is well defined. In this category of business or sales proposal, the initiating company requires the material, machinery, manpower, services, etc. For this purpose, after finalizing the deal, agreement will be made with the selected proposer. Apart from technical aspects of the deal, the financial aspects like initial quoted price, negotiated price, taxes, transportation charges, mode of payment, penalty clause, etc. should be well clarified by both the sides otherwise, there may be a legal dispute and the assignment may be delayed. Sometimes earnest money deposit (EMC) is taken against major proposals.

Characteristics of business proposals: Business proposals lead to some form of agreement for material, machinery manpower or services from one party to the other party, against the financial return. Proposer has to convince the solicitor that it can supply the quality material at competitive price, required machinery with required specification at competitive price, provide the required manpower with appropriate qualification and experience, in required number at given time. In case of services, quality and promptness of service are to be ensured. In all the cases, competitive price is a deciding factor. The proposal should be rich in information. The arrangement of proposal should be easy to follow. The arguments should be logical. The information and supporting documents should be able to convince the initiator, about the competence, market standing, financial and technical soundness, capability and bonafide intentions of the proposer. Technical capability is also reflected by style of presentation in the proposal. Personal touch is desirable. The customer should fill that he is not spending the money but, is investing the money. This investment will earn value addition for him. The requirements and expectations of the solicitor should be well addressed. Doubts and expected queries, from the solicitor, should be addressed before hand. The structure of

the proposal should be sound. All the points related to the background of the proposer, objective of proposal, steps to be followed in implementation, schedule of implementation, quality assurance, specialty of approach, financial and technical soundness, keenness to undertake the assignment, payment terms, willingness to negotiate or justification of quoted price, strength of qualified and experienced manpower, major assignments successfully undertaken in past and final summary of the proposal should be covered and main points should be highlighted.

2. Research proposals : A research proposal is a document proposing a research project, generally in the sciences or academia, and generally constitutes a request for sponsorship of that research. Proposals are evaluated on the cost and potential impact of the proposed research, and on the soundness of the proposed plan for carrying it out. Research proposals generally address several key points:

- What research questions will be addressed and how they will be addressed.
- How much time and expense will be required for the research.
- What prior research has been done on the topic.
- How the results of the research will be evaluated.
- How the research will benefit the sponsoring organization and other parties.

Research proposals may be invited internally or externally. Government and public undertakings, private organizations and educational institutes encourage the research activities. For the technological growth of the nation and survival of the industries in global competition, research is inevitable. There are certain funding agencies at national and international level for the research. All India Council for Technical Education (AICTE), Government Department of Electronics (DOE), Department of Science and Technology (DST), Defence Research and Development Organization (DRDO), Central Scientific and Industrial Research Organization (CSIR) and University Grants Commission (UGC) are some of the government agencies, sponsoring research in science and technology. From time to time, they invite research proposals on thrust areas and other areas decided by them. There is some government organizations, exclusively set up to carry out full time research, in specific area already identified or areas which may be identified in future. Space Application Centre (SAC), Indian Space

Research Organization (ISRO), National Physical Research Laboratories (NPRL), Institute of Plasma Research (IPR), etc. have scientists and engineers employed on regular basis to carry out the research.

Characteristics of research proposals : R&D proposals stress on the research concept, its originality, and social, economical, academic gains. If some indigenous development, import substitution, cost effectiveness, energy conservation, new source of energy or efficiency gain is envisaged, it should be highlighted. If it is likely to lead to patents or intellectual property rights or prestige for the organizations involved, should be highlighted. Over all line of action and road map of project implementation should be presented. Capital and recurring costs, necessary infrastructure and other facilities required should be estimated. The contribution of either side should be well defined. This type of proposals are weighted by the qualification, experience, expertise and achievements of the coordinator or principal investigator of the R&D project proposed and his team, reputation and track record of the organization, where the project is to be carried out, infrastructure, library and computational facility available, and commitment of the organization.

3. Educational Proposals : Education spans from playgroup, kindergarten, primary school, secondary school, higher secondary school; under graduate colleges of various faculties like arts, commerce, science, law, pharmacy, engineering; post graduate institute and institutes offering Ph.D. programmes in various faculties and various branches in case of engineering. At lower, school level, local authority may be involved. At the higher school level, district and state authority may be involved. At the college level, state government, university and national authorities, like University Grants Commission (UGC), AICTE, Medical Council, Pharmacy Council, etc. may be involved. In all the cases, proposals are to be submitted to the competent authority, to initially start the institute with specified intake and later on to continue to run the institute. For schools and colleges, not offering professional courses, usually proposals are not invited. A particular trust or a body plans to start a school or a college. It prepares the proposal in a specified format and approaches the concerned authority.

Characteristics of educational proposals : For technical institute proposals, social commitment should be highlighted rather than profit orientation. The experience of running such other organizations, their success stories, acceptance by

students and employers, employability of students, quality assurance plan, faculty strength, infrastructure and utilities development, library, computer center, canteen facility, hostel are the key requirements. If some facility is existing, it should be highlighted with plan of expansion with time schedule and resources available or identified. The bonafide intentions are also projected by academicians, technocrats, social leaders, economists, government officers, and other renowned representatives in the governing council or board of trustees.

4. Accreditation Proposals : There is a national board accreditation for the accreditation of programmes offered by engineering institutes. Accreditation is programme wise, for the strengths and weaknesses, as per specified format. The detailed criterion covers teaching learning aspects, management aspects, administrative aspects and financial aspects. The information covers student intake, admission process, result, recruitment, student feedback, teacher feedback, employer feedback, etc. The visiting team interacts with the students, the faculty, the supporting staff, the management, the administration, the guardians and the employers too. Similarly, for Institutes and Universities, there is NAAC accreditation. These accreditations are not mandatory but, are useful for the institute or university to find its strengths and weaknesses and get third party assessment. There are various laboratories, testing equipments of different types. For the manufacturer, such testing is necessary for quality assurance with third party check. For the customer, it is necessary to verify the specifications and expected performance. The Bureau of Indian Standards (BIS) permits laboratories at some institutes to carry out the testing and issue certificate on behalf of BIS. There are some government and semi-government laboratories having good testing facility, to test as per the compliance of national or international standards. Electronic Quality and Development Centers (EQDC) and Software Testing and Quality Centers (STQC) also offer such facilities. These laboratories are approved by NABL, to undertake testing of certain items and software.

Nowadays, there is a trend for ISO certification. BIS and testing laboratories concentrate on quality of the product, in terms of verification of specifications and claimed performance. On the other hand, ISO concentrates on the verification of process and procedures. Logic is that, if the process and procedure are as per established standards, the product quality and performance of an organization can be

ensured. It may be a debatable point. But, ISO certification agencies are doing a good business. They do not invite the proposals. Government and private offices, educational institutes, banks and industries opt for such certification. There are number of consultants, who help in preparing and submitting the proposals for ISO certification and they also help for the defence of the proposal.

Characteristics of Accreditation Proposals :

Accreditation proposal formats for technical institutes are also well defined. Specific, quantitative information is required on various aspects. It may be laboratory facility, library facility, computer facility, faculty and staff strength, management, administrative, financial and academic details, hostel, transport, utilities. Feedback of all stakeholders is given due weightage. The format is so specific that it is difficult to hide weaknesses or negative points. For test laboratories, all the test equipments necessary to test with compliance of necessary standards are to be ensured. For ISO certification, procedures and practices ensuring smooth, systematic working and presenting transparency are to be highlighted.

Structure of the Proposal : The structure of the proposal is usually specified by the soliciting agency, as some well designed format.

Introductory Part

Cover Page: When some organization, industry or a statutory body invites the proposal in a specified format, this type of cover page may contain the name and address of the soliciting body inviting the proposal, along with its complete address, phone numbers, email addresses and its websites. In case of a statutory body, it mentions that the state or central government department under which it is working.

In bold letters and in a well defined rectangular block, it may mention the specific purpose, for which the proposals are invited and the target group of proposers.

The same page may also contain some instructions regarding preparation of the proposal. The dos and do not's for the proposal, the number of copies to be submitted and mailing address for the proposal may be specified by the soliciting organization.

Title Page : Title page is usually, the first page from the proposer side. Sometimes the format of this page may also be specified by the inviting organization. Otherwise, the proposer may prepare its design. The first option results in some uniformity. It is for administrative purpose. Again, it indicates the title of the proposal or project or assignment. It

also, covers the name of the authority to which proposal is submitted. The proposal usually goes in the name of the competent authority from proposer side. The authorized person may be a head of the department in an engineering educational institute, head of the educational institute, a registrar of the university, a principal investigator for the R & D proposal, a co-ordinator of the training programme or activity, or a chief operating officer of the company. The bottom of the title page requires dated signature with name, designation and official seal of the authorized signatory and institute. The whole responsibility about the correctness of facts and figures presented in the proposal and authenticity of documents solely lies with the authorized signatory. It is a part of the proposal.

Covering Letter: Particularly in government, semi government offices and some private organizations also, there is a system of writing a covering letter for any application or proposal. It starts with the name and address of the sender. It is addressed to the head or the controlling officer of the receiving organization, with or without name, designation and address. It is followed by "subject: ...", which mentions the subject or purpose of the communication, in one line. It may be followed by "Ref:.....", which covers the reference no. of the letter or the advertisement, inviting the offers or proposals. The next one or two paragraphs, mention the purpose of sending the proposal. This type of covering letter is given some outward number with date, which can be referred in the future communication or the reminder.

Summary of Proposal : In the beginning of the proposal, this type of summary highlights the whole proposal in short. It is the overview of the proposal. For initial scrutiny of the proposal, it gives idea about the purpose, nature of proposal, background and strength of the proposer. In initial scrutiny, if it is short listed, the proposal may go for final scrutiny or may be considered for personal visit of the expert team for personal interaction and assessment. If the summary is impressive, it will encourage the receiver to go through the whole proposal. Otherwise, it may be put aside.

Q.15 Describe Technical reports, mentioning types and characteristics of technical reports in detail.

[R.T.U. 2018]

Ans. Technical Reports : Refer to Q.3.

Types of Technical Reports

Technical background report : This type of report is a technical description, to give some background of topics like

Q.1 Define the term GDP.

[R.T.U. 2019]

Ans. Gross Domestic Product (GDP) : It is a broad measurement of a nation's overall economic activity. GDP is the monetary value of all the finished goods and services produced within a country's borders in a specific time period.

GDP includes all private and public consumption, government outlays, investments, additions to private inventories, paid-in construction costs and the foreign balance of trade (exports are added, imports are subtracted). It may be contrasted with Gross National Product (GNP), which measures the overall production of an economy's citizens, including those living abroad, while domestic production by foreigners is excluded. Though GDP is usually calculated on an annual basis, it can be calculated on a quarterly basis as well (in the United States, for example, the government releases an annualized GDP estimate for each quarter and also for an entire year).

PART-B

Q.8 What is circular flow of economic activity? Explain the circular flow in four sector model.

[R.T.U. 2019]

Ans. Concept of circular flow of economic activity:

The economy of any human society must involve the activities of production and consumption. To be noted here is the fact that firms are both producers and consumers in the circular flow chain while the household serves the consumption aspect of our economic chain. Production can be defined as any act of using available economic resources in the process of creating merchandise and services, which match the needs of the consumers. Consumption on the other side can be defined as the act of purchasing merchandise and services either for household use or for use in the process of making new merchandise and services. The latter is the reason behind the qualifying of firms as consumers. It is also to be noted that economic activities also include employment and earning generation. Employment is the acquisition of human resources for the benefits of realizing the production of merchandise and services by a firm. Earning generation as an economic activity includes all the costs incurred by the organization in the process of producing merchandise and services. Such could include costs for purchasing raw materials, labor wages and other expenses incurred by the firm during the production process.

Circular Flow of Economic Activity : The circular flow of economic activity is a model, showing the basic economic relationships within a market economy. It illustrates the balance between injections and leakages in the economy. The circular flow model shows where money goes and what it is exchanged for. The model includes household, businesses and governments.

The four sectors are household, firm, government and foreign. The arrows denote the flow of income through the units in the economy. It also shows injections and leakages (withdrawals). In the four sector economy, international trade is added. International trade includes exports and imports.

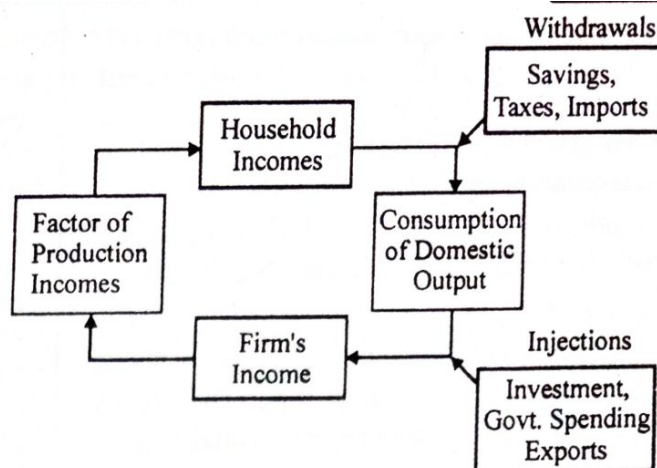


Fig. Circular Flow in four Sector Model

Q.9 Discuss the nature and scope of economics.

Ans. Nature of Economics : When it comes to the nature of economics, there are several things that strike the doors of our mind, i.e. whether the economics is considered as a science or an art. This matter has always been in the great argument regarding various economists.

1. Economics as a science: Science is mainly defined as the systematic study that signifies the cause and effect relationship. In science, it is the collection of facts and figures that should be analyzed correctly. Economics, yet, is treated as a social science due to these following features of science.

(i) It depends on the systematic collection and analysis of facts and figures. Likewise, in economics, all the theories connected with micro and macroeconomics is examined carefully.

(ii) It is based on the formulation of theories and laws.

(iii) It believes in tracing the cause and effect relationship.

(iv) It can also make upcoming forecasts.

(v) It has a scale of measurement, as in economics, 'money' is considered as the measuring rod.

So, it can be clearly stated that the nature of economics is correlated to science. Alike in science, various economic theories are based on logical reasoning. Let's take an example to understand this. In science, there is Newton's law of gravitation and in economics; law of demand signifies that there will be rise in demand if it is decreasing in price.

2. Economics as an art: According to the Marshall, art is the application of information and

knowledge. As, in economics, art acts as a solution to all economic complications. Furthermore, there are several terms in economics like distribution, consumption, and production that offer the guidelines to us, that can be used to sort out the economic problems in society.

Thus, from the above discussion, it can be clearly said that the economics is both a science and an art.

Scope of Economics

The basic concept of economics has a very vast scope and to understand this crucial aspect, it is really necessary to go through this concern very carefully.

Scope of economics can be classified broadly into two categories:

1. Microeconomics: Microeconomics is the study of examining every individual economic activity, industries, and their interactions. It mainly observes how a person earns and spends his income. Besides it, the nature of microeconomics has certain key areas that must be taken into consideration.

(i) Elasticity: Elasticity is used to determine the ratio of change in the proportion of one variable to the change in the proportion of another variable. Price elasticity of demand and the income elasticity of demand, the price elasticity of supply etc. are commonly used elasticities in the market.

(ii) Theory of production: In this study of production, the input is converted into output efficiently. Production can include storing, shipping, packaging, and manufacturing.

(iii) Cost of production: In this theory, it states that the object price is determined by the price of resources. The cost can be comprised of land, labor, capital and technology.

(iv) Monopoly: A monopoly can be defined as the state where a single firm is the one and only supplier of a specific commodity.

(v) Economics of information: Information economics is a kind of theory, which shows how information can affect economic decisions.

(vi) Oligopoly: It is termed as the situation where the small numbers of sellers dominate an industry or a market.

2. Macroeconomics : Refer to Q.2.

(i) Output and income: Output can be defined as the total income that generates from the sold commodity. It is usually measured by the Gross Domestic Product (GDP). Moreover, there are many reasons for the rise in output: technological advancement, human capital and better education.

(ii) Unemployment: It is typically measured by the overall employment rate, which means the ratio of workers without employment in the workforce. But the people who are pursuing their education and retired are excluded from this unemployment rate.

(iii) Inflation and deflation: In general terms, increase in price refers to inflation; while on the other hand, decrease in price refers to deflation in the economy. These fluctuations in the price can be easily measured by using price indexes.

Q.10 Discuss the methods of economic analysis.

Ans. 1. Deductive method : Deduction means interpretation or implication from general to the particular or from the universal to the individual. The deductive method develop new finale from elementary hypothesis or from truth recognized by other methods. It engrosses the process of analysis from certain laws or principles, which are supposed to be true to the analysis of facts. This involves four steps.

(i) Selecting the Problem : The problem which a researcher plump for, for enquiry must be fixed clearly. It may be very broad like redundancy, scarcity, price rises etc. or tapered relating to an industry.

(ii) Formulating Assumptions : The next step in deduction is the outlining of postulations which are the basis of propositions. To be productive for survey, the assumptions must be general.

(iii) Formulating Hypothesis : The next step is to formulate a hypothesis on the basis of logical reasoning whereby conclusions are drawn.

(iv) Testing and Verifying Hypothesis : This is the final step of deductive method, to test and verify the hypothesis. For this purpose, economists use statistical and econometric methods. Verification consists in corroborating whether the hypothesis is in agreement with facts.

2. The Inductive Method : Induction is the process of way of thinking from a part of the intact, from details to general from the individual to the widespread. The inductive method involves the following steps.

(i) The Problem : In order to appear at a simplification relating to an economic occurrence, the dilemma should be correctly selected and clearly stated.

(ii) Data : The second step is the compilation, enumeration, categorization and investigation of data by using suitable statistical techniques.

(iii) Observation : Data are used to make surveillance about particular facts relating to the problem.

(iv) Generalisation : On the basis of inspection, generalised is logically derived which establishes a general truth from particular facts.

situations at different points of time are compared, (or in other words, two static situations are compared) is called comparative statics. This can be seen from the figure.

At OM demand and supply the equilibrium price was PM but now demand and supply have changed to D_1, D_1 and S_1, S_1 and, therefore, equilibrium price also changed to P_1, M_1 . This should be noted that price PM was determined at certain point of time and price P_1, M_1 is determined at some other point to time. When these 'two different points of time' are compared it is known as 'comparative statics'. Here we are comparing two different 'STILL' pictures. There can happen a series of static models.

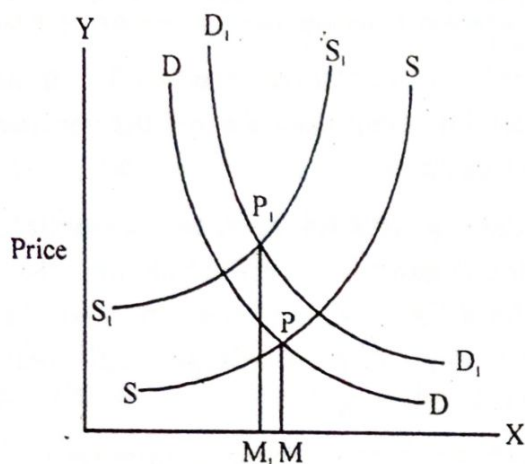


Fig. : Comparative Statics

PART-C

Q.13 Explain the various methods of measuring Nation Income.

[R.T.U. 2019]

Ans. Methods of Measuring Nation Income

There are three methods of measuring Nation Income.

- Product (or) Production Method
- Income method
- Expenditure Method

(a) Product (or) Production Method : It is also known as value added (or) output method. In this method the value added by various services and production goods are measured.

For the value added of the goods to be observed, the expenditure that is incurred on the intermediate goods is deducted from the goods itself.

These intermediate goods include unfinished goods which are purchased from enterprises, raw materials and the value of output produced by that enterprise.

(b) Income Method : To calculate income from distribution side, the income method is used. Thus, when the income is measured, during the time of distribution, it appears as the income paid (or) received by the people in that country.

In this method, the income is obtained by adding up all the incomes of the people in the country.

Services like land and capital which produces its own services are included in this method.

Thus, in this method, the payments done through transfer are not included in nation income. Smuggling money or hawala money is also not included.

(c) Expenditure Method : It includes all the expenditures done by the people in that country. The nation income is calculated based on the total expenditures made of goods and services during that year.

All types of expenditures are calculated in this method. This method is used mainly to calculate the incomes in the construction sector.

Some precautions in this method are it does not include the expenditure on second-hand goods, old bonds and shares.

Q.14 Explain the Principles of economics as explained by the circular flow of economic activities.

Ans. The economic progress and sustainability of any society is highly dependent on production and consumption. It is to be noted here that the production process is determined by the availability of economic resources such as physical facilities, labor and capital investment. Land is the physical resource required for the construction of the factory facilities and agricultural activities. It should also be noted that the land is the source of most of the economic resources required by firms in the production process. Such resources as electricity and minerals are found in the soil, making land a crucial factor in the production process.

The availability of qualified and reliable labor force is another basic requirement for sustainable production in any firm. It is the human resource that factors greatly in ensuring the efficient operation of any organization. It should also be noted that by providing employment, the firms are initiating the reciprocal factor of economic activities. With good wages, workers as consumers will increase their purchasing power for merchandise and services thus increasing the overall earnings margins for

DEMAND AND DEMAND ANALYSIS

2

PREVIOUS YEARS QUESTIONS

PART-A

Q.1 Define the Giffen Goods. [R.T.U. 2019]

Ans. Giffen goods (most inferior goods) are those inferior goods for which 'income effect' of change in price is negative and is greater than the substitution effect. Therefore, the demand of Giffen goods increases with rise in price and decreases with fall in their price.

Q.2 What is the purpose of Demand Forecasting? [R.T.U. 2018]

Ans. Purposes of Forecasting

Forecasting, according to time may be two types- short term and long term. The purposes of forecasting may be different according to types of forecasting.

(a) Short-term Forecasting

- (i) Evolving suitable production policy so as to avoid the problem of over-production and under-production.
- (ii) Helping the firm in reducing cost of purchasing raw materials and controlling inventory by assuring regular supply of raw materials.

(b) Long-term Forecasting

- (i) **New unit planning or expansion of an existing unit :** A long term demand forecasting helps to plan for new units or at the same time existing units to

expand their activities. A multi-product firm must determine total demand situation and the demand for different items.

- (ii) **Planning for long term financial requirements :** If the demand is more and it takes long time then such long term financial requirements could be planned and funds may be arranged and made available at the right time.

Q.3 Explain law of supply.

Ans. Law of supply explains the relationship between price of a commodity and its quantity supplied. Other things remaining the same, when price of a commodity falls the quantity supplied decreases and vice versa. Thus there is a positive relationship between price of a commodity and its supply. One thing should be understood here is that law of supply is only an indicative statement, not a quantitative statement. It means that when the price will change the quantity supplied will also change in the direction of price change but will not explain 'how much'. This quantitative relationship will be explained under the heading 'elasticity of supply'. For example, if the price of a commodity increases by 10% the quantity supplied, as per the law of supply, will also increase but will not explain how much the quantity supplied will increase.

Q.4 Explain Determinants of Elasticity of Supply

Ans. The elasticity of supply is determined by the following factors :

- 1. **Time Period :** Elasticity of supply depends upon the time which the seller requires to bring about adjustment in supply. In the very short period where the supply of

PART-B

Q.6 What is Elasticity of Demand? What are the factors affecting the Elasticity of Demand?

[R.T.U. 2019]

Ans. "Elasticity of demand is the responsiveness for the quantity demanded of a commodity to changes in one of the variables on which demand depends. In other words, it is the percentage change in quantity demanded divided by the percentage in one of the variables on which demand depends."

Various factors which affect the elasticity of demand of a commodity are following :

1. Nature of Commodity : Elasticity of demand of a commodity is influenced by its nature. A commodity for a person may be a necessity, a comfort or a luxury.

- When a commodity is a necessity like food grains, vegetables, medicines, etc., its demand is generally inelastic as it is required for human survival and its demand does not fluctuate much with change in price.
- When a commodity is a comfort like fan, refrigerator, etc., its demand is generally elastic as consumer can postpone its consumption.
- When a commodity is a luxury like AC, DVD player, etc., its demand is generally more elastic as compared to demand for comforts.
- The term 'luxury' is a relative term as any item (like AC), may be a luxury for a poor person but a necessity for a rich person.

2. Availability of Substitutes : Demand for a commodity with large number of substitutes will be more elastic. The reason is that even a small rise in its prices will induce the buyers to go for its substitutes. For example, a rise in the price of Pepsi encourages buyers to buy Coke and vice-versa.

Thus, availability of close substitutes makes the demand sensitive to change in the prices. On the other hand, commodities with few or no substitutes like wheat and salt have less price elasticity of demand.

3. Income Level : Elasticity of demand for any commodity is generally less for higher income level groups in comparison to people with low incomes. It happens because rich people are not influenced much by changes in the price of goods. But, poor people are highly affected

by increase or decrease in the price of goods. As a result, demand for lower income group is highly elastic.

4. Level of Price : Level of price also affects the price elasticity of demand. Costly goods like laptop, Plasma TV, etc. have highly elastic demand as their demand is very sensitive to changes in their prices. However, demand for inexpensive goods like needle, match box, etc. is inelastic as change in prices of such goods do not change their demand by a considerable amount.

5. Postponement of Consumption : Commodities like biscuits, soft drinks, etc. whose demand is not urgent, have highly elastic demand as their consumption can be postponed in case of an increase in their prices. However, commodities with urgent demand like life saving drugs, have inelastic demand because of their immediate requirement.

6. Number of Uses : If the commodity under consideration has several uses, then its demand will be elastic. When price of such a commodity increases, then it is generally put to only more urgent uses and, as a result, its demand falls. When the prices fall, then it is used for satisfying even less urgent needs and demand rises.

For example, electricity is a multiple-use commodity. Fall in its price will result in substantial increase in its demand, particularly in those uses (like AC, Heat convector, etc.), where it was not employed formerly due to its high price. On the other hand, a commodity with no or few alternative uses has less elastic demand.

7. Share in Total Expenditure : Proportion of consumer's income that is spent on a particular commodity also influences the elasticity of demand for it. Greater the proportion of income spent on the commodity, more is the elasticity of demand for it and vice-versa.

Demand for goods like salt, needle, soap, match box, etc. tends to be inelastic as consumers spend a small proportion of their income on such goods. When prices of such goods change, consumers continue to purchase almost the same quantity of these goods. However, if the proportion of income spent on a commodity is large, then demand for such a commodity will be elastic.

8. Time Period : Price elasticity of demand is always related to a period of time. It can be a day, a week, a month, a year or a period of several years. Elasticity of demand varies directly with the time period. Demand is generally inelastic in the short period.

It happens because consumers find it difficult to change their habits, in the short period, in order to respond to a change in the price of the given commodity. However, demand is more elastic in long run as it is comparatively easier to shift to other substitutes, if the price of the given commodity rises.

9. Habits : Commodities, which have become habitual necessities for the consumers, have less elastic demand. It happens because such a commodity becomes a necessity for the consumer and he continues to purchase it even if its price rises. Alcohol, tobacco, cigarettes, etc. are some examples of habit forming commodities.

Finally it can be concluded that elasticity of demand for a commodity is affected by number of factors. However, it is difficult to say, which particular factor or combination of factors determines the elasticity. It all depends upon circumstances of each case.

Q.7 Following table exhibits demand and supply condition of a commodity at different price level –

Price (Rs.)	Demand ('000/month)	Supply ('000/month)
110	-	1000
100	40	900
90	80	800
80	120	700
70	160	600
60	200	500
50	240	400
40	280	300
30	320	200
20	360	100
10	400	-

(a) Draw the demand and supply curves.

(b) What is the equilibrium price and quantity?

[R.T.U. 2019]

Ans.(a) Demand and Supply Curves

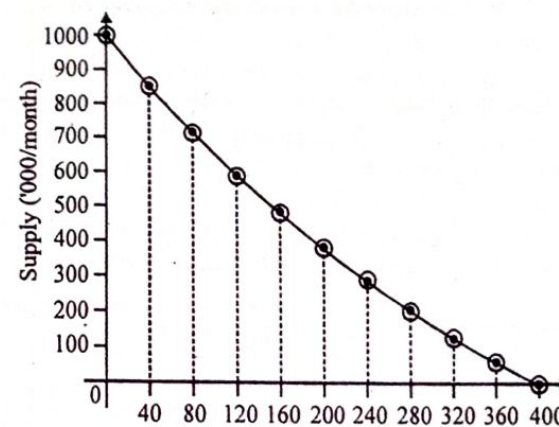


Fig. Supply-Demand Curve

Q.8 What is demand function? How do you determine it? [R.T.U. 2018]

Ans. Demand function : In economics, demand means the willingness and ability of a consumer to have the commodity at a certain price during a particular time period. The quantity desired to be purchased may be different from the quantity of commodity actually purchased by the consumer. As quantity demanded is a flow concept⁶, the relevant time dimension has to be mentioned which will indicate the quantity demanded per unit of time. Demand for a commodity, say X, is a relationship between the price and the quantity demanded of commodity X, *ceteris paribus*. Here, other things mainly include income of the consumer, prices of substitute and complementary products, consumer's tastes, and preferences that influence the decision of the consumer to purchase the product.

If Q_x and P_X denote the quantity demanded and the price per unit of commodity X, respectively, then, other things remaining constant, the demand function will be written as $Q_dX = f(P_X)$. This means that any change in the price of commodity X will result in a corresponding change in the quantity demanded of commodity X.

Determine Demand Function : Demand function with more than one independent variable at a time equation (1). In economic literature, it has become customary to distinguish between a demand function and the Law of Demand. The former includes the variables other than price too. A hypothetical example of the demand function is given below.

$$D_x = 10.0 - 2.0 P_x + 1.5 Y + 0.8 P_s - 3.0 P_c + 1.5 A \quad \dots(1)$$

where,

D_x = Quantity demanded of brand X tea (in millions of packets)

P_x = Price per packet of tea (in rupees)

Y = Personal disposable income (in crores of rupees per year)

P_s = Price per packet of other brands of tea (in rupees)

P_c = Price of a complement

A = Advertisement expenditure (in rupees)

- (A) If the values of independent variables for the year 1999 are given as : $P_x = 20$, $Y = 40$, $P_s = 25$, $P_c = 10$ and $A = 1$, the demand for brand X tea can be estimated by replacing the values as follows :

$$D_x(1999) = 10 - 2(20) + 1.5(40) + 0.8(25) - 3(10) + 1.5(1) = 21.5 \text{ million tea packets}$$

- (B) Assuming the same values of independent variables, the demand function with each independent variable separately can also be estimated. For example :

$$1. D_x = f(P_x) \parallel Y, P_s, P_c, A \parallel \text{(i.e., Other variables remains unchanged)}$$

$$D_x = 61.5 - 2P_x \quad \dots(2)$$

$$2. D_x = f(Y) \parallel P_x, P_s, P_c, A \parallel$$

$$= -38.5 + 1.5Y \quad \dots(3)$$

$$3. D_x = f(P_s) \parallel P_x, Y, P_c, A \parallel$$

$$= 1.5 + 0.8P_s \quad \dots(4)$$

$$4. D_x = f(P_c) \parallel P_x, Y, P_s, A \parallel$$

$$= 51.5 - 3.0P_c \quad \dots(5)$$

$$5. D_x = f(A) \parallel P_x, Y, P_s, P_c \parallel$$

$$= 20.0 + 1.5A \quad \dots(6)$$

Here, by replacing the values of the respective independent variables, the estimated demand for tea remains 21.5 million packets in all the cases.

Q.9 What do you mean by Income Elasticity of Demand? Distinguish its various types.

OR

Write short note on 'Income Elasticity of Demand'.

OR

Explain the concept of income elasticity of demand and explain its role in business decisions.

Ans. Income elasticity of demand is the ratio of proportionate change in quantity demanded of a commodity to proportionate change in income, other things being constant. It is measured by the following formula :

$$\text{Income Elasticity of Demand} = \frac{\text{Proportionate change in quantity demanded}}{\text{Proportionate change in income}}$$

$$ey = \frac{\Delta Q}{Q} \times \frac{Y}{\Delta Y} \text{ or } ey = \frac{\Delta Q \cdot Y}{\Delta Y \cdot Q}$$

Where :

ey = Income elasticity of demand

ΔQ = Change in quantity demanded.

ΔY = Change in income

Q = Quantity demanded at previous income

Y = Previous income or income before change.

Income elasticity of demand is, thus, the degree of responsiveness of demand to a change in income. Generally, the income elasticity of demand for most goods is positive, i.e., their demand increases with increase in income and decreases with decrease in income. Demand for inferior goods, however, shows a negative relation to changes in income.

Income elasticity of demand can be expressed in terms of change in expenditure on the commodity as a result of change in income. We know that :

Expenditure = PQ (Quantity $\times$ Price)

$$ey = \frac{\Delta Q}{Q} \times \frac{Y}{\Delta Y}$$

By multiplying both ΔQ and Q by P , we get :

$$ey = \frac{\Delta Q \cdot P}{Q \cdot P} \times \frac{Y}{\Delta Y}$$

If we denote expenditure incurred on the commodity by 'X', the above equation will become :

$$ey = \frac{\Delta X}{X} \times \frac{Y}{\Delta Y}$$

(because $Q \cdot P = X$ and $\Delta Q \cdot P = \Delta X$)

$$\text{Or } ey = \frac{Y \cdot \Delta X}{X \cdot \Delta Y}$$

Important Numerical Values of Income Elasticity Of Demand

The three numerical values of income elasticity of demand are very interesting. These are as follows :

- When the whole of any increase in income of consumer is spent on the commodity in question,

then income elasticity will be equal to $\frac{1}{KX}$. Where

KX is the proportion of the consumer's income spent on any good, X, whose income elasticity is being measured. When income elasticity (ey) = $\frac{1}{KX}$

then the entire increase in consumer's income is spent on increasing his purchases of commodity 'X'.

Proof :

We know that : $ey = \frac{Y \cdot \Delta X}{X \cdot \Delta Y}$

When the whole of any increase in income of a consumer is spent on the commodity then $\Delta X = \Delta Y$. Substituting ΔY for ΔX in the above equation, we get :

$$ey = \frac{Y \cdot \Delta Y}{X \cdot \Delta Y} = \frac{Y}{X} = \frac{1}{X/Y} = \frac{1}{KX}$$

(Because $\frac{X}{Y}$ is the proportion of income spent on the commodity X)

- Another most interesting value of income-elasticity of demand is when it is equal to one. This means that the proportion of the consumer's income spent on the commodity in question is exactly the same both before and after income increases. If the $ey > 1$, it means that proportion of consumer's income spent on the commodity increase when he becomes richer. On the other hand, if $ey < 1$, the proportion of consumer's income spent on the commodity falls when income rises. Therefore, unit income elasticity of demand is an important dividing line. From this it seems reasonable to conclude that if $ey > 1$ then the commodity, in some sense, is a luxury. On the other hand if $ey < 1$ then the commodity is a necessity. One cannot, of course, give a precise definition of necessities or luxuries in terms of income elasticities of demand. But the notion is useful one.
- Zero income-elasticity of demand is also an interesting value. Income elasticity of demand is zero when the demand for a commodity remains unchanged even after increase in income. If the income elasticity of demand is less than zero, i.e., negative then the commodity is inferior. If income elasticity of demand is more than zero, i.e., positive then the commodity will be normal or luxury.

Income Elasticity of Demand and Business Decisions

- As most of the goods are superior goods, economic growth (means percentage growth in real GNP per capital) is closely associated with the increase in their sales. However, an individual businessman is

interested in knowing whether the sale of his product will be responsive to the economic growth, and the income elasticity of demand for his product will enable him to get the answer. In this connection, there are two rules:

- If income elasticity is less than unity, the sales of his product will increase but at a slower rate than the rate of economic growth.
 - If, on the other hand, income elasticity of the product is more than one, the sale of his product will increase faster than the rate of economic growth.
- Similarly, in times of recession, or times when the government squeezes the credit or reduces the level of economic activity, the sale of the firm's product will decline at a slower or faster rate than the fall in general economic activity depending upon whether income elasticity of demand is less than or more than unity.
 - As the study of income elasticity of demand is useful in forecasting the effect of changes in economic activity on demand, the managerial economist should regularly and periodically review the effects of change in income on the sale of the company's product. If the area of business of the firm is large enough, the managerial economist should study the income elasticity of demand separately for each region or state. The sale forecasting for each region/state depends upon such analysis.

Although income elasticity of demand helps the businessman to forecast the sale of his product for a given period or region, the forecasting is limited to several difficulties –
 - The relation between changes in income and demand is not straight forward. Much depends upon the suddenness or permanence of a change in income. If the change in income is temporary no change in plan should be stipulated as it takes time to plan the change.
 - Sales of a product are influenced not only by the changes in income but by several other factors as well.
 - With the passage of time, the relationship between income and demand also changes.

Q.10 Define Cross Elasticity of Demand and give its importance.

OR

Write short note on Cross Elasticity of Demand and Substitution Elasticity of Demand.

Ans. The responsiveness of demand to change prices of related commodities is called cross elasticity of demand. Prof. Watson defines it as, "Cross elasticity of demand is the rate of change in quantity associated with a change in the price of related goods".

Thus, cross-elasticity of demand is the responsiveness of demand for commodity X to change in price of commodity Y and is represented as follows:

$$Ec = \frac{\text{Proportionate change in the quantity demanded of commodity X}}{\text{Proportionate change in the price of commodity Y}}$$

Symbolically:

$$Ec = \frac{\Delta q_X}{\Delta p_Y} \cdot \frac{p_Y}{q_X}$$

The relationship between X and Y commodities may be substitutive as in case of tea and coffee or complementary as in the case of ball pens and refills. Main measures of cross elasticity with description are as follows:

- Cross elasticity = Infinity – Commodity X is nearly a perfect substitute for commodity Y
- Cross elasticity = Zero – Commodities X and Y are not related
- Cross elasticity = Negative – Commodities X and Y are complementary

Thus, if Ec approaches infinity, it means that commodity X is nearly a perfect substitute for commodity Y. On the other hand, if Ec approaches Zero it would mean that the two commodities in question are not related at all. Ec shall be negative when commodity Y is complementary to commodity X.

The two commodities may be substitutes or complementary to each other.

- Substitute Goods :** In case, the two commodities are substitutes to each other such as tea and coffee,

the cross elasticity of demand will be positive. The fall in the price of one commodity may result in the increase of demand of that commodity or in other words, the demand of other competitive goods will decrease and vice versa. Thus, the relationship is positive. In a case, where two commodities are perfectly substitutes, the cross elasticity will be plus infinity. The demand curve in such cases goes upward.

- Complementary Goods :** Complementary goods are those goods which are used side by side. If two commodities are complementary, their demands will increase or decrease together. For example, if X and Y are complementary, a fall in the price of X commodity will increase the demand of X commodity, (with the operation of law of demand) and consequently the demand of Y will also increase without any change in prices of Y commodity. The cross elasticity in such case is negative. In the case of perfectly complementary goods, cross elasticity will be minus infinity. The demand curve in such cases goes downward.

Importance of Cross Elasticity of Demand

The knowledge of cross elasticity of demand is important for a businessman/managerial economist in pricing decisions. How the product should be priced, below or above the price of competitors product, is a decision of wide importance. The producer estimates the effect of price change of his product on the demand of competitive articles and consequently on the demand of his own product. He should therefore, adjust the price of his product in a way, it does not help shift the demand to other commodities. The businessman should also study the effect of price change of other commodities on the sale/demand of his product. By studying the cross elasticity of demand, he is to revise the price of his product. The study of cross elasticity reveals the relationship of two products i.e. whether they are competitive or complementary or they are not at all related. In a competitive market, the intensity of monopoly and degree of competition can be gauged with the help of the study of cross elasticity of demand.

Elasticity of Substitution

Elasticity of substitution measures the extent to which one goods (X) can be substituted for another goods (Y) as a result of goods X, becoming relatively cheaper when the level of satisfaction of the consumer remains the same. In other words, elasticity of substitution is a

relative measure of substitution effect. The mathematical measure of elasticity of substitution (es) between any two goods X and Y can be given by the following formula :

$$es = \frac{\text{Relative increase in the ratio of two goods possessed X/Y}}{\text{Relative change in Marginal Rate of Substitution of X for Y (MRS}_{XY})}$$

$$\text{Or } es = \frac{\Delta\left(\frac{QX}{QY}\right)}{\frac{QX}{QY}} \div \frac{\Delta\left(\frac{\Delta Y}{\Delta X}\right)}{\frac{\Delta Y}{\Delta X}}$$

Where :

es = elasticity of substitution

$\frac{QX}{QY}$ = Original proportion between the quantities of goods X and Y

$\Delta\left(\frac{QX}{QY}\right)$ = Change in the proportion of X and Y goods

$\frac{\Delta Y}{\Delta X}$ = Original rate of substitution of goods X for Y (MRS_{XY})

$\Delta\left(\frac{\Delta Y}{\Delta X}\right)$ = Change in MRS_{XY}

The following conclusions are drawn from elasticity of substitution :

1. The elasticity of substitution between perfect substitute goods is infinite. But in real life perfect substitutes are not present and if present then the two goods are considered as one goods. Goods substitutes are found in real life.
2. In case of goods close substitutes, the elasticity of substitution is high because here marginal rate of substitution is quite high.

3. > Elasticity of substitution is zero where no substitution is possible between two goods or two goods are perfect complement of each other. The perfect complements goods are used in fixed proportion. Therefore, in case of perfect complements the marginal rate of substitution is zero. But in real life perfect complement goods are rarely found because some substitution is always possible between the goods. Thus, when two goods are bad substitutes the elasticity of substitution is near to zero.

Q.11 Discuss the inter-relationship of Price Elasticity, Income Elasticity and Substitution Elasticity of demand.

Ans. Relationship Between Price Elasticity, Income Elasticity and Substitution Elasticity of Demand

We know that 'price effect' is the combination of 'income effect' and 'substitution effect'. Price elasticity of demand is a measure of price effect. Similarly, income elasticity of demand measures the income effect of change in price on the demand of a commodity and substitution elasticity of demand measures the substitution effect on demand of change in price. Thus, price elasticity of demand is a combination of income elasticity of demand and substitution elasticity of demand. In other words, price elasticity of demand is a compromise between income-elasticity and substitution-elasticity of demand. The relationship between three elasticities of demand can be expressed by the following equation :

$$ep = KX.ey + (1 - KX) es.$$

Where :

ep = price elasticity of demand for goods, X

ey = income elasticity of demand

es = substitution elasticity of demand

KX = the portion of consumer's income spent on goods X in question.

(1 - KX) = the proportion of income spent on other goods.

The first part of the equation, i.e., KX. ey shows the impact of income-effect on the price elasticity of demand. the income effect of change in price depends on two things: (a) the proportion of income spent on the

Q.17 What is demand function ?

Ans. Demand Function : Refer to Q.8.

Q.18 Briefly discuss the price elasticity of demand.

Ans. Price Elasticity of Demand : Demand elasticity measures the responsiveness of quantities demanded of an item relative to changes in its price. Supply elasticity gauges the responsiveness of quantities supplied of an item relative to changes in its price. Demand elasticity is usually called price elasticity of demand to distinguish it from measures of the sensitivity of amounts bought to changes in economic variables other than the item's price. Similarly, supply elasticity is commonly referred to as price elasticity of supply.

Price elasticity of demand is calculated by dividing the percentage change in the quantity demanded of an item by the percentage change in the price that caused it, other things being equal. The following formula can be used to calculate price elasticity of demand for small changes in price.

$$\text{Price elasticity of demand} = \frac{dQ/Q}{dP/P}$$

where Q is the initial quantity demanded, dQ is the change in quantity demanded caused by the change in price dP, and P is the initial price of the item.

Because the demand for a good is likely to fall as its price rises, the price elasticity of demand will be a negative number indicating the percentage change in the quantity demand of an item resulting from a 1 percentage change in its price. Price elasticity of demand is equal to the inverse of the negative slope of a demand curve (dQ/dP) multiplied by the ratio of the initial price of the item to its quantity demanded.

The absolute value of the price elasticity of demand can range from zero to infinite. When this value is between 0 and 1, demand is said to be inelastic; when it is greater than 1, demand is said to be elastic. The demand for an item is said to be of unitary elasticity if the absolute value of its price elasticity of demand is exactly equal to 1.

Q.19 Discuss the concept of elastic and inelastic demand.

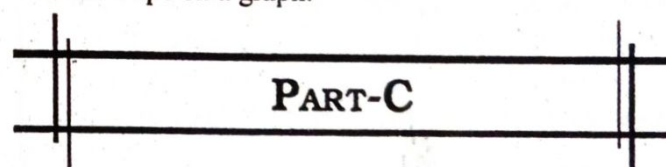
Ans. Concept of elastic and inelastic demand: Inelasticity and elasticity demand are the respective end ranges for the formulaic comparison of price and demand for a given product in an economy. If the change in

demand for a given product corresponds closely to the change in price for that product, the demand is considered to be elastic. If the change in demand for a given product does not correspond closely to a change in price for that product, the demand is considered to be inelastic.

Elasticity of demand is calculated by dividing the percent change in quantity demanded by the percent change in price. If the elasticity quotient is greater than or equal to 1, the demand is considered to be elastic. If the elasticity quotient is less than 1, the demand is considered to be inelastic. When the data is graphed, elasticity of demand has a negative slope. An elastic demand is displayed as a more horizontal, or flatter, slope. An inelastic demand is displayed as a more vertical, or steeper, slope.

The most utilized example of a product with inelastic demand is salt. The human body requires a specific amount of salt per pound of the body weight. Too much or too little salt could cause illness or even death. Therefore the demand for salt, changes very little with the price. Salt has an elasticity quotient close to zero and a steep slope on a graph.

A common example of an elastic product is gasoline. As the price of gas increases and falls with the international market, the demand (the distance driven by the population) rises and falls in near direct correlation. Gasoline has an elasticity quotient of 1 or greater and has a flatter slope on a graph.



Q.20 How does the elasticity of supply is measured.

OR

Explain how do you measure elasticity of demand.

[R.T.U. 2018]

Ans. As already stated, elasticity of supply measures the responsiveness of the quantity supplied to a change in the commodity's own price. It is defined as

$$E_s = \frac{\text{Percentage change in Quantity supplied}}{\text{Percentage change in price}}$$

$$E_s = \frac{\frac{\Delta Q_s}{Q_s}}{\frac{\Delta P}{P}}$$

To understand the concept of 'Elasticity of Demand', we must know the meaning of the word 'elasticity'. Elasticity is the ratio of relative change in a dependent variable to the relative change in independent variable. There are three main determinants of demand, namely price of the commodity, income and price of related goods. Therefore, elasticity of demand refers to the degree of responsiveness of demand of commodity to a change in its price, income or price of related goods. Accordingly, there are three kinds of elasticity of demand- (i) price elasticity of demand, (ii) income elasticity of demand and (iii) cross elasticity of demand.

Q.23 What is price elasticity of demand? Give its various types or degrees. How it can be measured?

Ans. The demand of a commodity is closely related to its price. Price elasticity of demand quantity change in demand as a result of change in price. The degree of responsiveness of demand for a commodity to a change in its price is known as price elasticity of demand or elasticity of demand as it is generally known. In other words, price elasticity of demand is the ratio of proportionate change in the quantity demanded of a commodity to proportionate change in its price. Thus, it represent the rate of change in demand as a result of change in price of the commodity. The price elasticity of demand can be expressed by the following formula :

$$\text{Elasticity of Demand} = \frac{\text{Proportionate change in quantity demanded}}{\text{Proportionate change in price}}$$

The price elasticity of demand, if other thing remain the same, is always negative, signifying the inverse relationship between price and demand. In practice, however, the minus sign is ignored as the inverse relationship is implied.

Types or Degrees of Price Elasticity of Demand

A small fall in the price of a product may lead to a considerable increase in the quantity demanded, but sometimes even a considerable fall in price may not lead to any increase in demand. The degree of responsiveness of demand to small change in price differ from commodity to commodity. Degrees of elasticity of demand is classified into five types:

1. Unit Elasticity : Demand is unit elastic when percentage change in quantity demand and percentage change in price are equal. In case of unit elastic demand, the demand curve is a Rectangular Hyperbola. In practice it is difficult to find such commodities as have a demand curve whose elasticity is unit throughout.

2. Relatively elastic demand (ed>1) : The demand is relative elastic or more than unity when relative change in quantity demanded is more than the relative change in price. In such cases the demand curve is of less slope.

3. Relatively inelastic demand (ed<1) : Demand is said to be relatively inelastic or less than unity when proportionate change in demand is less than proportionate change in price. In such cases the slope of demand curve falls rapidly.

4. Perfectly inelastic demand (ed = 0) : When there is no change in demand as a result of increase or decrease in price then the demand is perfectly inelastic. The demand curve is vertical on OX axis.

5. Perfectly elastic demand (ed = ∞) : The demand is perfectly elastic when even a small change in price cause and infinite large change in amount demanded. A small rise in price on the part of a seller reduces the demand to zero. In such cases the demand curve is parallel to OX axis.

Measurement of Price Elasticity of Demand

Price elasticity of demand can be measured by four methods. These methods are as follows :

1. Proportionate Method : According to this method, price elasticity of demand is estimated by dividing the percentage change in quantity demand by the percentage change in the price of commodity. If the percentage change in quantity demanded is less than the percentage change in price, the elasticity of demand will be less than one. The formula is :

Price Elasticity of Demand (ep)

$$= \frac{\text{Percentage change in Quantity demanded}}{\text{Percentage change in price}}$$

$$ep = (-) \frac{\Delta Q}{Q} \times \frac{P}{\Delta P} = \frac{\Delta Q}{\Delta P} \times \frac{P}{Q} = \frac{\Delta Q.P}{\Delta P.Q}$$

Where

ep = price elasticity of demand.

$$\Delta Q = \text{change in quantity demanded} \\ (\Delta Q = Q_1 - Q).$$

Q = quantity demand before change in price and

Q₁ = quantity demanded after change in price.

$$\Delta P = \text{change in price } (\Delta P = P_1 - P).$$

P = Initial price and P₁ = changed price.

Negative sign(-) symbolises the inverse relationship between price and demand. While calculating price elasticity of demand negative sign is ignored. Many a times the formula given above is modified as:

$$ep = \frac{\Delta Q}{Q + Q_1} \div \frac{\Delta P}{P + P_1}$$

$$\text{or } ep = \frac{\Delta Q}{Q + Q_1} \times \frac{P + P_1}{\Delta P}$$

In the above formula, to calculate price elasticity of demand, instead of taking Q and P as denominator, the average of Q and Q₁ and P and P₁ are taken as denominator. The modification is because of two reasons. Firstly, the modification ensure reversibility. Here reversibility means that the coefficient of elasticity of demand will remain the same whether we take Q and P or Q₁ and P₁ as the base of calculating elasticity of demand. Secondly, the first formula point elasticity of demand and the second formula measures the elasticity of demand between two finite points on the demand curve. In other words the second formula gives us the 'Arch Elasticity of Demand'.

2. Total Expenditure Method : With the change in price of a commodity total expenditure incurred (total revenue received by firm) on the commodity may increase, decrease or remain constant. It depends upon the relative change in amount demanded of the commodity as a result or relative change in its price. If the proportionate increase in amount demanded is more than the proportionate fall in price then the total revenue (total expenditure on the commodity) of the firm will increase. But when the proportionate increase in amount demanded and proportionate fall in price are equal then the total revenue (total exp.) will remain same. On the other hand, when the percentage increase in amount demanded is less than

percentage fall in price then the revenue (Total exp on the goods) of the firm will fall. Thus, this method gives three co-efficients of elasticity of demand which are as follows:

- Greater than Unit Elastic Demand ($ep > 1$)
- Unit Elastic Demand ($ep = 1$)
- Less than Elastic Demand ($ep < 1$)

(a) **Greater than unit elastic demand ($ep > 1$)** : Elasticity of demand is greater than unit when total expenditure incurred by the buyers on the commodity and price changes in the opposite direction. It means elasticity of demand is greater than are when total expenditure on the commodity increases with a fall in price and decreases with a rise in price of the commodity. It is illustrated with the help of following example:

Demand Schedule of Sugar

Price (Rs.) per kg	Q (kg.)	Total Expenditure (Q.P)
10	15	150
9	20	180
8	25	200

(b) **Unit elastic demand ($ep = 1$)** : Demand is unit elastic when total expenditure remains the same with rise or fall in price. It is illustrated by the following example:

Demand Schedule of 'X' Commodity

Price (Rs.) per kg	Q (kg.)	Total Expenditure (Q.P)
2	100	200
4	50	200
8	25	200

(c) **Less than unit elastic demand ($ep < 1$)** : If price and total expenditure on the commodity change in the same direction then elasticity of demand is less than unit. In other words, if with a fall in the price of commodity total expenditure on it also falls or with an increase in price of the commodity total expenditure on it also increases, then elasticity of demand is less than unit. Example:

Demand Schedule of 'Y' Commodity

Price (Rs.) per kg	Q (kg.)	Total Expenditure (Q.P)
2	9	18
3	8	24
4	7	28

From the above discussion it follows that total expenditure incurred on the commodity and its price stand in an important relationship. This relationship is of vital importance to business firms to decide price policies.

3. Point Method or Geometric Method : This method measures the elasticity of demand on different points of a demand curve. It is a variant of 'proportionate method'. To measure elasticity of demand we take a straight line demand curve as shown in the fig. below. In the fig. AB is the demand curve.

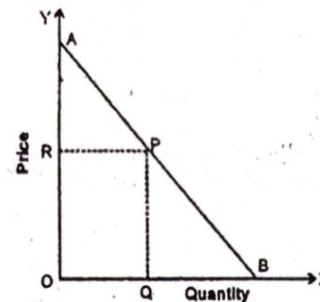


Fig.

We know that

$$ep = \frac{P \cdot \Delta Q}{Q \cdot \Delta P}$$

The term $\frac{\Delta Q}{\Delta P}$ is the reciprocal of the slope of demand curve. The slope of AB demand curve is PQ/QB in the fig. price is PQ and quantity demanded at PQ price

is OQ and $\frac{\Delta Q}{\Delta P} = \frac{QB}{PQ}$. Therefore:

$$ep = \frac{P \cdot \Delta Q}{Q \cdot \Delta P} = \frac{QB}{PQ} \times \frac{PQ}{OQ} = \frac{QB}{OQ}$$

$$\text{And } \frac{QB}{OQ} = \frac{QB}{RP} \quad \text{because } OQ = RP$$

Now $\triangle ARP$ and $\triangle PQB$ are similar right angle triangles. Therefore part PB is the lower section and part PA is the upper section of the AB demand curve. Thus,

the elasticity of demand at point P is equal to $\frac{PB}{PA}$. That is

$$\text{Point Elasticity of Demand} = \frac{\text{Lower segment of demand curve}}{\text{Upper segment of demand curve}}$$

Curved Demand Curve : Elasticity of demand can also be estimated on a curved demand curve. We shall have to draw a tangent to the demand curve at the point where elasticity of demand is to be measured. Elasticity shall be equal to the ratio of the lower part of the tangent and upper part of the tangent. In the fig. the point elasticity at H point is equal to HN/HR.

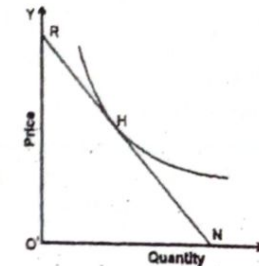


Fig.

4. Revenue Method : Revenue refers to the sale proceeds of a firm. Elasticity of demand can be estimated if average revenue and marginal revenue are known. Average revenue is the price per unit of the commodity. Marginal revenue is the addition to total revenue by the sale of an additional unit of the commodity. The following formula will give the elasticity of demand:

$$ep = \frac{A}{A - M}$$

where

ep stands for elasticity of demand

A stands for average revenue

M stands for marginal revenue

Q.24 What are the determinants of Price Elasticity of demand? Describe the importance of price elasticity of demand.

Ans. Followings are the main determinants of elasticity of demand:

1. Nature of Commodity : Commodities are classified as necessities, luxuries and comforts.

- A necessity that has no close substitute (salt, newspaper, polish, etc.) will have an inelastic demand because its consumptions cannot be postponed. Moreover, consumers purchase almost a fixed amount of a necessity per unit of time whether the price is sameway higher or lower.

THEORY OF PRODUCTION

3

PREVIOUS YEARS QUESTIONS

PART-A

Q.1 Define the term Marginal Cost. [R.T.U. 2019]

Ans. Marginal cost is the extra cost of producing one additional unit. It may at times be impossible to measure marginal cost, e.g. if a firm produces 10,000 metres of cloth, it can become impossible to determine the change in cost involved in producing 10,001 metres of cloth. The difficulty can be solved by taking units of significant size.

Q.2 Define Managerial Economics. [R.T.U. 2018]

Ans. Managerial Economics : With the increase of the size of the firm, the efficiency of management usually increases as there can be greater specialization in managerial staff. In large concerns, experts can be appointed to look after various sections or divisions of the industry such as purchasing, sales, production, financing, personnel etc. But a firm having small means cannot hope to have a full time employment to such experts. Normally the different aspects of the concern have to be looked by few persons only who may not be experts.

Q.3 Define Marketing Economics.

Ans. Marketing Economics : As for its purchase and sale, a large concern can have more economies. It can buy its necessities in large amount and on better terms. Very often it gets quick and safe delivery, careful attention and special facilities from its suppliers. Similarly from the

transport agencies special concessions are secured for bulk movement of goods and raw materials. Expert buyers can be appointed and so also with appointment of expert salesman, the business is bound to progress. A large scale concern can also afford to spread its advertisement costs over bigger output. These costs do not go up in proportion to a rise in sales.

Q.4 Write short note on private cost and social cost.

Ans. Private Cost and Social Cost : While private cost refers to the cost of production to an individual producer, social cost refers to the cost of producing commodity to society in form of resources that are used to produce it. From the social point of view, the economy has a certain volume of resources in the form of capital, land etc. which it would like to put to the best uses. This depends upon the efficient and full utilization of resources and also the specific list of commodities to be produced. If available irrigated area can be put under rice or sugarcane the social cost of producing rice is the amount of sugarcane which irrigated land could have produced instead. It would be ideal if the social cost coincided with the private costs of producing a commodity.

PART-B

Q.5 What is Isoquant? Explain with the help of diagram. [R.T.U. 2019]

Ans. Isoquants : Isoquants are similar to indifference curves of the theory of consumer behavior. An isoquant

Managerial Economics and Financial Accounting

represents all those combinations of inputs which are capable of producing the same level of output. Isoquants are also called equal-product or iso-product curves. Since an equal-product curve represents all those combination of inputs which yield an equal quantity of output, the producer is indifferent between them. Therefore, another name for an isoquant is production-indifference curve. The concept of isoquant can be easily understood with the help of the following schedule.

Table : Various combinations of X and Y to produce a given level of output

Factor combination	Factor X	Factor Y
A	1	12
B	2	08
C	3	05
D	4	03
E	5	02

When we plot them, we get a curve IQ as shown in Figure 1.

Isoquants have properties similar to indifference curves, but there is one important difference between the two : where as in an indifference curve it is not possible to quantify the level of satisfaction acquired by the consumer, the level of production acquired by the producer is easily quantified. Thus, while isoquant IQ1 represents 100 units, curves IQ2, IQ3 etc. representing higher levels of production can be drawn. While a curve on the right represents a higher level of output, that on the left represents a lower level of output.

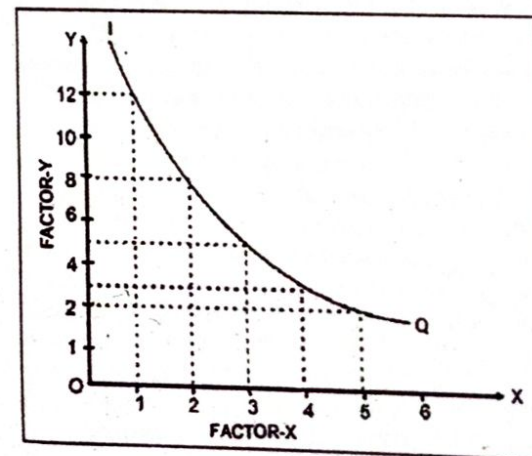


Fig. 1 : Equal Product Curve or Isoquant

(Langhans et al., 2010) have been conducted. It models all economic relevant parameters along with the aircraft life cycle. The aircraft operational lifecycle is initiated by the acquisition of an aircraft and ends with the decommissioning. The model includes aircraft specific parameters (e.g. acquisition cost, fuel consumption, seating capacity, crew size and aircraft specific charges), operational aspects (e.g. route network, maintenance concepts and costs, and ticket prices), as well as global boundary conditions (e.g. fuel price trend, annual inflation rate). AIRTOBS focuses on the perspective of an aircraft operator and includes methods to account for costs and revenues. An overview of AIRTOBS is shown in Figure. It consists of three main modules. The Flight Schedule Builder (FSB) generates a generic aircraft lifecycle flight schedule based on airline route data assuming full aircraft availability (i.e. no maintenance). Routes are considered based on the aircraft cycle time including flight time, taxi and runway operation times and turnaround time. This provisional flight schedule serves as the fundamental for the Maintenance Schedule Builder (MSB). The MSB executes a simulation run of the flight operation and maintenance events over the aircraft lifecycle. The MSB uses input data from maintenance databases for the modeling of scheduled and unscheduled maintenance events, including airframe, engine and component maintenance.

Q.7 Give comparison between the Law of Variable Proportions and Return of Scale.

Ans. Distinction :

S. No.	Bases	Law of Variable Proportions	Law of Returns to Scale
1.	Time Period	Short run phenomenon	Long run phenomenon
2.	Variability of Factor	Only one factor variable others are kept fixed.	All factors are assumed to be variables
3.	Factor Proportion	Factor proportion is varied.	Factor proportion is constant, scale of operations is altered.
4.	Reasons for operation of law:		

(a) Phase I :	Stage of increasing returns due to under-utilisation of fixed factors, specialization of factors.	Increasing returns to scale due to economies of scale.
(b) Phase II :	Stage of diminishing returns : due to (i) Non-optimal factor proportional (ii) Imperfect elasticity of substitution of factors.	Stage of decreasing returns of scale : due to diseconomies of scale.
(c) Phase III :	Stage of Negative Returns, when variable factor is used more and more in relation to fixed factor.	Stage of Constant Returns to Scale, when economies stop to operate yet the diseconomies have not set in.

Q.8 Write a note on Production Function.

Ans. A production function explains the relationship between factor input and output in physical terms. The production function illustrates the technological relationship between input and output. In other words, it shows that with a given state of technology and during a particular period of time how much can be produced with give amount of inputs. It also explains how output changes with the change in the quantity of inputs. It can be written as:

$$Q_x = f(a, b, c, \dots, z)$$

Where Q_x is the quantity of a commodity say X and $a, b, c, \dots, z$ are amounts of various inputs such as land, labour, capital, raw materials, etc. There is a positive function relationship implying that the output varies in the same direction as the inputs quantities. In other words, if all other inputs are held constant, the output will increase upto a limit if the quantity of one input is increased. It has

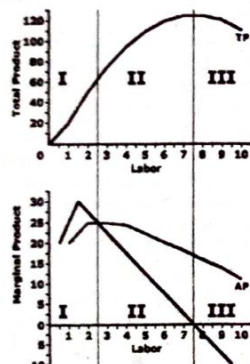


Fig. Three Stages of Production Function

Q.14 Explain the Law of Variable Proportion.

Ams. In the short-run the level of production can be changed by changing the factor proportions. This law examines the production function with one factor variable, keeping the other factors quantities fixed. In other words this law explains the short-run production function. When the quantity of one input is varied, keeping other inputs constant, the proportion between factors changes. When the proportion of variable factors increases, the total output does not always increase in the same proportion, but in varying proportion. This is why the law is named 'Law of Diminishing Returns'. The 'law of variable proportion' or the law of diminishing returns has been defined by a number of economists. In the words of F. Benham, "As the proportion of one factor in a combination of factors is increased, after a point, first the marginal and then the average product of that factor will diminish". This law explains return to a factor.

Thus, the law states that if more and more units of a variable factor are applied to a given quantity of fixed factor, the total output may initially increase at an increasing rate, but beyond a certain level total output, the rate of

increase in total output eventually diminishes in the use of additional units of the variable factor. The volume of goods produced can be looked at from three different angles viz :

- Total Product,
- Marginal Product,
- Average Product.

Total product refers to total volume of goods produced during a specified period of time. Total product can be raised only by increasing the quantity of variable factors employed in production. For instance, more shirts will be produced when more labour and capital are used. Total product, generally goes on increasing with an increase in the quantity of the factor services employed. But there is a limit to which total product can increase with increase in the quantity of variable factors of production.

Marginal Product (MP) : The rate at which total product increases is known as marginal product. We also define marginal product as the addition to the total product resulting from a unit increase in the quantity of the variable factor. Initially marginal product rises, but ultimately it begins to fall down, it becomes zero and at last becomes negative. It would be seen that the total product is maximum when the marginal product is zero.

Average Product (AP) : Average product can be known by dividing total product by the total number of units of the variable factor.

$$AP = \frac{\text{Total Product}}{\text{Units of variable factor}}$$

It can be easily seen that the average product also shows almost the same tendency as does the marginal product. Initially, both the marginal product and the average product rise but ultimately both of these fall. However, marginal product may be zero. The output does not increase at a constant rate as more of any one input is added to the production process. For example on a small plot of land, we can improve the yield by increasing the fertilizer use to some extent. However, excessive use of fertilizer beyond the optimum quantity may lead to reduction in the output instead of any increase as per the law of Diminishing Returns (for instance, single application of fertilizers may increase the output by 50 percent, a second application by another 30 percent and the third by 20 percent. However, if we apply fertilizer five to six times in a year, the output may drop to zero).

The principle of diminishing marginal productivity

(return) states that as additional units of a variable inputs are added to other inputs that are fixed in supply, the increment to output eventually decline (for a constant technology). This phenomenon has been widely observed and there is enough empirical evidence to support it. For business managers, marginal productivity of an input plays an important part in determining how much of that input will be employed.

A hypothetical production function is presented in the following table with the total, average and marginal products of the variable factor labour. Needless to say that the amount of other inputs, and the state of technology are fixed in this example.

Total Average and Marginal Product

Fixed Input	No. of labour	Total output	AP	MP
X	0	0	-	-
Do	1	3	3	3
Do	2	8	4	5
Do	3	12	4	4
Do	4	15	3-3/4	3
Do	5	17	3-2/5	2
Do	6	17	2-5/6	0
Do	7	16	2-2/7	-1
Do	8	13	1-5/8	-3

The Law of Variable Proportions explains the relation between proportions of fixed variable inputs, on the one hand, and output on the other. When a firm expands output by employing more units of a variable factor, it alters the proportions between the fixed and the variable factors. There is always an optimum combination of factors of production at which cost per unit is minimum. Too less or too much of the variable factors leads to cost increases. The law speaks about three stages of production. The first stage goes from origin to the point where the average output is maximum. When a firm expands output by increasing the quantity of variable factors in proportion to fixed factors it moves towards optimum combination of factors of production. In this stage, the law of increasing return may be said to operate and marginal product begins to fall i.e. law of diminishing returns set in.

The second stage goes from the point where the average output is maximum to the point where the marginal output is zero. After having attained the optimum. Combination of the fixed inputs and the variable input, if the firm increases still further the quantity of the variable input, the per unit output of the variable input falls. In this stage, total output rises but only at a diminishing rate.

The third stage covers the range over which the marginal output is negative and total output naturally falls. No producer will operate at this stage, even if he can procure the variable input at zero price.

The first and the third stages are known as stages of economic absurdity or economic non-sense. A producer will always seek to operate in the second stage. At which point the producer will operate in this stage will depend upon the prices of the factor inputs. In the following figures we have drawn TP and units of variable inputs in one figure and AP and MP and units of variable inputs in the other figure. In both the table and the graphic representation, we see that both average and marginal products first increase, reach the maximum and eventually decline.

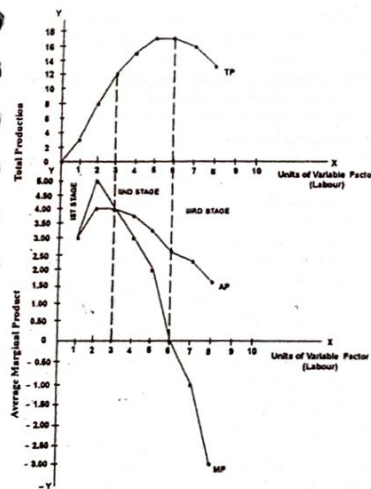


Fig.

Note that $MP = AP$ at the maximum of average product function. This is always the case if $MP > AP$, the average will be pushed up by the incremental unit and if $MP < AP$, the average will be pulled down. It follows that the average product will reach its peak where $MP = AP$.

Assumptions : The application of the law is subject to the following assumptions.

- Technique of production remain unchanged.
- Prices of inputs is given and do not change.
- Units of the variables factors are homogeneous.
- Factor proportion can be altered.

The diminishing return stage of the law is almost universal. This generally, applies to every productive activity.

Causes of The Operation of The Law

The following are the main causes :

1. Fixed factors of production : In the short-run, some factors are fixed. In the beginning, the quantity of variable factor is less in comparison to fixed factor. Therefore, fixed factor remain unutilized. When more and more units of variable factor are employed then fixed factor is used more intensively and effectively. This causes the total production to increase at an increasing rate. Total production increases at an increasing rate till optimum proportion between factors is achieved. But when the units of variable factor are increased after optimum proportion, the total production begin to increase at a diminishing rate because now there is deficiency of fixed factor.

2. Indivisibility of same factor : Some factors are indivisible. Therefore, at least one unit of such factor is to be employed whether production is less or more.

3. Factors of production are not perfect substitutes : Factors of production are substitutes for each other only to a certain extent. Variable factors has to work with lesser and lesser quantity of fixed factor.

4. Internal economics is also a cause of the operation of the law.

Q.15 Give the meaning of ISO-Product curves, their properties and explain the producer's equilibrium (Least Cost Combination).

Ams. Isoquants are a geometric representation of the production function. The same level of output can be produced by various combinations of factor inputs. Assuming continuous variation in the possible combinations of labour and capital we can draw a curve by plotting all these alternative combinations for a given level of output. This curve which is the locus of all possible combinations yielding the same amount of production is called Isoquant or Isoproduct curve. Each isoquant corresponds to a

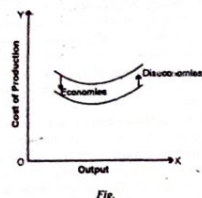


Fig.

In Fig. the LAO curve A moves downwards; it means that the firm derives economies of scale with increase in its scale of production. After a point of optimum capacity, the curve begins to move upwards.

Q.17 Distinguish between private cost and social cost, real cost and opportunity cost, past cost and future cost, explicit cost and implicit cost, short-run and long-run cost.

OR

Explain

- Actual cost and opportunity cost.
- Opportunity cost and out of pocket expenses.
- Opportunity cost and sunk cost.
- Differential and Incremental cost.

Ans.(a) Private Cost and Social Cost : Refer to Q.4.

(b) Actual Cost and Opportunity Cost : Actual Costs mean the actual amount of expenses incurred for producing or acquiring a good or service. These are the costs which are generally recorded in the books of accounts for cost or financial purpose such as payment for wages, raw materials purchased, other expenses paid etc. These costs are also known as 'Outlay costs or Absolute costs'.

The cost of production of any unit of a commodity 'A' is the value of the factors of production used in producing the unit. The value of these factors of production is measured by the best alternative use to which they might have been put had a unit of 'A' not been produced. This concept of cost has been popularized by the American

writers. Opportunity costs are also known as alternative costs.

(c) Past Costs and Future Costs : Actual costs or historical costs are records of past costs. Future costs on the other hand, are based on forecasts. The costs relevant for most managerial decisions are forecasts of future costs or comparative conjunctures concerning future situations. Forecasting of future costs is required for expenditure control, projection of future income statements, appraisal of capital expenditure, decision on new projects and on expansion programmes and pricing. The essence of expense control is to have a standard by which performance can be judged; it is therefore necessary to get scientific standards that will realistically reflect the operating conditions that will govern the level of costs during the future accounting period for which the budget is set. For policy decisions on price too, the business enterprise depends upon future cost and on not past cost. Past cost or historical cost is relevant only under the assumption that, the cost conditions of the previous period would be duplicated in the future too.

(d) Explicit Cost and Implicit Cost : The total cost of production of any particular goods can be said to include 'expenditure or explicit costs and non-expenditure or implicit costs'.

Expenditure or outlay costs are those which are paid by the employer to owners of the factor units which do not belong to the employer itself. These costs are in the nature of contractual payments and they consist of wages and salaries paid, payments for raw materials, interest on borrowed capital funds, rent on hired land and taxes paid to the government. Non-expenditure or implicit costs arise when factor units are owned by the employer himself. The employer is not obligated to anyone also in order to obtain these factors. The two normal non-expenditure costs are depreciation and an average or normal return on the money capital supplied by the shareholder; in the case of small business units the wages of the entrepreneur or organizer himself will have to be included in this category. Expenditure costs are explicit since they are paid to factors outside the firm, while, non-expenditure costs are implicit and hence they are imputed costs. But the latter are costs in the real sense of the term, since the factor units owned by the organizer himself can be supplied to other producers for a contractual sum if they are not used in the business of the organizer.

(e) Incremental Costs (Differential Costs) and Sunk Costs : Incremental cost is the additional cost due to change in the level or nature of business activity. The change may take several forms, e.g., addition of a new product line, changing the channel of distribution, adding a new machine, replacing a machine by a better one, expansion into additional markets, etc. The question of incremental or differential costs would not arise when a business has to be set up a fresh. It arises only when a change is contemplated in the existing business.

Sunk cost is one which is not affected or altered by a change in the level of nature of business activity. It will remain the same whatever the level of activity may be. The most important example of sunk cost is the amortization of past expenses e.g., depreciation. The distinction between the sunk cost and incremental costs assumes importance in evaluating alternatives. Hence incremental costs are relevant to the management in the analysis for decision making. Sunk Costs, on the other hand, will remain the same irrespective of the alternatives selected. Thus, it need not be considered by the management in costs evaluating the alternatives as they are common to all of them.

(f) Short-Run and Long-Run Costs : Short-run costs are costs that vary with output or sales when fixed plant and capital equipment remain the same. Long run costs are those which vary with output when all input factors including plant and equipment vary. Short-run costs become relevant when a firm has to decide whether to produce more or not in the immediate future and when setting up of a new plant is ruled out and the firm has to decide whether to set up a new plant or not. Long-run cost can help the businessman in planning the best scale of plant or the best size of the firm for his purposes. Thus, long-run costs can be helpful both in the initiation of new enterprises as well as the expansion of existing ones.

(g) Fixed and Variable Costs : Fixed costs remain constant in total regardless of changes in volume of production and sales, upto a certain level of output. There is an inverse relationship between volume and fixed costs per unit. If volume of production increases, the fixed costs per unit decreases. Thus, total fixed costs do not change with a change in volume but vary per unit of volume inversely.

Variable costs vary in total in direct proportion increase in the volume. An increase in the volume means

a proportionate increase in the total variable costs and a decrease in volume results in a proportionate between volume and variable costs. They are constant per unit.

Many costs fall between these two extremes. They are called as semi-variable costs. They are neither perfectly variable nor absolutely fixed in relation to changes in volume. They change in the same direction as volume but not in direct proportion thereto. To take an example, electricity bills often include both a fixed charge and a charge based on consumption. This is known as two part tariff.

The distinction between fixed and variable cost is very important in forecasting the effect of short-run changes in volume upon costs and profits. This distinction has given rise to the concepts of Break-Even chart, Direct costing and Flexible Budgets.

(h) Direct and Indirect Costs (Traceable and Common Costs) : A direct or traceable cost is one which can be identified easily and indisputably with a unit of operation (costing unit/cost centre) Common or indirect costs are not traceable to any plant, department or operation, as well as those that are not traceable to indirect final products. The salary of a Divisional Manager when one of the divisions is a costing unit, would be an indirect cost. The distinction is important because in some situations important costs which can be traced to individual units of a product are never-less variable with output and are affected in complex ways by specific output decisions.

Costs of Multiple Products : In some manufacturing enterprises two or more different products emerge from a single raw material. For example a variety of petroleum products are derived from the refining of crude oil.

In a cigarette factory different parts of the tobacco leaves are used for different qualities and products. They are identifiable as separate products only at the conclusion of common processing generally known as the split off point. The costs incurred upto the split off point are common costs-costs which cannot be traced to the separate products in any direct or logical manner. Still for managerial analysis these costs should not be identified with individual products if it is not meaningful and useful to identify them. The identification of such costs, however, becomes meaningful in decisions like the discontinuance or otherwise of a product, addition of new product,

modification or redesigning of existing products and selection of the most appropriate price differential among members of an existing product line.

Some common costs are unaffected by the change that had to be decided upon (e.g. cost of factory building). Fixed common costs need not be allocated since they are irrelevant for any decision, and will remain constant. Common costs that vary with the decision must be allocated to individual products.

For product costing, it is desirable to distinguish between two broad categories of common products i.e. joint product causes an increase in the output of another product, then the products and their costs are traditionally defined as joint. In contrast when an increase in the output of a product is accompanied by a reduction in other products, they may be called as alternative products. Slag and steel are joint products, but steel rails and steel bars are alternative products.

For joint products cost problems relate commonly to the incremental effect of an increase in the output rate to meet new demand for one of the firm products. Such an increase involves higher output for all the products. This may call for reduction in the prices of some joint products to get rid of their increased output. Then the added revenue from one joint product must cover not only the added cost of the whole product package but also any loss of revenue from reduced prices of the other joint products.

(i) Sunk, Shut-Down And Abandonment Costs : A past cost resulting from a decision which can no more be revised is called a sunk cost. It is usually associated with the commitment of funds to specialized equipment or other facilities not readily adaptable to present or future, e.g. brewery plant in times of prohibition.

Shut-down costs may be defined as those costs which would be incurred in the event of suspension of the plant operation and which have been saved if the operations had continued e.g. cost of sheltering the plant and equipment and construction of sheds for storing exposed property. Further, additional expenses may have to be incurred when operations are restarted.

Abandonment costs are the cost of retiring altogether a plant form service. Abandonment arises when there is complete cessation of activities. These costs become important when management is faced with the

alternatives of either continuing the existing plant or suspending its operation or abandoning it altogether.

(j) Out-Of-Pocket and Book Cost : Out of pocket costs refer to costs that involve current cash payments to outsiders. On the other hand, book costs such as depreciation, do not required current cash payments. Book costs can be converted into out-of-pocket costs by selling the assets and having them on hire. Rent would then replace depreciation and interest. While undertaking expansion, book costs do not come into the picture until the assets are purchased.

(k) Replacement And Historical Costs : Historical costs mean the cost of an asset or the price originally paid for it. Replacement cost means the price that would have to be paid currently for acquiring the same plant. The assets are usually shown in the conventional financial accounts at their historical costs. During the period of changing price levels historical costs may not be correct basis for projecting future costs. Historical costs must be adjusted to reflect current or future price levels.

(l) Controllable and Non-controllable Costs : A controllable cost may be defined as one which is reasonably subject to regulation by the executive with whose responsibility that cost is being identified. A cost which is uncontrollable at one level of responsibility may be regarded as controllable at some other higher level. The controllability of certain costs may be shared by two or more executives. The distinction is important for controlling the expenses and efficiency.

(m) Average Cost, Marginal Cost and Total Cost : Average cost is the total cost divided by the total quantity produced. Marginal cost is the extra cost of producing one additional unit. It may at times be impossible to measure marginal cost, e.g. if a firm produces 10,000 metres of cloth, it can become impossible to determine the change in cost involved in producing 10,001 metres of cloth. The difficulty can be solved by taking units of significant size.

In general, economist's marginal cost is cost accountant's differential cost. The total cost of a firm for various levels of output are the sum of total fixed costs and total variable costs. Symbolically

$$\begin{aligned} \text{Total cost} & \text{ or } TC = TFC + TVC \\ \text{Average cost} & \text{ or } AC = TC \div Q \\ \text{Marginal Cost} & \text{ or } MC = TC_n - TC_{n-1} \end{aligned}$$

PRICE-OUTPUT DETERMINATION UNDER DIFFERENT CONDITIONS

4

PREVIOUS YEARS QUESTIONS

PART-A

Q.1 Define the term Perfect Competition.

[R.T.U. 2019]

Ans. Perfect competition : Perfect competition exists when there are large number of buyers and producers (sellers) of a homogeneous product and the price of such a product is determined by the industry. All producers sell the commodity at the price fixed by market forces and no one can change this price.

Q.2 Write short note on price discrimination.

Ans. Price Discrimination : The act of selling the same product by a producer at different prices to the different self-markets is called price discrimination. But it is very difficult to sell the same product at different prices. Therefore, the products are slightly differentiated by the seller to successfully practice price discrimination. Thus, price discrimination means charging different prices for the technically similar products.

Q.3 What is monopolistic competition?

Ans. Monopolistic competition : Monopolistic competition is a situation of market wherein there are large number of buyers and sellers (firms), the sellers are selling differentiated but close substitutes products.

Q.4 What is perfect competition?

Ans. Perfect competition : Refer to Q.1.

According to Mrs. Joan Robinson, "Perfect competition prevails when demand for the output of each producer is perfectly elastic".

Q.5 Explain brief effect of change in demand on price.

Ans. Effect of a Change in Demand on Price : Change in demand conditions cause change in equilibrium price.

(i) **Increase in demand :** When there is increase in demand, the demand curve shifts upward to the right. As a result of increase in demand, supply conditions remaining same, the equilibrium price will rise. The increase in price will be more or less, it depends upon the elasticity of supply curve. If the supply is inelastic then the increase in price will be more for a given increase in demand. On the other hand if supply curve is elastic then increase in price will be less for a given increase in demand.

(ii) **Decrease in demand :** When demand decrease the demand curve shifts downwards to the left. As a result of decrease in demand, supply conditions remaining same, the equilibrium price would fall. The extent to which price falls depends upon the elasticity of supply curve. If the supply curve is elastic then the fall in price will be less and vice-versa.

PART-B

Q.6 "In perfect competition a firm is price taker". Explain the statement with the help of a suitable illustration.

[R.T.U. 2019]

Managerial Economics and Financial Accounting

Ans. "In a perfect competition a firm is price taker". It means every firm is a price taker, setting their price at the market price of "P". Each firm's demand curve is perfectly elastic. This means they can sell as much as they want at the given market price. If one of the firms were to raise its price above P, their sales would fall to zero because the buyers would go to one of the other numerous firms selling the identical good at P. Each firm can sell as much as they want at price P.

As all the firms produce homogeneous products and moreover, consumers have perfect knowledge of producers who can supply goods at lower prices.

Q.7 Define oligopoly. What are its peculiar features?

OR

What is Oligopoly? Explain the features of Oligopoly Market.

[R.T.U. 2018]

Ans. Oligopoly refers to a market situation wherein there are 'a few' (but more than two) sellers selling homogenous or differentiated but close substitute products. Thus, there can be two types of oligopoly – (a) Oligopoly with product differentiation, and (b) oligopoly without product differentiation. Oligopoly is popularly known as 'competition among few'. An oligopoly situation may emerge in any industry when:

- Economies of large scale production exist, as a result, a few firms are able to supply all or much of the total industry output (steel industry, automobiles cement, petroleum industries in India are good example).
- Most efficient entrepreneurs reduce the large number of firms into a few, through mergers and holding companies.
- The exclusive ownership of patents and other right by a few firms or explanation of the advantage of product differentiation creates oligopoly market.
- The controlling power of a few firms over the available national resources reduce competition to only a few.
- The requirement of heavy investment of capital in some industries prevents the entering of a large number of firms in the market.

Generally in an oligopoly market, there are a few well established firms with goodwill of their products in the market. It might be really difficult for a new entrant to popular its new product against the already established brands in the market.

MEFA.59

Features of Oligopoly :

The following prominent features of oligopoly will enable us to distinguish the nature of oligopoly easily from other market situations such as perfect competition, monopoly, etc.

1. **A few sellers :** In oligopoly market the number of sellers (firms) is small. Here small number of firms mean that every firm produces a significant fraction of the total output of the industry and each firm can exercise noticeable impact on the market conditions.

2. **Interdependence :** It is the unique feature of oligopoly. Every firm under oligopoly has to be conscious of the reaction of the rivals. As the number of firms is a few, any change in output, product, price, etc. by one firm will have a direct effect on the fortunes of other who will naturally, retaliate by changing their own prices, output, product, etc. Each firm will have to estimate, as best as it can, the nature of its rival's reaction to its product, price, output, etc. policies. Thus, there is very close interdependence in the decisions of firms in the oligopoly market. Every move in policies by one firm leads to countermoves by other firms.

3. **Indeterminate demand curve :** The oligopoly in characterized by interdependence. Under oligopoly the demand curve of each firm's product is indeterminate because the behavior of rivals is uncertain and unpredictable. The impact of change in price and advertisement cannot be ascertained accurately because of counter moves by rival firms. Thus demand curve cannot be drawn accurately. Since the traditional assumption of "other things being equal" does not apply here.

4. **Aggressive and defensive marketing methods :** Oligopoly firms resort to various aggressive or defensive marketing techniques to increase their share of the market or to maintain their share market. They resort to extensive advertisement and sales promotion. Prof W.J. Bannal has rightly said, "Under oligopoly advertising can become a life and death matter". The reason is that every oligopolistic faces a fierce competition.

5. **Competition and combination :** In oligopoly the competition is not perfect. There may be fierce, violent, cruel and cut throat competition on the one hand. But on the other hand oligopolist realize the disadvantage of competition and rivalry. Therefore, the oligopolist firms may work out some policy of collusion to avoid harmful competition.

MEFA.60

6. Entry of New Firms : The ease of entry may be measured by Bain's concept of the 'condition of entry' which is given by the expression.

$$E = \frac{P_a - P_c}{P_c}$$

Where

E = condition of entry

P_c = Price under pure competition

P_a = Price actually charged by firms

The condition of entry is a measure of the amount by which established firms in an industry can raise price above P_c without attracting entry. In conclusion we can say that the theory of oligopoly is a study of group behavior not of mass or individual behavior, but of behavior of small group of firms. Here profit maximization behavior of firms is not a very valid assumption. There are a few firms in a group which are very much interdependent in their behavior. There is no single accepted theory of such a group behavior. Thus, there can be various models of oligopolistic behavior each based upon (a) degree of interdependence, and (b) case of entry of new firms in the industry. The degree of interdependence of firms may be measured by an unconventional degree of cross elasticity of demand for the products of firm.

As a result of increase in demand the short period price rises to P_1 because demand curve D_1D_1 cuts the supply curve SS at E_2 point. At this increased price OP_1 , present firm starts getting super normal profits. Due to super normal profits new firms enter into the industry and consequently the supply curve shifts to S_1S_1 . Now supply curve S_1S_1 cuts the demand curve D_1D_1 at E_3 point and price falls to OP_2 level. Since it is a decreasing cost industry, the AC and MC curves of firms shifts downward. It is clear that the new long period equilibrium price OP_2 is less than initial increases from Om to Om_1 .

From the above discussion it is clear that long period normal price is affected by the cost change in the long period.

PART-C

Q.11 What do you mean by monopoly? How is price determined under monopoly? Is monopoly price always higher than competitive price?

Ans. Monopoly is a market situation where there is a single producer (seller) of a product which has no close substitute. The concept of monopoly implies the following things:

- 1. Single Seller :** There is a sole producer of a product. This single producer may be individual proprietorship partnership firm or a joint stock company, i.e. the product is under the control of single management. Single firm constitutes the industry.
- 2. Absence of close substitute :** There is no close substitute of the product produced by monopolist because monopoly means absence of competition. In case of monopoly, the cross elasticity of demand for the product is very small.
- 3. Strong barriers to entry :** It means that the new firms are prohibited to enter into monopoly industry. The barriers to entry may be economic, legal, organizational, huge investment, monopoly over raw material, etc.
- 4. Independent price policy :** A monopolist is a price maker. He can increase or decrease price

independently. If he wants to sell more than he can do so by reducing price. Therefore, the demand curve, i.e., average revenue curve of the monopolists has a negative slope.

No one can be pure monopolist because in the end all producers compete for the limited income of consumers. Ultimately, all goods are competitive with each other, however imperfectly competitive they may be.

Average and Marginal Revenue of Monopolist : A monopolist faces a negatively falling demand curve. It means that more quantity can be sold at a lower price, that is why the average revenues (AR) curve of the monopolist falls downward. Similarly the slope of the marginal revenue (MR) curve is also negative. MR curve lies below AR curve and thus MR will be less than AR (price) at each level of output. How much below MR will be from AR, it depends on co-efficient of elasticity of demand (e). We know that AR and MR at a quantity are related to each other through elasticity of demand. by equation—

$$MR = AR \frac{(e-1)}{e}$$

Since the term $\frac{(e-1)}{e}$ will be less than one.

Therefore, MR will be less than AR (price). Thus, the extent to which MR curve lies below AR curve depends

on the value of $\frac{(e-1)}{e}$.

Another point to note is that monopolist demand curve (AR) and industry demand curve is the same as there is no difference between firm and industry in monopoly situation.

Price-output Determination : While determining the price of the commodity, the monopolist is faced with two problems. Firstly, he may fix the price and let the supply (output) be fixed by the conditions of demand. secondly, he may fix the supply and leave the price to be determined by conditions of demand. But he cannot fix both price and output.

The aim of the monopolist is to earn maximum profits. In order to find profit maximizing output, he will compare his marginal cost (MC) with marginal revenue

(MR). He can increase his profits so long as MR is greater than MC. He will incur losses when MC is greater than MR. Thus the monopolist will be in equilibrium when (i) $MC = MR$ and (ii) MC must be rising at the point of equality between $MC = MR$.

Price determination in short-run : In the short period a monopolist cannot change fixed factors of production. Price cut determination under monopoly is explained by fig. (A).

In fig (A), AC and MC are short run average and marginal cost curve respectively. AR and MR are average and marginal revenue curves of the monopolist. MC curve cuts the MR curve at E point. Therefore, the equilibrium output (profits maximizing output) is OQ and OP price is determined. At OQ level of output AC is QR (OS). Per unit super normal profits are RB (QB - QR) and total super-normal profits of the monopolist are equal to the area of the rectangle SRBP. In the short-period a monopolist may earn super-normal profits, normal profits or can even incur losses. If he stops production to avoid losses, he stops production to avoid losses, even in situation of losses he will continue production in the short-run so long as price is equal to or greater than his average variable cost of production to minimize his losses.

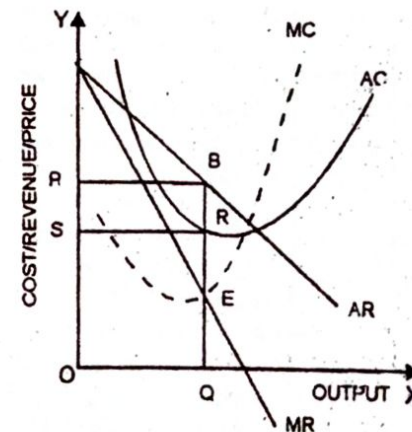


Fig. A

It is clear from the fig. (A) that price under monopoly is greater than MC. But under monopoly price stands in certain relation to MC. We know that :

$$\text{Price (AR)} = MR \frac{e}{(e-1)}$$

In equilibrium, $MR = MC$. Therefore, price

$(AR) = MC \frac{e}{(e-1)}$. The value of the term $\frac{e}{(e-1)}$ will be

more than one for a given value of elasticity of demand. Thus, it follows that under monopoly price is greater than marginal cost (Price > MC). The extent to which monopoly price will be greater than his marginal cost is given by the

value of the term $\frac{e}{(e-1)}$. The smaller the co-efficient of

elasticity of demand, the larger will be the value of the

term $\frac{e}{(e-1)}$ and hence the greater will be the difference

between price (AR) and MC. The monopoly price is, thus the function of MC and degree of elasticity of demand.

Another important point about monopolists equilibrium is worthy of note. A monopolist would always produce that level of output at which elasticity of demand for his product is greater than one. We know that when co-efficient of elasticity of demand is equal to one or less than one, then the MR of monopolist will be zero or negative, since MC can never be negative. Thus, the equality between MC and MR cannot be achieved at that level of output where elasticity of demand is negative, at the most it can be where elasticity of demand is equal to one. Thus a monopolist will be in equilibrium at that price output combination where elasticity of demand is greater than one.

Long-run equilibrium: In the long period a monopolist can make adjustment in the size of this plant to changed demand conditions. In the long-run a monopolist will be in equilibrium at that price output combination where $LMC = MR$ (LMC = long run marginal cost curve). Long run equilibrium is explained by fig. (B), drawn below:

In fig. (B) SAC and SMC are the short run average cost and marginal cost curves of the relevant plant. LAC and LMC are long run average cost and marginal cost curves respectively. The monopolist is in equilibrium at point E where $LMC = MR$ and equilibrium output is OM. At this equilibrium supply the equilibrium price is determined at OR level. Average cost of production on OM output is MT. Thus monopolist earns BTNR super normal profits. In the long period a monopolist earns super normal profits because he can adjust the size of his plant.

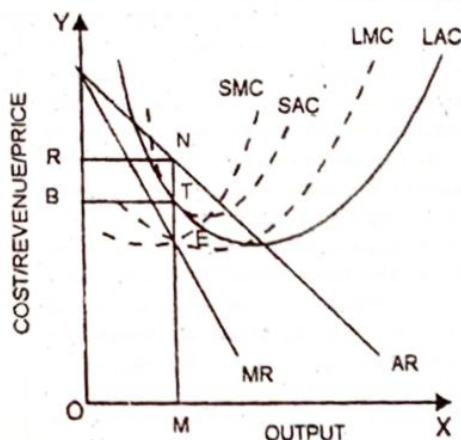


Fig. B

Is monopoly, price always higher than the competitive price. In order to maximize profits a monopolist must fix a higher price. A competitive price is always equal to MC. But monopoly prices are higher than the MC. Thus, we can say that monopoly prices are higher than perfectly competitive prices. But it is not always true. It is possible that, in some cases, under perfect competition cost of production may be higher than cost of production under monopoly. Monopolist is a large scale producer and as such he may have advantages of internal economies of scale of production. He has large resources and can use most modern techniques of production. In all these situations a monopolist may produce at lower cost than competitive firms. Although his price is more than MC yet it may be lower than price under perfect competition.

But in conclusion, we can say that even though the monopolist can produce at low cost, but he fixes a higher price by restricting supply. But not very high price. He is afraid of the government intervention. The government might control the price of this product, if its priced high. Again consumer may boycotts the purchase of his commodity if they consider the behavior of the monopolist as anti-social. Fear of the discovery of some substitutes may be another limitation on the monopolist power of fixing very high price. Thus, in order to maintain goodwill the monopolist will not charge very high price. Thus, it is not always true that monopoly prices are necessarily higher prices.

Q.12 What is price discrimination? Give the degree and conditions of price discrimination. How are price and output determined under discriminatory monopoly?

OR

What are the conditions of price discrimination? Is price discrimination socially desirable?

Ans. Price Discrimination: Refer to Q.2.

Kinds of Price Discrimination: There can be different kinds of price discrimination (i) Sale of doctor's services is an example, (ii) geographical price discrimination, (iii) discrimination according to the use of commodity. Examples are railways, transport services, supply of electricity, etc.

A monopolist who practice price discrimination is known as discriminating monopolist.

Degrees of Price Discrimination: Prof. A.C. Pigion has given three degrees of Price discrimination. Degree of price discrimination refers to the extent or limit to which a monopolist can charge different prices for his product to extract consumer's surplus.

- 1. First Degree Price Discrimination:** It is that price discrimination where a monopolist charges as high price from each buyer as he is willing to pay for each unit of the commodity rather than to go without the commodity. Here the monopolist is in a position to know the price each buyer is willing to pay. In this type of price discrimination the consumer surplus is completely exhausted. In other words monopolist charge so high price from each buyer that his entire consumer is extracted by the seller. This is why it is known as perfect price discrimination. An example of limited price discrimination of this type can be a greedy doctor.
- 2. Second degree price discrimination:** Second degree price discrimination aims at extracting the major part of the consumer's surplus and not the entire consumer's surplus.

price discrimination of second degree occurs when monopolist divides the all buyers into different groups and from each group so much price is charged which the marginal buyer of that group is willing to pay. It is illustrated by fig.

Price monopolist sells OM_1 units at OP_1 price, M_1M_2 units at OP_2 price and M_2M_3 units at P_3 price.

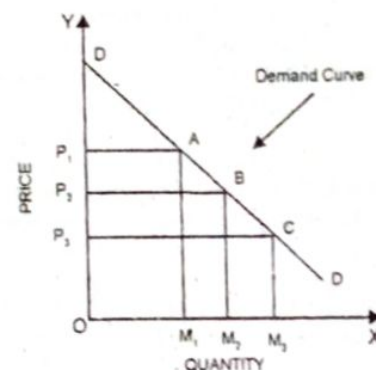


Fig.

- 3. Price Discrimination of third degree:** Price discrimination of third degree means that the monopolist divides his whole market into two or more separate sub-markets on the basis of difference in elasticity of demand and fix different price for each sub-market. Price in different sub-market depends upon quantity supplied in a market and elasticity of demand in that sub-market. This type of price discrimination is common.

First and second degree price discriminations are difficult and not practiced.

Conditions of Price Discrimination

Price discrimination requires specific conditions in the market. These are as follows:

- 1. Market imperfection:** The degree of price discrimination depends upon degree of imperfection in the market. But the likelihood of price discrimination is more when the producer is a monopolist.
- 2. Division of the market into sub-markets:** To exercise price discrimination the monopolist shall divide total market of his product into different sub-markets and keep them separate. It should not be possible that the units of the commodity sold in cheaper market be transferred to dearer market. It means that there should not be any contact between the buyers of different sub-market. Similarly, the cost of keeping the different markets separate should not be more.
- 3. Difference in the elasticity of demand:** Price discrimination is possible and profitable only when elasticity of demand is different in different sub-markets. He can

charge lower price in the market where demand is elastic and higher price in the markets where demand is inelastic. Price discrimination is profitable when elasticity of demand is different in different sub-markets. It can be illustrated with an example.

Suppose elasticity of demand in A and B market is 1 and 4 respectively. Further it is supposed that monopolist fixes price at Rupees 12 in both markets. We know that

$$MR = AR \frac{(e-1)}{e}$$

By using above formula we get :

$$MR_A = 12 \frac{2-1}{2} = 6 \text{ Rupees and}$$

$$MR_B = 12 \frac{5-1}{5} = 9.6 \text{ Rupees}$$

Thus $MR_B > MR_A$ ($9.6 > 6$).

Now the monopolist can increase his profits by reducing supply in A market and increasing supply in B market. As a result of price in market A will increase and market B price will fall. He will maximize profits when marginal revenue in both markets is equal. Thus monopolist can increase his profits if sells at higher price in A market and at low price in B market because elasticity of demand is different in the two market.

4. Legal sanction : In some cases government legally permits price discrimination such as public utilities.

5. Production by order : Producer can charge different prices from different consumers when they get the commodity prepared according to order.

6. Product differentiation : Price discrimination is possible when a product is differentiated by packing, label, design, etc.

Price output determination under discriminating monopoly

The aim of price discrimination is to earn maximum profits. A discriminating monopolist is confronted with three questions:

- (i) What is profit maximizing output? A discriminating monopolist will be in equilibrium at that level of output at which his marginal cost (MC) is equal to his combined marginal revenue (CMR) from different markets of his product. That is when

$$MC = CMR$$

- (ii) How much of the equilibrium output be sold in each sub-market? The discriminating monopolist will distribute his equilibrium output in different sub-markets in such a way that marginal revenue of each market is equal to the marginal cost of equilibrium output. By equation

$$MC = MR_A = MR_B = \dots$$

- (iii) What price be fixed in each market? The price in each market is fixed by supply and demand conditions in each market.

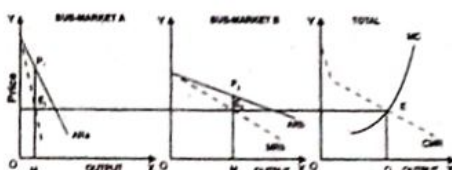


Fig. A

Suppose discriminating monopolist divides his total market into A sub-market and B sub-market. In 'A' market the demand is inelastic and in market B, the demand is elastic. The price output determination is illustrated by the fig. A drawn above.

In the above fig. A, AR_A and AR_B are the average revenue curves of sub-market A and sub-market B respectively. MR_A and MR_B are the marginal revenue curves of A and B sub market respectively. MC is the marginal cost curves of the discriminating monopolist. CMR is the combined marginal revenue curve of the two markets which is obtained by horizontal summation of MR_A and MR_B . The discriminating monopolist is in equilibrium at E point where MC curve cuts the CMR curve and equilibrium output is OQ . The discriminating monopolist will sell OM_1 quantity in market A at M_1P_1 price because at this price the marginal revenue of market A is equal to marginal cost of equilibrium output. He will sell OM_2 quantity in market B because MR_B is equal to MC (QE) of equilibrium output. Thus it is clear that he will sell his product at higher price (M_1P_1) in market A where demand is inelastic and at lower price (M_2P_2) in market B where demand is elastic.

Is Price Discrimination Beneficial to Society

Price discrimination may be useful to the society if the commodity or service cannot be supplied if price discrimination is not permitted. For example the services

of a doctor or of Railways. If the monopolist is to charge a single price, he must fix price at a higher level and this might lower his sales and hence revenue. Moreover, the poor classes may be deprived of those services. On the other hand, if he charges so low a price from all as to suit the poor people, the production cost may not be covered. Thus the monopolist would not like to serve or produce the commodity. As a consequence the society as a whole would be devoid of these services or commodities. But when price discrimination is practiced the monopolist would charge less from poor and more from rich. Though it is a discrimination against the rich, but the poor will be benefited from the sale of such a service or commodity. But when the monopolist charges high prices from the rich, the society has nothing to regret as it would increase the social welfare of the commodity.

But in case of place or geographical discrimination the society may suffer if the monopolist is selling his commodity at lower price in the foreign markets at the cost of the home market where he sells at a higher price. This is certainly harmful to the society. However, if the commodity is being produced under the law of increasing returns (or diminishing costs) the society will gain as larger output will be obtained at a lower cost of production and this is likely to lower the price in the home market.

Dumping : A special case of price discrimination. A special case of price discrimination occurs when a monopoly firm has two markets. (a) a protected home market in which it has monopoly, and (b) a foreign market in which it is to face competition from others. A higher price in domestic market will not restrict his sales at home as higher price would restrict his sales abroad. As result a monopoly firm will find it most profitable to charge a higher price at home and a lower price abroad. This type of price discrimination is known as 'dumping'. The existence of two prices – a higher one at home and a lower one abroad – naturally arouses the anger of competitors abroad, who complain that the monopolist is selling below cost price. Actually this is not true. The discriminating monopoly is actually following the simple objective of maximizing its profits by charging different prices according to the nature of the different market.

Objects of Price Discrimination

1. Through price discrimination, the monopolist aims a maximizing his profits by increasing sales. He tries to squeeze out the maximum from the society.

Thus monopolist charges on 'what the traffic can bear' principle.

2. Another object of price discrimination may be to dispose of his surplus stock in foreign markets at lower prices.
3. He may want to drive out his rivals out of the market by offering goods at very low prices or at dumping prices (i.e., at a price much below the cost) and may charge higher prices in other markets to make up the losses. The main purpose behind this type of behavior is to capture the market.
4. The object of price discrimination may be to develop a new market. The monopolist charges rather low prices in order to establish in the new market and may charge somewhat higher price in already established market.
5. When a monopolist could not utilize the plant to its full capacity, he can increase the production of the commodity by utilizing the unutilized capacity and offer the commodity at lower prices in some new – national or international – markets.

Q.13 Discuss the main features of monopolistic competition. How are price and output determined under monopolistic competition?

OR

Write short note of monopolistic competition.

Ans. In real world neither perfect competition nor monopoly exists. In fact, almost every market seems to exhibit characteristics of both perfect competition and monopoly. Mrs. Joan Robinson called this situation a situation of 'imperfect competition' while Prof. Edward Chamberlin called it monopolistic competition.

Actually imperfect competition is a very loose term and may be referred to as a multitude of market situations such as monopoly, duopoly and oligopoly. Monopolistic competition is the main form of imperfect competition. Thus, imperfect competition is a market situation wherein one or more conditions of perfect competition is absent.

Monopolistic competition is a situation of market wherein there are large number of buyers and sellers (firms), the sellers are selling differentiated but close substitutes products.

PREVIOUS YEARS QUESTIONS

PART-A

Q.1 Define the Capital Budgeting. [R.T.U. 2019]

Ans. Capital Budgeting : Accountants apply capital budgeting techniques as a method for evaluating the potential benefits and risks of large expenses. These include long-term ventures and assets, such as acquiring new land and equipment or building new facilities, with the central goal of yielding greater returns and increasing shareholder value.

Q.2 Explain Break Even Point. [R.T.U. 2018]

Ans. Break-Even Analysis : Break-even Analysis is a specific way of presenting and studying the inter-relationship between the costs, volume and profits. It is an effective and efficient financial reporting system. It provides information to management in most lucid and precise manner.

The Break-even Analysis (or B/E Analysis) establishes a relationship between revenues and costs with respect to volume. It indicates the level of sales at which costs and revenues are in equilibrium. The equilibrium points is commonly known as Break-even point.

The break-even point is that point of sales at which total revenue is equal to total costs. It is not profit, no loss point.

Q.3 Explain Balance Sheet. [R.T.U. 2018]

Ans. Balance Sheet : A Balance Sheet is a statement which is prepared for the purpose of finding out the Assets

and liability position of the concern for the particular period. A Balance Sheet is also described as "Statement showing the sources and application of funds". It is a statement and not an account and prepared from real and personal Accounts. It has two sides. The left side of the balance sheet describes the Liabilities and Capital position. The right hand side of the balance sheet describes all the assets and investments.

Q.4 How comparative statement is prepared?

Ans. These statements refer to the statement of profit and loss and the balance sheet prepared by providing columns for the figures for both the current year as well as for the previous year and for the changes during the year, both in absolute and relative terms. As a result, it is possible to find out not only the balances of accounts as on different dates and summaries of different operational activities of different periods, but also the extent of their increase or decrease between these dates. The figures in the comparative statements can be used for identifying the direction of changes and also the trends in different indicators of performance of an organisation. The following steps may be followed to prepare the comparative statements:

Step 1 : List out absolute figures in rupees relating to two points of time (as shown in columns 2 and 3 of Table).

Step 2 : Find out change in absolute figures by subtracting the first year (Col.2) from the second year (Col.3) and indicate the change as increase (+) or decrease (-) and put it in column 4.

Step 3 : Preferably, also calculate the percentage change as follows and put it in column 5.

$$\frac{\text{Absolute Increase or Decrease (Col. 4)}}{\text{First year absolute figure (Col. 2)}} \times 100$$

Table : Comparative Statement

Particulars	First Year	Second Year	Absolute Increase (+) or Decrease (-)	Percentage Increase (+) or Decrease (-)
1	2	3	4	5
	Rs.	Rs.	Rs.	%

Q.5 List out the tools of financial analysis.

Ans. There are various tools for financial analysis depending on the purpose. The important tools or techniques of financial analysis are as follows :



Q.6 What is the Meaning of Analysis of Financial Statements?

Ans. The process of critical evaluation of the financial information contained in the financial statements in order to understand and make decisions regarding the operations of the firm is called 'Financial Statement Analysis'. It is basically a study of relationship among various financial facts and figures as given in a set of financial statements, and the interpretation thereof to gain an insight into the profitability and operational efficiency of the firm to assess its financial health and future prospects. The term 'financial analysis' includes both 'analysis and interpretation'. The term analysis means simplification of financial data by methodical classification given in the financial statements. Interpretation means explaining the meaning and significance of the data. These two are complimentary to each other. Analysis is useless without interpretation, and interpretation without analysis is difficult or even impossible.

PART-B

Q.7 Compute the Net Present Value of a project if it requires an initial investment of Rs. 2,25,000 and is expected to generate the following net cash inflows.

Year 1	Year 2	Year 3	Year 4
Rs. 95,000	Rs. 80,000	Rs. 60,000	Rs. 55,000

The minimum desired rate of return is 12%.

[R.T.U. 2019]

Ans.

$$\text{Net Present Value} = \sum_{t=0}^n \frac{R_t}{(1+i)^t}$$

R_t = Net cash inflow during a single period (t)

i = Discount rate (or) return that could be earned in alternative investments

t = No. of time periods.

$$\text{So, NPV} = \frac{95000}{(1+0.12)^1} + \frac{80000}{(1+0.12)^2} + \frac{60000}{(1+0.12)^3} + \frac{55000}{(1+0.12)^4} - 225000$$

$$= 84821.42 + 63775.51 + 42706.81 + 34953.49 - 225000$$

$$= 2,26,257.23 - 2,25,000$$

$$= ₹1,275.23$$

Ans.

Q.8 Airline XYZ has reported the following earnings to its shareholders for financial year 2017-2018, as given below :

(a) Net profit before tax : ₹10000000/- (in words Rs. One crore only)

(b) Taxation at 50% of net profit

(c) Equity share capital (₹10/- per share) ₹10000000/- (in words Rs. One crore only)

Calculate the earnings per share for the shareholders. [R.T.U. 2018]

$$\text{Ans. EPS} = \frac{\text{NPAT}}{\text{Total No. of Equity Shares}}$$

NPAT is Net Profit After Tax

Calculation of NPAT

Net Profit Before Tax	10000000
Tax 50%	- 5000000
Net Profit After Tax (NPAT)	5000000

Total No. of Equity Shares = Equity Shares
Capital/Face value per share

$$= \frac{10000000}{10} = 1000000$$

As we know

$$\text{EPS} = \frac{\text{NPAT}}{\text{Total No. of Equity Shares}}$$

$$= \frac{5000000}{1000000}$$

$$\text{EPS} = \text{Rs. 5 Per Share}$$

Q.9 What do you understand by a Balance Sheet and its related concepts?

[R.T.U. 2018]

Ans. Balance Sheet : Refer to Q.3.

Balance Sheet Concepts : Trading, Profit & Loss Account disclose the financial results of the concern at the end of the year. But the Balance Sheet discloses the Assets and Liability Position of the concern as on the date.

Specimen form of Final Account Trading Account for the year ended31.3

Particulars	Amount Rs.	Amount Rs.	Particulars	Cr.	
				Amount Rs.	Amount Rs.
To Opening Stock		xx	By Sales		
To Purchase	xx		Less: Sales Return	xx	
Less: Purchase Return	xx		By Closing Stock		xx
To Wages		xx	By Goods destroyed by fire		xx
To Wages		xx	By Gross Loss		xx
To Carriage inward		xx	[Transferred to P&L A/c debit side]		xx
To Manufacturing Expenses					
To Fuel and Power		xx			
To Motive Power		xx			
To Coal, Water, Gas & Electricity		xx			
To Clearing Charges		xx			
To Import Duty		xx			
To Customs Duty		xx			
To Freight on Purchase		xx			
To Factory Rent and Insurance		xx			
To Works Managers Salary		xx			
To Gross Profit (B/f)		xx			
(Transferred to P&L A/c Credit Side)		xx			
		xx			xx

Profit and Loss Account for the year ended 31.3

Particulars	Amount Rs.	Amount Rs.	Particulars	Cr.	
				Amount Rs.	Amount Rs.
To Gross Loss		xx	By Gross Profit		xx
(Transferred from Trading A/c)			(Transferred from Trading A/c)		
To Advertisement Expenses		xx	By Rent received		xx
To Carriage outwards		xx	By Commission received		xx
To Bank Charges		xx	By Income from Investments		xx
To Salaries		xx	By Interest received		xx
To Rent and Taxes		xx	By Discount received		xx
To Stationeries		xx	By Income from another source		xx

To Insurance	xx	By Discount on Creditors	xx
To Trade Expenses	xx	By Interest on Drawings	xx
To Interest on Capital	xx	By All Business income other than that	
To Interest on Loan	xx	appeared in Trading Account credit side	
To Establishment Expenses	xx		xx
To Selling & Distribution Expn.	xx	By Net Loss	xx
To Sundry Expenses	xx	(Transferred to Balance sheet assets side)	
To Audit Fees	xx		
To Telephone Charges	xx		
To Depreciation	xx		
To Repair and Maintenance	xx		
To Bad Debts	xx		
To Bad Debts provision	xx		
To Loss on sale of fixed Assets	xx		
To Loss on stock by fire	xx		

Balance Sheet as on 31.3

Dr.			Cr.		
Liabilities	Amount Rs.	Amount Rs.	Assets	Amount Rs.	Amount Rs.
Capital		xx	Cash in Hand		xx
Add : Net Profit		xx	Cash at Bank		xx
		xx	Sundry Debtors		xx
(-) Less : Drawings	xx		Investments		xx
+ Interest on Drawings	xx	xx	Marketable Securities		xx
		xx	Bills Receivable		xx
Add : Interest on Capital		xx	Prepaid Expenses		xx
Less : Income Tax		xx	Machinery		xx
Sundry Creditors		xx	Building		xx
Bills payable		xx	Furniture and Fittings		xx
Bank Overdraft		xx	Loose tools		xx
Outstanding expenses		xx	Motor Car		xx
Loan from Banks		xx	House & Carts		xx
Mortgage		xx	Goodwill Patents & Trade Mark		xx
Debenture		xx	Preliminary Expenses		xx
Reserve fund		xx	Profit & Loss A/c (Net Loss)		xx
Income Received in advance		xx	Closing stock		xx

Q.19 Explain capital budgeting technique in detail.
OR

What are various types of capital budgeting techniques?
[R.T.U. 2018]

OR

Explain in brief the various capital budgeting techniques.
[R.T.U. 2018]

Ans. Capital Budgeting : Refer to Q.1.

The following overview of capital budgeting techniques shows the benefits and drawbacks of each method.

Net present value

Determining the net present value is one of the most efficient capital budgeting techniques, both mathematically and in terms of a time value of money perspective. This method analyzes the forecasted cash flow of a project by

discounting the cash flows to the present, factoring in the time span of the project and the weighted average cost of capital.

If the net present value is positive, the business is encouraged to make the investment or undertake the new project; however, the opposite is true if the result turns out negative. Understanding the type of project to be evaluated is the first step to making decisions based on the net present value. Independent projects, which are not affected by other projects' cash flows should be accepted if the net present value is greater than \$0. Mutually exclusive projects represent projects where there are two ways of accomplishing the same ends and, in this case, the project with the greater net present value should be accepted. If the net present value of both projects is negative, then both should be rejected.

Payback Period

The simplicity of the payback period technique makes for a quick budgeting analysis, though there are several deficiencies businesses should be aware of. The payback period is the amount of time it takes for the initial investment of a project to be paid back. Businesses use this method to decide whether or not to invest in a capital project based on the length of time it takes to recoup their initial investment. If the payback period is too long, they will likely not purchase the asset or invest in the project. One of the main deficiencies of this method is that it does not account for the time value of money, which means cash inflows that will be received in the future hold the same weight as those received in the first year. It does provide a quick rough estimate of when an investment will be paid back and can be used as a jumping-off point before employing more advanced capital budgeting techniques.

Accounting Rate of Return (ARR)

The accounting rate of return signifies a percentage value showing the rate of benefits that an asset can generate over its lifespan. Accountants determine the figure by dividing a fixed asset's net income by its average book

value, then multiplying that number by 100. Management determines the ARR for an investment and if the result fits within the established value, then the asset or investment is accepted. Similar to the payback period method, the ARR does not factor in the time value of money and cash flow timing. Also, the ARR does not account for the impact a capital project may have on the business overall as well as increased risks and other variables associated with long-term investments.

Internal Rate of Return (IRR)

This application represents the discount rate resulting in a net present value of zero and is used to analyze actual or potential investments that have varied over time. The IRR is commonly used for project analysis to figure out whether or not managers or investors should embark on a specific project. It is often used in presentations for those without financial backgrounds because it is easier to understand. The limitation of this type of forecasting is that it is more conceptual and does not address scale, which is important for determining real dollar amounts.

Profitability Index

Another technique of capital budgeting involves comparing the value of costs and the value of the proposed project's benefits. This is known as the profitability index, which is calculated by dividing the initial investment by the current value of a project's future cash flows. If the profitability index is greater than 1.0, the profitability is positive and the project is likely a good investment. If it is less than 1.0, the proposed project will lose value. A profitability index equaling 1.0 indicates that the projects cash gains or losses will be minimal.

In order to advise business owners properly regarding undertaking capital projects, accountants must make use of a variety of budgeting methods. Knowing when to use these techniques is an integral part of their job. These and other capital budgeting techniques are necessary functions that accountants can use to identify and recommend profitable investments for organizations.

Q.11 Prepare Trading Account from the following Balances:

	Rs.
Opening stock	80,000
Purchases	3,00,000
Sales	4,50,000
Purchase Returns	10,000
Sales Returns	20,000
Wages	5,000
Carriage & Freight	15,000
Freight on purchases	12,000

firm's present and future profitability. They are also interested in the firm's capital structure to ascertain its influences on firm's earning and risk. They also evaluate the efficiency of the management and determine whether a change is needed or not. However, in some large companies, the shareholders' interest is limited to decide whether to buy, sell or hold the shares.

(f) **Labour Unions** : Labour unions analyse the financial statements to assess whether it can presently afford a wage increase and whether it can absorb a wage increase through increased productivity or by raising the prices.

(g) **Others** : The economists, researchers, etc., analyse the financial statements to study the present business and economic conditions. The government agencies need it for price regulations, taxation and other similar purposes.

Q.16 Write Objectives of Analysis of Financial Statements.

Ans. Analysis of financial statements reveals important facts concerning managerial performance and the efficiency of the firm. Broadly speaking, the objectives of the analysis are to apprehend the information contained in financial statements with a view to know the weaknesses and strengths of the firm and to make a forecast about the future prospects of the firm thereby, enabling the analysts to take decisions regarding the operation of, and further investment in the firm. To be more specific, the analysis is undertaken to serve the following purposes (objectives):

- to assess the current profitability and operational efficiency of the firm as a whole as well as its different departments so as to judge the financial health of the firm.
- to ascertain the relative importance of different components of the financial position of the firm.
- to identify the reasons for change in the profitability/financial position of the firm.
- to judge the ability of the firm to repay its debt and assessing the short-term as well as the long-term liquidity position of the firm.

Through the analysis of financial statements of various firms, an economist can judge the extent of concentration of economic power and pitfalls in the financial policies pursued. The analysis also provides the basis for many governmental actions relating to licensing, controls, fixing of prices, ceiling on profits, dividend freeze, tax subsidy and other concessions to the corporate sector.

Q.17 Explain Tools of Analysis of Financial Statements.

Ans. The most commonly used techniques of financial analysis are as follows:

1. **Comparative Statements**: These are the statements showing the profitability and financial position of a firm for different periods of time in a comparative form to give an idea about the position of two or more periods. It usually applies to the two important financial statements, namely, balance sheet and statement of profit and loss prepared in a comparative form. The financial data will be comparative only when same accounting principles are used in preparing these statements. If this is not the case, the deviation in the use of accounting principles should be mentioned as a footnote. Comparative figures indicate the trend and direction of financial position and operating results. This analysis is also known as 'horizontal analysis'.

2. **Common Size Statements**: These are the statements which indicate the relationship of different items of a financial statement with a common item by expressing each item as a percentage of that common item. The percentage thus calculated can be easily compared with the results of corresponding percentages of the previous year or of some other firms, as the numbers are brought to common base. Such statements also allow an analyst to compare the operating and financing characteristics of two companies of different sizes in the same industry. Thus, common size statements are useful, both, in intra-firm comparisons over different years and also in making inter-firm comparisons for the same year or for several years. This analysis is also known as 'Vertical analysis'.

3. **Trend Analysis**: It is a technique of studying the operational results and financial position over a series of years. Using the previous years' data of a business enterprise, trend analysis can be done to observe the percentage changes over time in the selected data. The trend percentage is the percentage relationship, in which each item of different years bear to the same item in the base year. Trend analysis is important because, with its long run view, it may point to basic changes in the nature of the business. By looking at a trend in a particular ratio, one may find whether the ratio is falling, rising or remaining relatively constant. From this observation, a problem is detected or the sign of good or poor management is detected.

4. **Ratio Analysis**: It describes the significant relationship which exists between various items of a balance sheet and a statement of profit and loss of a firm. As a technique of financial analysis, accounting ratios measure the comparative significance of the individual items of the income and position statements. It is possible to assess the profitability, solvency and efficiency of an enterprise through the technique of ratio analysis.

5. **Cash Flow Analysis**: It refers to the analysis of actual movement of cash into and out of an organisation. The flow of cash into the business is called as cash inflow or positive cash flow and the flow of cash out of the firm is called as cash outflow or a negative cash flow. The difference between the inflow and outflow of cash is the net cash flow. Cash flow statement is prepared to project the manner in which the cash has been received and has been utilised during an accounting year as it shows the sources of cash receipts and also the purposes for which payments are made. Thus, it summarises the causes for the changes in cash position of a business enterprise between dates of two balance sheets.

Fund Flow Statement

Fund Flow Statement is basically a tool of financial analysis. This statement explains the changes in Working Capital and its effect on financial position. It indicates by what means new financing was obtained and for what purposes it was utilised.

Break-Even Analysis : Refer to Q.2.

Q.18 Write objectives and advantages of comparative statement.

Ans. Purpose or Objectives of Comparative Financial Statements – The main purpose of preparing comparative statements is to get insight into the reasons for changes in financial position and operating results of a business enterprise. In brief, the purpose or objectives of comparative statements may be listed as under:

- (1) The comparative statements are prepared to know the strengths and weakness as in various areas of a business.
- (2) The comparative statements help us to know the changes in financial position.
- (3) The comparative statements help us to develop and know the trend.
- (4) Comparative statements are prepared for forecasting the future trends of the business.

Q.19 From the following Statement of Profit and Loss 2014, prepare a Comparative Statement of Profit

Particulars	
Revenue from Operations	
Employees Benefit Expenses	
Other Expenses	
Tax	

Change in Profit $500 - 100 = 400$

Change in Sales $1200 - 500 = 700$

$$\text{PV Ratio} = \frac{\text{Change in Profit}}{\text{Change in Sales}}$$

$$= \frac{400}{700} = \frac{4}{7}$$

$$\text{BEP} = \frac{\text{Fixed Cost}}{\text{PV Ratio}}$$

$$\text{MOS} = \frac{\text{Profit}}{\text{PV Ratio}}$$

	Year 1	Year 2
BEP	$100/(4/7)$	$100/(4/7)$
= Fixed Cost - PV Ratio	= 175.44	= 175.44
MOS	$100/(4/7)$	$500/(4/7)$
= Profit/PV Ratio	= 175.44	= 877.19

Q.23 Explain final accounts in detail?

PDF

Ans. Final Accounts are otherwise called Annual accounts. Normally Final Accounts are prepared at the end of the financial year or Accounting year. The purpose of preparing Final Accounts is to enable one to know the progress of the business, profit or loss and financial position of the firm at the right time.

The preparation of Final Account is not the first step of the Accounting process but it is the final product of the accounting process. It will give valuable information to the management and outsiders at the end of the Accounting period. Only after the preparation of Trial Balance it is possible to prepare Final Account. Final Account consists of the following parts.

1. Trading Account,
2. Profit and Loss Account,
3. Balance Sheet.

1. Trading Account – Trading Account is prepared to know the trading results or gross profit on trading of the business. Simply, it is to find out the gross profit from business due to buying and selling of goods or services during a particular period. In other words, Gross profit is the difference between the sales and the cost of goods sold.

In business, the expenditure and incomes are classified into Direct and Indirect. All the direct expenditure are debited into the debit side of the Trading Account and all the indirect incomes are credited into the credit side of the Trading Account. After transferring all the above details, we have to balance the Trading Account. If the balance is in the debit side it means the business is getting profit and named as Gross profit. Suppose the balance is in the credit side, it shows Loss and it is called Gross Loss. The Gross Profit or Gross Loss is transferred to the profit and Loss Account either on the Credit side or Debit side.

2. Profit and Loss Account – Profit and Loss Account is prepared to find out the net profit or net loss of the business. All the indirect incomes and expenditure are transferred to the Profit and Loss Account on the credit side and debit side respectively. After transferring all details to the Profit and Loss Account, we balance this account. If the balance is in debit side it is called Net profit. Suppose, the balance is in the credit side it is called as Net Loss. Net profit is transferred to the balance sheet Liabilities side and added to the capital. If it is Net Loss it will be deducted from the capital.

3. Balance Sheet – Refer to Q.3.

Concepts of Balance Sheet : Refer to Q.9.

ADVANCED ENGINEERING MATHEMATICS

III C.S.

RANDOM VARIABLES

1

CHAPTER IN A NUTSHELL

Rayleigh Distribution

Rayleigh distribution is used for continuous random variable. It is produced from two Gaussian random variables. Let X and Y be independent Gaussian random variables having mean value μ and variance σ^2 .

$$m_x = m_y = 0 \quad \text{and} \quad \sigma_x^2 = \sigma_y^2 = \sigma^2$$

From figure it may be observed that independent Gaussian random variable X and Y are related to

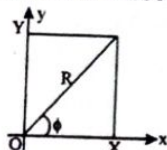


Fig.

Rayleigh distribution random variable R and ϕ such

$$R = \sqrt{x^2 + y^2}; \phi = \tan^{-1}\left(\frac{y}{x}\right)$$

From these equation it is evident that R will always be positive because the variables X and Y are squared.

Also the range of ϕ will be from 0 to 2π because it repeats after 2π radians. This means that ϕ completes one circle when it goes from 0° to 2π .

After transformation of Gaussian random variable X and Y into R , it is found that R is having Rayleigh Probability Density Function (PDF) which is expressed as :

$$f_R(r) = \frac{r}{\sigma^2} e^{-r^2/2\sigma^2}$$

Figure shows the Rayleigh PDF defined by above equation.

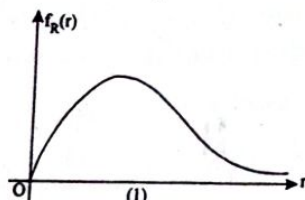


Fig.

From PDF curve it is clear that

$$f_R(r) = \begin{cases} 0 & \text{for } r < 0 \\ \frac{r}{\sigma^2} e^{-r^2/2\sigma^2} & \text{for } r \geq 0 \end{cases}$$

This means that r has always positive value and never goes for negative value

Gaussian Distribution

Gaussian distribution is defined for a continuous random variable. The PDF of a Gaussian random variable is expressed as Gaussian PDF.

$$f_x(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-m)^2/2\sigma^2}$$

m = Mean value of the random variable

σ^2 = Variance of the random variable

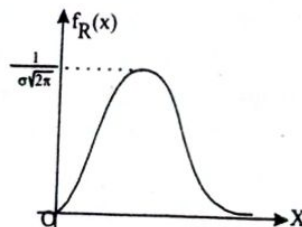


Fig.

Uniform Distribution

The Probability Density Function (PDF) of an uniform distribution is expressed as :

Uniform PDF:

$$f_x(x) = 0 \quad \text{for } x < m - \frac{A}{2} \text{ and } x > m + \frac{A}{2}$$

$$= \frac{1}{A} \quad \text{for } \left(m - \frac{A}{2}\right) \leq x \leq \left(m + \frac{A}{2}\right)$$

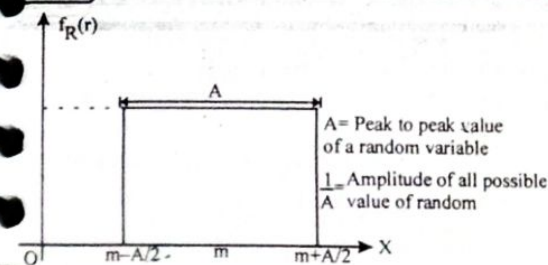


Fig.

In above figure the peak to peak value is A and amplitude uniform i.e. A from figure above. It is also clear that the random variable and its PDF are continuous. Therefore uniform distribution is utilized for continuous random variable.

Also the value of PDF, $f_x(x)$ is same for all possible value of a random variable. Thus this distribution is known as uniform distribution.

Binomial Distribution

Bernoullian Trials : Many problems in the theory of probability involve independent repeated trials of an experiment whose outcomes may be classified into two categories called successes and failures. Such an experiment which has only two possible outcomes, is called a Bernoullian trial.

Binomial Distribution : The binomial probability distribution deals with situation when trial is repeated a number of times and our interest lies in determining the probability of happening or not happening of a particular event. In all the trial the happening of the event is called success and not happening is called failure.

Mathematically

$$B(x, n, p) = P(X = x) P(x) = \begin{cases} \binom{n}{x} p^x q^{n-x} & x = 0, 1, 2, \dots, n \\ 0 & \text{otherwise} \end{cases}$$

Here n = Independent trial

p = Probability of success which is constant for each trial

x = Success, p and n also known as parameters

$P(x)$ = Probability mass function

Moment of Binomial Distribution

$x \sim B(n, p)$ (read x as binomial variate or x is distributed with parameter n and p)

Let $x \sim B(n, p)$ then the r^{th} order moment of x about origin (i.e. $a = 0$) will be given by

$$\mu_r' = E(x^r) = \sum_x x^r P(x)$$

Put $r = 1$

Therefore $f(x) = \begin{cases} \frac{1}{b-a}, & a < x < b \\ 0, & \text{otherwise} \end{cases}$

Distribution function $F(x)$ is given by

$$F(x) = \begin{cases} 0, & -\infty < x < a \\ \frac{x-a}{b-a}, & a \leq x \leq b \\ 1, & b < x < \infty \end{cases}$$

Mean : $\bar{x} = E(x) = \int_a^b x f(x) dx = \int_a^b \frac{x}{b-a} dx$
 $= \frac{1}{b-a} \left[\frac{x^2}{2} \right]_a^b = \frac{1}{2} \frac{(b^2 - a^2)}{b-a}$

$P\{0 < x_1 < \frac{1}{2} < x_2 < 1 < x_3 < 1\}$

$$\begin{aligned} &= \int_0^1 \int_{1/2}^1 \int_0^{1/2} (x_1 + x_2) e^{-x_3} dx_1 dx_2 dx_3 \\ &= \int_0^1 \int_{1/2}^1 e^{-x_3} \left[\frac{(x_1 + x_2)}{2} \right]_0^{1/2} dx_2 dx_3 \\ &= \int_0^1 \int_{1/2}^1 \frac{e^{-x_3}}{2} \left\{ \left(x_2 + \frac{1}{2} \right)^2 - x_2^2 \right\} dx_2 dx_3 \\ &= \int_0^1 \frac{e^{-x_3}}{2} \left\{ \frac{\left(x_2 + \frac{1}{2} \right)^3}{3} - \frac{x_2^3}{3} \right\} \Big|_{1/2}^1 dx_3 \\ &= \frac{1}{6} \int_0^1 e^{-x_3} \left\{ \left(1 + \frac{1}{2} \right)^3 - 1 - \left(\frac{1}{2} \right)^3 \right\} dx_3 \\ &= \frac{1}{6} \int_0^1 e^{-x_3} \left\{ \frac{27}{8} - 2 + \frac{1}{8} \right\} dx_3 = \frac{1}{6} \times \frac{12}{8} \int_0^1 e^{-x_3} dx_3 \\ &= \frac{1}{4} [-e^{-x_3}]_0^1 = \frac{1}{4} [1 - e^{-1}] \end{aligned}$$

PREVIOUS YEARS QUESTIONS

PART-A

Q.1 The probability density function of the random variable x is given by

$$f(x) = \begin{cases} \frac{K}{\sqrt{x}}, & \text{for } 0 < x < y \\ 0, & \text{elsewhere} \end{cases} \text{ find the value of } K.$$

[R.T.U. 2018]

Ans. Probability density function is given by

$$f(x) = \begin{cases} \frac{K}{\sqrt{x}}, & \text{for } 0 < x < y \\ 0, & \text{elsewhere} \end{cases}$$

∴ Above given is a pdf, so

$$\int_0^y f(x) dx = 1$$

$$\int_0^y \frac{K}{\sqrt{x}} dx = 1$$

$$K \int_0^y \frac{1}{x^{\frac{1}{2}}} dx = 1$$

$$K \left[\frac{x^{-\frac{1}{2}+1}}{-\frac{1}{2}+1} \right]_0^y = 1$$

$$2K \left[\frac{1}{\sqrt{y}} \right] = 1$$

$$K = \frac{1}{2\sqrt{y}}$$

Q.2 Find the correlation coefficient between x and y when it is given that.

$$n = 15, \Sigma x = 50, \Sigma y = -30, \Sigma x^2 = 290, \Sigma y^2 = 300, \Sigma xy = -115$$

[R.T.U. 2018]

Ans. The coefficient of correlation is given by

$$r = \frac{\frac{1}{n} \Sigma xy - \bar{x} \bar{y}}{\sqrt{\frac{1}{n} \Sigma x^2 - \bar{x}^2} \sqrt{\frac{1}{n} \Sigma y^2 - \bar{y}^2}}$$

∴ Given that $n = 15, \Sigma x = 50, \Sigma y = -30, \Sigma x^2 = 290, \Sigma y^2 = 300, \Sigma xy = -115$

$$\bar{x} = \frac{\Sigma x}{n} = \frac{50}{15} = \frac{10}{3}$$

$$\bar{y} = \frac{\Sigma y}{n} = \frac{-30}{15} = -2$$

So

$$r = \frac{\frac{1}{15} \times (-115) - \frac{10}{3} \times (-2)}{\sqrt{\frac{1}{15} \times 290 - \frac{100}{9}} \sqrt{\frac{1}{15} \times 300 - 4}}$$

$$= \frac{-43}{\sqrt{\frac{58}{3} - \frac{100}{9}} \sqrt{16}}$$

$$[r = 0.4358]$$

Ans.

Q.3 Explain all the properties of conditional distribution.

[R.T.U. 2016]

Ans. Properties of Conditional Distribution

(i) For the distribution function of RV, X subject to B , the equation is:

$$F_x\left(\frac{-\alpha}{B}\right) = 0 \text{ and } F_x\left(\frac{+\alpha}{B}\right) = 1$$

(ii) For $x = x_1, 0 \leq F_x\left(\frac{x}{B}\right) \leq 1$

(iii) If $x_1 < x_2$ then:

$$F_x\left(\frac{x_1}{B}\right) \leq F_x\left(\frac{x_2}{B}\right)$$

and $F_x\left(\frac{x_2 \leq X \leq x_1}{B}\right) = F_x\left(\frac{x_2}{B}\right) - F_x\left(\frac{x_1}{B}\right)$

(iv) $\lim_{x \rightarrow x^+} F_x\left(\frac{x}{B}\right) = F_x\left(\frac{x}{B}\right)$

(v) Conditional Density Function:

$$F_x\left(\frac{n}{M}\right) = \lim_{\Delta x \rightarrow 0} \frac{P\left[\frac{x \leq X \leq x + \Delta x}{M}\right]}{\Delta x}$$

$$= \frac{d}{dx} F_x\left(\frac{x}{M}\right)$$

(vi) $0 \leq F_x(X) \leq 1$

Q.4 What is a random variable?

Ans. Random Variable: A random variable X is a function that associates a real number with each element in the sample space S .

Let E be a random experiment and S be a sample space associated with it, a function $X(s)$, where $s \in S$, is called a random variable.

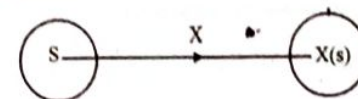


Fig.

Suppose that our experiment E consists of tossing two fair coins, letting X denote the number of heads appearing then X is a random variable taking one of the values 0, 1, 2 with respective probabilities,

$$P\{X=0\} = P\{(T,T)\} = \frac{1}{4}$$

$$P\{X=1\} = P\{(H,T), (T,H)\} = \frac{1}{2}$$

$$P\{X=2\} = P\{(H,H)\} = \frac{1}{4}$$

We have $P\{X=0\} + P\{X=1\} + P\{X=2\} = 1$.

Hence, X is a random variable taking values 0, 1, 2. A random variable is also called chance variate or stochastic variate.

Q.5 Suppose that X is a random variable such that $E(X) = 13$ and $VaT(X) = 25$. Using Chebyshev's inequality, give a bound on $P(5 < X < 21)$.

Ans. This is similar to the last example from class on 11/5. The idea is to write

$$P(5 < X < 21) = P(|X - 13| < 8) = 1 - P(|X - 13| \geq 8)$$

Chebyshev's inequality states that

$$P(|X - \mu| \geq k\sigma) \leq \frac{1}{k^2}$$

In this case $\sigma = 5$ and we want $k\sigma = 8$. So we'll take $k = 1.6$, giving a bound

$$P(|X - 12| \geq 8) \leq \frac{1}{(1.6)^2} \approx 0.391$$

This means that

$$P(5 < X < 21) \geq 1 - .391 \approx 0.609$$

PART-B

Q.6 The joint probability mass function of (x, y) is given by $P(x, y) = k(2x + 3y)$, $x = 0, 1, 2$; $y = 1, 2, 3$. Find :

- (a) K
 (b) Marginal probability distribution of X .
 (c) Marginal probability distribution of Y .
 (d) Conditional probability distribution of X given $y = 1$.

[Note : Read K as k , X as x and Y as y .]

[R.T.U. 2018]

Ans. The joint probability distribution of (x, y) can be represented in tabular form as :

$x \backslash y$	1	2	3
0	$3k$	$6k$	$9k$
1	$5k$	$8k$	$11k$
2	$7k$	$10k$	$13k$

(i) As above given is a pmf, hence

$$\sum_{j=1}^3 \sum_{i=0}^2 P(x_i, y_j) = 1$$

$$\Rightarrow 72k = 1$$

$$\Rightarrow k = \frac{1}{72}$$

$$(ii) P(x=0) = P_{0\bullet} = \sum_{j=1}^3 P_{0j} = P_{01} + P_{02} + P_{03} = 18k$$

$$= \frac{18}{72}$$

$$P(x=1) = P_{1\bullet} = \sum_{j=1}^3 P_{1j}$$

$$= P_{11} + P_{12} + P_{13} = 24k = \frac{24}{72}$$

$$P(x=2) = P_{2\bullet} = \sum_{j=1}^3 P_{2j}$$

$$= P_{21} + P_{22} + P_{23} = 30k = \frac{30}{72}$$

Hence Marginal probability distribution of x is :

$x = i$	P_i
0	$\frac{18}{72}$
1	$\frac{24}{72}$
2	$\frac{30}{72}$
Total	$= 1$

(iii) Similarly marginal probability distribution of y is given by

$y = j$	$P_j = \sum_{i=0}^2 P_{ij}$
1	$P_{01} + P_{11} + P_{21} = 15k = \frac{15}{72}$
2	$P_{02} + P_{12} + P_{22} = 24k = \frac{24}{72}$
3	$P_{03} + P_{13} + P_{23} = 33k = \frac{33}{72}$

(iv) Conditional distribution of x given $y = 1$ is

$$P\left\{\frac{x=i}{y=1}\right\} = \frac{P_{i1}}{P_{\bullet 1}} = \frac{P_{i1}}{15k} \text{ (using (iii)) it can be}$$

represented in tabular form

$x = i$	$\frac{P_{i1}}{P_{\bullet 1}} = \frac{P_{i1}}{15k}$
0	$\frac{P_{01}}{15k} = \frac{3k}{15k} = \frac{1}{5}$
1	$\frac{P_{11}}{15k} = \frac{5k}{15k} = \frac{1}{3}$
2	$\frac{P_{21}}{15k} = \frac{7k}{15k} = \frac{7}{15}$

Q.7 The first four moment of a distribution about the value 5 of the variable are 2, 20, 40, and 50. Find the mean, variance, β_1 and β_2 . Comment upon the nature of distribution. [R.T.U. 2018]

Ans. Here $a = 5$ and moments about this point are

$$\mu_1' = 2, \mu_2' = 20, \mu_3' = 40, \mu_4' = 50$$

We know that

$$\mu_1' = \bar{x} - a \Rightarrow 2$$

$$\bar{x} - 5 = 2$$

AEM.9

$$= \int_{-11}^{-11} f(x) dx + \int_{-11}^c f(x) dx$$

$$= 0 + \int_{-11}^c f(x) dx$$

Given, $c = \frac{1}{2} = 0.5$

$$\therefore P\left(x \leq \frac{1}{2}\right) = \int_{-11}^{0.5} k dx$$

$$= \frac{1}{13} [x]_{-11}^{0.5}$$

$$= \frac{1}{13} [0.5 + 11]$$

$$= \frac{1}{13} [11.5]$$

$$P\left(x \leq \frac{1}{2}\right) = 0.88$$

Ans.

Q.9 Calculate the correlation coefficients for the following ages of husband (x) and wives (y).

x:	23	27	28	28	29	30	31	33	35	36
y:	18	20	22	27	21	29	27	29	28	29

[R.T.U. 2017]

Ans. Coefficient of Correlation :

$$r = \frac{n \sum xy - (\sum x)(\sum y)}{\sqrt{n(\sum x^2) - (\sum x)^2} \times \sqrt{n(\sum y^2) - (\sum y)^2}}$$

x	y	xy	x ²	y ²
23	18	414	529	324
27	20	540	729	400
28	22	616	784	484
28	27	756	784	729
29	21	609	841	441
30	29	870	900	841
31	27	837	961	729
33	29	957	1089	841
35	28	980	1225	784
36	29	1044	1296	841
$\sum x = 300$	$\sum y = 250$	$\sum xy = 7623$	$\sum x^2 = 9138$	$\sum y^2 = 6414$

AEM.10

Now $n = 10$

$$r = \frac{10(7623) - (300)(250)}{\sqrt{10(9138) - (300)^2} \sqrt{10(6414) - (250)^2}}$$

$$r = \frac{1230}{\sqrt{1380} \sqrt{1640}} = \frac{1230}{37.15 \times 40.50}$$

$$r = 0.82$$

Q.10 Consider a continuous random variable X, prove that

$$E[X] = \int_0^{\infty} [1 - F_X(x)] dx - \int_{-\infty}^0 F_X(x) dx$$

Where $F_X(x)$ is the cdf of X. [R.T.U. 2016]

Ans. X is a continuous RV with pdf $F(x)$.

$$\text{so, } F(x) = \int_{-\infty}^x F(x) dx$$

$$\therefore \frac{dF(x)}{dx} = F(x)$$

Now, $E(x) = \int_{-\infty}^{\infty} x F(x) dx$

$$= \int_{-\infty}^0 x F(x) dx + \int_0^{\infty} x F(x) dx$$

$$= \int_{-\infty}^0 x \frac{dF(x)}{dx} - \int_0^{\infty} x \frac{d}{dx} (1 - F(x))$$

$$= [xF(x)]_{-\infty}^0 - \int_{-\infty}^0 F(x) dx - [x(1 - F(x))]_0^{\infty} + \int_0^{\infty} F(x) dx$$

$$= 0 - \int_{-\infty}^0 F(x) dx - 0 + \int_0^{\infty} (1 - F(x)) dx$$

$$= \int_{-\infty}^0 (1 - F(x)) dx - \int_0^{\infty} F(x) dx$$

Q.11 Calculate the rank correlation coefficient for the following data:

X:	81	78	73	73	69	68	62	58
Y:	10	12	18	18	18	22	20	24

[R.T.U. 2016]

B.Tech. (III Sem.) C.S. Solved Paper

Ans. Here $n = 8$, now

X	81	78	73	73	69	68	62	58	Total
Rank $X(x_i)$	1	2	3.5	3.5	5	6	7	8	
Y	10	12	18	18	18	22	20	24	
Rank $Y(y_i)$	8	7	5	5	5	2	3	1	
$d_i = x_i - y_i$	-7	-5	-1.5	-1.5	0	4	4	7	0
d_i^2	49	25	2.25	2.25	0	16	16	49	159.5

Also for observation X:

Rank 3.5 repeated 2 times hence its correction factor

$$= \frac{2(2^2 - 1)}{12} = \frac{1}{2}$$

For observation Y:

Rank 5 repeated 3 times, its correction factor

$$= \frac{3(3^2 - 1)}{12} = 2$$

$$\text{Total correction factor (c.f.)} = 2 + \frac{1}{2} = 2.5$$

$\therefore$ Rank correlation coefficient

$$\rho = 1 - \frac{6[\sum d_i^2 + \text{C.F.}]}{n(n^2 - 1)}$$

$$= 1 - \frac{6[159.5 + 2.5]}{8(8^2 - 1)}$$

$$= 1 - \frac{972}{504} = -0.92857 \text{ (approx.)}$$

Q.12 What is random variable? Explain all the types of random variables with suitable example.

[R.T.U. 2011]

OR

What is a random Variable? Classify them.

[R.T.U. 2014]

Ans. Random Variable : Refer to Q.4.

Discrete Sample Space : A sample space S is said to be Discrete sample space if it contains a finite number of possibilities or an unending sequence with as many elements as there are whole numbers. For example; tossing of a coin and throwing a dice make the discrete sample space which is countable finite. The sequence 1, 2, 3, is countably infinite is also discrete sample space.

Continuous Sample Space : A sample space S is said to be Continuous sample space if it contains an infinite number of possibilities equal to the number of points on a line segment.

For example, the set of points on a plane, the length of time for a chemical reaction to take place and the set of points on a line etc., are all continuous or uncountable sample space.

Types of Random Variables

There are two types of random variables :

- (1) Discrete random variables
- (2) Continuous random variables

(1) Discrete Random Variables

A random variable X is said to be discrete if its set of possible outcomes is countable or a random variable X is said to be discrete if it assumes only a finite number or countably infinite number of values of X . For example; a number of

students who fails in a quiz, $\left(1, \frac{1}{2}, \frac{1}{3}, \dots, \frac{1}{n}, \dots\right)$ marks obtained in an exam, number of telephone calls per unit time, number of accidents per month, number of successes in n trials and number of complaints received at the office of a police station in a day is a countable infinite sequence.

(2) Continuous Random Variables

A random variable X is said to be continuous if it takes the infinite number of values in an interval or when a random variable can take on values on a continuous scale. For example; price of a house, weight of a student, height of a person, distance of two cities, temperature of a room, length of time during which a vacuum tube installed in a circuit function and duration of a call in telephone exchange etc.

Q.13 A continuous random variable can assume any value between $x = 2$ and $x = 5$. It has a density function given by

$$f_x(x) = k(1 + x)$$

Find $P(x < 4)$?

[R.T.U. 2014]

Ans. By the property of pdf

$$\int_{-\infty}^{\infty} f(x) dx = 1$$

$$\Rightarrow \int_2^5 k(1+x) dx = 1$$

$$\Rightarrow \frac{27}{2}k = 1$$

$$\Rightarrow k = 2/27$$

$$\text{Now } P(X < 4) = P(2 < X < 4)$$

$$= \int_2^4 k(1+x) dx = \frac{16}{27}$$

$$\bar{y} = \frac{90}{6} = 15$$

$$\text{Cov}(x, y) = \frac{1}{n} \sum xy - \bar{x} \bar{y}$$

$$= \frac{1}{6} (582 - 5.67 \times 15) = 82.82$$

$$\sigma_x = \sqrt{\frac{1}{n} \sum x^2 - (\bar{x})^2}$$

$$= \sqrt{\frac{1}{6} (248 - (5.67)^2)} = 5.99$$

$$\sigma_y = \sqrt{\frac{1}{n} \sum y^2 - (\bar{y})^2}$$

$$= \sqrt{\frac{1}{6} (1446 - (15)^2)} = 14.26$$

Hence

$$r_{xy} = r = \frac{\text{Cov}(x, y)}{\sigma_x \sigma_y} = \frac{82.82}{5.99 \times 14.26} = 0.97$$

Q.18 Compute the Moment coefficient of skewness (β_1) from the following data.

Marks Obtained	0-10	10-20	20-30	30-40	40-50	50-60	60-70
Frequency	6	12	22	24	16	12	8

Ans. Table for the computations of mean, s.d. and μ_3 .

Class Intervals	Frequency (f)	Mid values (X)	$u = \frac{X-35}{10}$	fu	fu^2	fu^3
0-10	6	5	-3	-18	54	-162
10-20	12	15	-2	-24	48	-96
20-30	22	25	-1	-22	22	-22
30-40	24	35	0	0	0	0
40-50	16	45	1	16	16	16
50-60	12	55	2	24	48	96
60-70	8	65	3	24	72	216
Total	100			0	260	48

Since $\sum fu = 0$, the mean of the distribution is 35.

The second moment μ_2 is equal to the variance (σ^2) and its positive square root is equal to standard deviation (σ).

$$\mu_2 = \frac{260}{100} \times 100 = 260 \text{ and}$$

AEM.13

$$\text{s.d. } \sigma = \sqrt{260} = 16.12$$

$$\text{Also } \mu_3 = \frac{48}{100} \times 1000 = 480$$

$$\text{Thus, } \beta_1 = \frac{(480)^2}{(260)^3} = 0.01$$

Since the sign μ_3 is positive and β_1 is small, the distribution is slightly positively skewed.

PART-C

Q.19 From a lot of 25 items containing 5 defectives, a sample of 4 items is drawn at random,
(i) without replacement (ii) with replacement
Find the expected value of the number of defectives in the sample in each case. [R.T.U. 2018]

Ans. Let X denote the number of defective items. Hence X = 0, 1, 2, 3, 4 as four items are drawn. Defective items = 5 and non defective items = 20.

(i) In this case probability distribution will be

X	P(X=x)
0	$\frac{{}^5C_0 \times {}^{20}C_4}{{}^{25}C_4} = 0.3830$
1	$\frac{{}^5C_1 \times {}^{20}C_3}{{}^{25}C_4} = 0.4506$
2	$\frac{{}^5C_2 \times {}^{20}C_2}{{}^{25}C_4} = 0.1502$
3	$\frac{{}^5C_3 \times {}^{20}C_1}{{}^{25}C_4} = 0.0158$
4	$\frac{{}^5C_4 \times {}^{20}C_0}{{}^{25}C_4} = 0.0004$

$$\text{Hence } E(X) = 0 \times 0.3830 + 1 \times 0.4506 + 2 \times 0.1502 + 3 \times 0.0158 + 4 \times 0.0004 = 0.8$$

(ii) As here the items are replaced, hence the probability of being defective and non defective remains constant.

$$\text{Hence } P(\text{defective}) = \frac{5}{25} = 0.2 = p$$

$$P(\text{non defective}) = \frac{20}{25} = 0.8 = q$$

AEM.14

Hence probability distribution is:

X	P(X=x)
0	${}^4C_0 p^0 q^4 = 0.4096$
1	${}^4C_1 p^1 q^3 = 0.4096$
2	${}^4C_2 p^2 q^2 = 0.1536$
3	${}^4C_3 p^3 q^1 = 0.0256$
4	${}^4C_4 p^4 q^0 = 0.0016$

$$\text{Hence } E(X) = (0 \times 0.4096) + (1 \times 0.4096) + (2 \times 0.1536) + (3 \times 0.0256) + (4 \times 0.0016) = 0.8$$

Q.20 (a) A random variable (continuous) 'x' has a pdf $f(x) = kx^2 e^{-x}$; $x \geq 0$. Find k, mean and variance.

(b) If a continuous random variable 'x' has

Rayleigh density $f(x) = \frac{x}{\alpha^2} e^{-\frac{x^2}{2\alpha^2}} Xv(x)$, find $E(X^n)$ and deduce the values of $E(X)$ and var (X). [R.T.U. 2017]

$$\text{Ans. (a) } \int_0^{\infty} f(x) dx = 1$$

$$\text{or } \int_0^{\infty} kx^2 e^{-x} dx = 1$$

$$\text{or } k \frac{\sqrt{3}}{1} = 1$$

$$\text{or } 2k = 1$$

$$\text{or } k = \frac{1}{2}$$

$$\text{Mean, } \mu = \frac{1+2}{2} = 3$$

$$\text{Now, mean} = 1^{\text{st}} \text{ moment of } x \text{ about origin} = \mu'_1$$

$$\mu'_2 = \frac{4}{2} = 2$$

$$\therefore \text{Variance, } \sigma^2 = \mu'_2 - \mu_1^2 = 2 - 9 = -7$$

$$\boxed{\text{Mean} = 3}$$

$$\boxed{\text{variance} = 3}$$

$$\text{Ans. (b) } E(x^n) = \int_0^{\infty} x^n \cdot \frac{x}{\alpha^2} e^{-\frac{x^2}{2\alpha^2}} dx \quad [\text{by definition}]$$

$$= \int_0^{\infty} (2\alpha^2 t)^{n/2} e^{-t} dt \quad \left[\text{Let } \frac{x^2}{2\alpha^2} = t \right]$$

PART-A

Q.1 Out of 800 families with four children each, how many families would be expected to have at least one boy? Assume equal probability for boys and girls. [R.T.U. 2018]

Ans. Let random variable x denote the number of boys.

$$\text{Here } n = 4, N = 800, p = q = \frac{1}{2}$$

$$P(x = r) = {}^nC_r p^r q^{n-r}$$

$$\begin{aligned} P(x \geq 1) &= P(x = 1) + P(x = 2) + P(x = 3) + P(x = 4) \\ &= {}^4C_1 \left(\frac{1}{2}\right)^1 \left(\frac{1}{2}\right)^3 + {}^4C_2 \left(\frac{1}{2}\right)^2 \left(\frac{1}{2}\right)^2 + {}^4C_3 \left(\frac{1}{2}\right)^3 \left(\frac{1}{2}\right)^1 + {}^4C_4 \left(\frac{1}{2}\right)^4 \\ &= \left(\frac{1}{2}\right)^4 [{}^4C_1 + {}^4C_2 + {}^4C_3 + {}^4C_4] \\ &= \frac{1}{16} [4 + 6 + 4 + 1] \\ &= \frac{15}{16} \end{aligned}$$

$$\begin{aligned} \text{Here number of families having atleast 1 boy} &= 800 \times \frac{15}{16} \\ &= 750 \quad \text{Ans.} \end{aligned}$$

Q.2 Define Binomial distribution. [R.T.U. 2018]

Ans. Binomial Distribution : Let A be some event associated with a random experiment E , such that $P(A) = p$ and $P(\bar{A}) = 1 - p = q$. Assuming that p remains the same for all repetitions, if we consider n independent repetition of E and if the random variable X denotes the number of times, the event A has occurred, then X is called a binomial random variable with parameters n and p . The probability mass function of a binomial R.V. is given by

$$\begin{aligned} P(X = r) &= {}^nC_r p^r q^{n-r}; r = 0, 1, 2, \dots, n. \\ \text{with } p + q &= 1 \end{aligned}$$

Q.3 Prove the reproductive property of independent Poisson's random variable. Hence find the

probability of 5 or more telephone calls arriving in a 9 min. period in a college switch-board. If the telephone calls that are received at the rate of 2 every 3 min. follow a Poisson distribution.

[R.T.U. 2016, 2011]

Ans. The probability mass function of Poisson is given by

$$P[x = k] = \frac{e^{-\lambda} \lambda^k}{k!}, k = 0, 1, 2, \dots$$

Required probability

$$P[x > 5] = 1 - P[x \leq 5]$$

$$\begin{aligned} &= 1 - [P(x = 0) + P(x = 1) + P(x = 2) \\ &\quad + P(x = 3) + P(x = 4) + P(x = 5)] \\ &= 1 - \left[e^{-\lambda} + \lambda e^{-\lambda} + \frac{\lambda^2}{2} e^{-\lambda} + \frac{\lambda^3}{3} e^{-\lambda} + \frac{\lambda^4}{4} e^{-\lambda} + \frac{\lambda^5}{5} e^{-\lambda} \right] \\ &= 1 - \left[e^{-\lambda} \left[1 + \lambda + \frac{\lambda^2}{2} + \frac{\lambda^3}{3} + \frac{\lambda^4}{4} + \frac{\lambda^5}{5} \right] \right] \\ &= 1 - e^{-\lambda} \left[1 + 2 + \frac{4}{2} + \frac{8}{6} + \frac{16}{24} + \frac{32}{120} \right] \\ &= 1 - 0.984 = 0.016 \end{aligned}$$

Q.4 Consider X and Y are two random variables where they are defined as

$$X = \cos \psi$$

$$\text{and } Y = \sin \psi$$

Where ψ is another random variable uniformly distributed over $(0, 2\pi)$. Show that X and Y are uncorrelated. [R.T.U. 2011]

OR

If X and Y are two random variable, given by $X = \cos \Psi$ and $Y = \sin \Psi$ where, Ψ is another random variable, uniformly distributed over $(0, 2\pi)$. Show that, X and Y are uncorrelated. [R.T.U. 2014]

Ans.

$$X = \cos \psi$$

$$Y = \sin \psi$$

Then we know that

$$f_X(x) = \begin{cases} \frac{1}{b-a} & ; a \leq x \leq b \\ 0 & ; \text{otherwise} \end{cases}$$

$$\text{So } F_\psi(\theta) = \begin{cases} \frac{1}{2\pi} & ; 0 < \theta < 2\pi \\ 0 & ; \text{otherwise} \end{cases}$$

Q.8 Two random variables have the least square regression lines with equations:

$3x + 2y - 26 = 0$ and $6x + y - 31 = 0$. Find the mean values and the coefficient of correlation between x and y . ✓ [R.T.U. 2017]

Ans. Since we know that the two lines of regression pass through the point $(\bar{x}, \bar{y})$.

$$\text{We have } 3\bar{x} + 2\bar{y} = 26 \quad \dots(1)$$

$$6\bar{x} + \bar{y} = 31 \quad \dots(2)$$

On solving 1 and 2 we get

$$\bar{x} = 4, \bar{y} = 7$$

$$\text{Also } 3x + 2y = 26$$

$$y = -\frac{3}{2}x + 13$$

$$\therefore b_{yx} = \frac{\mu_{11}}{\sigma_x^2} = \frac{-3}{2} \quad \dots(3)$$

$$\text{Again } 6x + y = 31$$

$$x = -\frac{1}{6}y + \frac{31}{6}$$

$$\therefore b_{xy} = \frac{\mu_{11}}{\sigma_y^2} = \frac{-1}{6} \quad \dots(4)$$

On multiplying 3 and 4, we get

$$r^2 = b_{yx} \cdot b_{xy}$$

$$r^2 = \frac{-3}{2} \times \frac{-1}{6} = \frac{1}{4}$$

$$\therefore r = \pm \frac{1}{2} = \pm 0.5$$

Since both the regression coefficient are negative (i.e. $r \neq 0$)

$$\therefore r = -0.5$$

Q.9 In a normal distribution 31% of the items are under 45 and 8% are over 64. Find the mean and standard deviation of the distributions.

[Given $P = 0.19, 0.23, 0.42$ for $t = 0.50, 0.60, 1.4$] [R.T.U. 2010]

OR

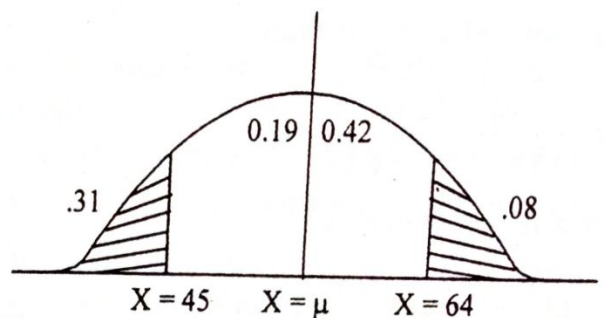
In a normal distribution, 31% of the items are under 45 and 8% are over 64. Find the mean and S.D. of

the distribution. Given that $P = \frac{1}{\sqrt{2\pi}} \int_0^z e^{-\frac{t^2}{2}} dt$, the

values of Z corresponding to $p = 0.19$ and $p = 0.42$ are 0.50 and 1.40 respectively. [R.T.U. 2017]

Ans. Let us consider μ and σ as mean and standard deviation of the distribution.

$$P(-z_1 < z < 0) = P(0 < z < z_1) = 0.19$$



Given

$$P(X < 45) = 0.31$$

$$\text{and } P(X > 64) = 0.08$$

$$\text{As } P(X < 45) = 0.31$$

i.e. $x = 45$ should be on left hand side of mean μ and we must have

$$-Z_1 = \frac{45 - \mu}{\sigma} \quad \dots(i)$$

$$\text{Also } P(X > 64) = 0.08$$

i.e. $X = 64$ should be right hand side of the mean μ and we must have

$$Z_2 = \frac{64 - \mu}{\sigma} \quad \dots(ii)$$

$$\text{Now } P(X < 45) = 0.31 \Rightarrow P(Z < -Z_1) = 0.31$$

$$\Rightarrow P(-Z_1 < Z < 0) = 0.19$$

$$\Rightarrow P(0 < Z < Z_1) = 0.19$$

$$\Rightarrow Z_1 = 0.50$$

$$\text{Also } P(X > 64) = 0.08 \Rightarrow P(Z > Z_2) = 0.08$$

$$\Rightarrow P(0 < Z < Z_2) = 0.42$$

$$\Rightarrow Z_2 = 1.4$$

$$\text{From (i) and (ii) } 0.50 = \frac{45 - \mu}{\sigma} \text{ and } 1.41 = \frac{64 - \mu}{\sigma}$$

dividing above, we get $\mu = 49.9$

$$\text{and } 1.41\sigma = 64 - \mu \Rightarrow \sigma = \frac{14}{1.4} = 10$$

$$\text{Hence } X \sim N(50, 10)$$

Q.10 Prove that poisson distribution is a limiting case of binomial distribution. [R.T.U. 2016]

Ans. Let x be a binomial variate with parameters n and p then

$$P(r) = P(x = r) = {}^nC_r p^r (1-p)^{n-r}, r = 0, 1, 2, \dots, n$$

$$\text{and } p + q = 1$$

PART-C

5 Define Poisson distribution and find its mean and variance. [R.T.U. 2018]

5. Poisson Distribution

There exist situations where the probability P of the opening of an event is very small, but the number of trials n is very large so that the event can happen several times.

Example : Number of persons born blind per year in a certain city.

Example : Number of accidents that take place on a busy road at any particular instant.

The probability of getting 0, 1, 2, 3, r such event is

$$P(r) = e^{-m} \frac{m^r}{r!} \text{ where } r = 0, 1, 2, \dots$$

Mean and Variance of Poisson Distribution

$$\begin{aligned} \text{(i) Mean} &= \sum p_i x_i = \sum_{r=0}^{\infty} r e^{-m} \frac{m^r}{r!} = \sum_{r=1}^{\infty} e^{-m} \frac{m^r}{(r-1)!} \\ &= e^{-m} \left[m + \frac{m^2}{1!} + \frac{m^3}{2!} + \frac{m^4}{3!} + \dots \right] \\ &= m e^{-m} e^m = m \end{aligned}$$

$$\text{Mean} = m$$

$$\begin{aligned} \text{(ii) Variance} &= \sum p_i x_i^2 - (\bar{x})^2 = \sum_{r=0}^{\infty} r^2 e^{-m} \frac{m^r}{r!} - m^2 \\ &= \sum_{r=0}^{\infty} [r(r-1) + r] e^{-m} \frac{m^r}{r!} - m^2 \\ &= e^{-m} \sum_{r=2}^{\infty} \frac{m}{(r-2)!} + m - m^2 \\ &= m^2 e^{-m} \sum_{r=2}^{\infty} \frac{m^{r-2}}{(r-2)!} + m - m^2 = m^2 + m - m^2 = m \end{aligned}$$

$$\text{So, Variance} = m, \text{ Standard Deviation} = \sqrt{m}$$

Q.16 Determine the mean and variance of the random variable X of the following distribution

(i) Uniform distribution

(ii) Exponential distribution

[R.T.U. 2016]

OR

Determine the mean and variance of the uniform distribution $\cup(a, b)$. [R.T.U.]

Ans. Uniform Distribution

Let a variable x assume $(n+1)$ values $0, 1, 2, \dots, n$, with constant probability k then

$$1 = \sum_{i=0}^n p_i = k + k + \dots (n+1) \text{ terms}$$

$$1 = (n+1)k$$

$$\text{So } k = \frac{1}{n+1}$$

Mean of uniform distribution

$$\begin{aligned} \mu'_1 &= \bar{X} = \sum_{r=0}^n x_r P_r \\ &= 0k + 1k + \dots + nk \\ &= k(1 + 2 + \dots + n) \\ &= \frac{1}{n+1} \frac{n(n+1)}{2} = \frac{n}{2} \\ \mu'_2 &= \sum_{r=0}^n x^2 P_r \\ &= k[0^2 + 1^2 + \dots + n^2] \\ &= \frac{1}{n+1} \frac{n(n+1)(2n+1)}{6} = \frac{n(2n+1)}{6} \end{aligned}$$

Variance of uniform distribution

$$\begin{aligned} \sigma^2 &= \mu'_2 - \mu_1'^2 \\ &= \frac{n(2n+1)}{6} - \left(\frac{n}{2}\right)^2 \\ &= \frac{n(n+2)}{12} \end{aligned}$$

If the variable values be $1, 2, \dots, n$ i.e. only n

$$\text{Then } k = \frac{1}{n}$$

$$\bar{x} = \frac{n+1}{2}$$

$$\sigma^2 = \frac{(n+1)(2n+1)}{6} - \frac{(n+1)^2}{4} = \frac{n^2-1}{12}$$

Q.17 Determine the mean and variance of the random variable X of the following distribution

- (i) Normal distribution
- (ii) Rayleigh distribution

[R.T.U. 2016]

Ans.(i) **Normal/Gaussian Distribution:** A continuous random variable (x) is said to follow a normal distribution or Gaussian distribution with parameters μ and σ , if its probability density function is given by

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2} ; -\infty < x < \infty$$

$$-\infty < \mu < \infty$$

$$\sigma > 0$$

Symbolically x follows $N(\mu, \sigma)$

$f(x)$ is a Legitimate density function, as

$$\begin{aligned} \int_{-\infty}^{\infty} f(x) dx &= \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-(x-\mu)^2/2\sigma^2} dx \\ &= \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-t^2} \sigma\sqrt{2} dt \quad (\text{putting } t = \frac{x-\mu}{\sigma\sqrt{2}}) \\ &= \frac{1}{\sqrt{\pi}} \int_{-\infty}^{\infty} e^{-t^2} dt = \frac{1}{\sqrt{\pi}} 2 \int_0^{\infty} e^{-t^2} dt \\ &= \frac{1}{\sqrt{\pi}} \left[\frac{1}{2} \right] = \frac{1}{\sqrt{\pi}} \times \sqrt{\pi} \\ &= 1. \end{aligned}$$

Mean and Variance of Normal Distribution

$$E(X) = \int_{-\infty}^{\infty} xf(x) dx$$

$$= \int_{-\infty}^{\infty} x \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2} dx$$

$$= \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{\infty} x e^{-\frac{(x-\mu)^2}{2\sigma^2}} dx$$

Let

$$t = \frac{x-\mu}{\sigma\sqrt{2}}$$

$$dt = \frac{1}{\sigma\sqrt{2}} dx \quad \text{and} \quad x = \sigma\sqrt{2}t + \mu$$

$$\therefore E(X) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{\infty} (\mu + \sigma\sqrt{2}t) e^{-t^2} dt (\sigma\sqrt{2})$$

$$= \frac{1}{\sqrt{\pi}} \int_{-\infty}^{\infty} e^{-t^2} (\mu + \sigma\sqrt{2}t) dt$$

$$= \frac{\mu}{\sqrt{\pi}} \int_{-\infty}^{\infty} e^{-t^2} dt + \sqrt{\frac{2}{\pi}} \sigma \int_{-\infty}^{\infty} te^{-t^2} dt$$

$\xrightarrow{\text{odd function}}$

$$= \frac{\mu}{\sqrt{\pi}} \times \left[\frac{1}{2} + 0 \right] = \frac{\mu}{\sqrt{\pi}} \times \sqrt{\pi}$$

$$\therefore \boxed{E(X) = \mu} \rightarrow \text{mean of normal distribution}$$

$$E(X^2) = \int_{-\infty}^{\infty} x^2 f(x) dx$$

$$= \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{\infty} x^2 e^{-(x-\mu)^2/2\sigma^2} dx$$

Let

$$t = \frac{x-\mu}{\sigma\sqrt{2}}$$

$$dt = \frac{1}{\sigma\sqrt{2}} dx \quad \text{and} \quad x = \mu + \sqrt{2}\sigma t$$

$$\therefore E(X^2) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{\infty} (\mu + \sqrt{2}\sigma t)^2 e^{-t^2} \cdot \sigma\sqrt{2} dt$$

$$= \frac{1}{\sqrt{\pi}} \int_{-\infty}^{\infty} (\mu + \sqrt{2}\sigma t)^2 e^{-t^2} dt$$

$$= \frac{1}{\sqrt{\pi}} \left[\int_{-\infty}^{\infty} \mu^2 e^{-t^2} dt + 2\sqrt{2}\mu\sigma \int_{-\infty}^{\infty} te^{-t^2} dt + 2\sigma^2 \int_{-\infty}^{\infty} t^2 e^{-t^2} dt \right]$$

$\xrightarrow{\text{odd}} \quad \quad \quad \xrightarrow{\text{even}}$

$$= \frac{1}{\sqrt{\pi}} \left[\mu^2 \int_{-\infty}^{\infty} e^{-t^2} dt + 0 + 2\sigma^2 \int_{-\infty}^{\infty} t^2 e^{-t^2} \cdot 2t dt \right]$$

$$= \frac{1}{\sqrt{\pi}} \left[\mu^2 \left[\frac{1}{2} + \frac{2\sigma^2}{\sqrt{\pi}} \int_0^{\infty} t^3 e^{-t^2} \cdot 2t dt \right] \right]$$

$$\text{Let } u = t^2 \Rightarrow du = 2t dt$$

$$= \frac{\mu^2}{\sqrt{\pi}} \times \sqrt{\pi} + \frac{2\sigma^2}{\sqrt{\pi}} \int_0^{\infty} u^{1/2} \cdot e^{-u} du$$

$$= \mu^2 + \frac{2\sigma^2}{\sqrt{\pi}} \left[\frac{3}{2} \right] = \mu^2 + \frac{2\sigma^2}{\sqrt{\pi}} \times \frac{1}{2} \times \sqrt{\pi} = \mu^2 + \sigma^2$$

$$\therefore \text{Var}(X) = E(X^2) - \{E(X)\}^2$$

$$= \mu^2 + \sigma^2 - \mu^2 = \sigma^2$$

$$\therefore \boxed{\text{Var}(X) = \sigma^2} \rightarrow \text{variance of normal distribution}$$

(ii) **Rayleigh Distribution** : A continuous random variable (X) is said to follow Rayleigh distribution with parameter (σ^2), if its probability density function is given by

$$f(x) = \begin{cases} \frac{x}{\sigma^2} e^{-x^2/2\sigma^2} & ; x \geq 0 \\ 0 & ; x < 0 \end{cases}$$

It is a Legitimate distribution as

$$f(x) = \int_0^{\infty} \frac{x}{\sigma^2} e^{-x^2/2\sigma^2} dx = \frac{1}{\sigma^2} \int_0^{\infty} x e^{-x^2/2\sigma^2} dx$$

$$\text{Let } \left(\frac{x}{\sqrt{2}\sigma} \right)^2 = t$$

$$2x dx = (\sqrt{2}\sigma)^2 dt \Rightarrow x dx = \sigma^2 dt$$

$$f(x) = \frac{1}{\sigma^2} \int_0^{\infty} \sigma^2 dt e^{-t} = \int_0^{\infty} e^{-t} dt = \left[-e^{-t} \right]_0^{\infty}$$

$$\therefore f(x) = -(e^{-\infty} - e^0) = -(0-1) = 1$$

Mean and Variance of Rayleigh Distribution

$$E(X) = \int_0^{\infty} x f(x) dx = \int_0^{\infty} x \cdot \frac{x}{\sigma^2} e^{-x^2/2\sigma^2} dx$$

$$\text{Let } t = \frac{x^2}{2\sigma^2} \Rightarrow x = (2t\sigma^2)^{1/2}$$

$$\therefore dt = \frac{2x dx}{2\sigma^2}$$

$$\text{or } dt = \frac{x dx}{\sigma^2}$$

$$\therefore E(X) = \int_0^{\infty} (2t\sigma^2)^{1/2} \cdot dt \cdot e^{-t}$$

$$= \sqrt{2}\sigma \int_0^{\infty} t^{1/2} \cdot e^{-t} dt = \sqrt{2}\sigma \left[\frac{3}{2} \right] = \sqrt{2}\sigma \times \frac{1}{2} \times \sqrt{\pi}$$

$$\boxed{E(X) = \sqrt{\frac{\pi}{2}} \sigma} \rightarrow \text{mean of Rayleigh Distribution}$$

$$E(X^2) = \int_0^{\infty} x^2 f(x) dx = \int_0^{\infty} x^2 \cdot \frac{x}{\sigma^2} e^{-x^2/2\sigma^2} dx$$

$$\text{Let } \frac{x^2}{2\sigma^2} = t$$

$$\Rightarrow x = (2t\sigma^2)^{1/2} \quad \text{and} \quad dt = \frac{x dx}{\sigma^2}$$

$$\therefore E(X^2) = \int_0^{\infty} t \cdot 2\sigma^2 dt \cdot e^{-t}$$

$$= 2\sigma^2 \int_0^{\infty} t \cdot e^{-t} dt = 2\sigma^2 \left[-te^{-t} + \int e^{-t} dt \right]_0^{\infty}$$

$$= 2\sigma^2 \left[-te^{-t} - e^{-t} \right]_0^{\infty} = 2\sigma^2 \left[-e^{-t}(t+1) \right]_0^{\infty}$$

$$= 2\sigma^2 \left[-e^{-\infty}(\infty+1) + e^0(0+1) \right]$$

$$= 2\sigma^2 [0+1] = 2\sigma^2$$

$$\therefore \text{Var}(X) = E(X^2) - \{E(X)\}^2$$

$$= 2\sigma^2 - \left(\sqrt{\frac{\pi}{2}} \sigma \right)^2 = 2\sigma^2 - \frac{\pi}{2} \sigma^2$$

$$\boxed{\text{Var}(X) = \sigma^2 \left(2 - \frac{\pi}{2} \right)} \rightarrow \text{Variance of Rayleigh distribution.}$$

□□□

HISTORICAL DEVELOPMENT

3

PREVIOUS YEARS QUESTIONS

PART-A

Q.1 Write short note on History of optimization.

[R.T.U. 2018]

Ans. History of Optimization : The word operational research was came into existence during world war-II (1939-1943). At that time the various team of scientists, mathematicians, economist were constituted to deciding as how to best design and operate man-machine system requiring the allocation of scare resources. The solutions of various problems, suggested by the teams, proved to be so effective their adoption ultimately led to winning the war. The methods develop by those research teams anciently adopted by industry and society after and now known as operation research network or optimization techniques.

Q.2 An animal food company must produce 200 kg of a mixture containing ingredients A_1 and A_2 daily. Ingredient A_1 costs ₹ 3.00 per kg and A_2 costs ₹ 8.00 per kg. Not more than 80 kg of A_1 can be used and at least 60 kg of A_2 must be used. Formulate the problem.

[R.T.U. 2018]

Ans. Let x_1 and x_2 be the number of units produce by the company of type A_1 and A_2 respectively.

Objective function

$$\text{Min } z = 3x_1 + 8x_2$$

Subject to conditions

$$x_1 + x_2 = 200$$

$$x_1 \leq 80$$

$$x_2 \geq 60$$

and $x_1, x_2 \geq 0$

Q.3 Define the slack, surplus and artificial variables in linear programming problem.

[R.T.U. 2018]

Ans.(i) Slack Variables : The positive variable which are added to left hand side of the constraints to convert them into equation (equalities) are called slack variables.

Ex. $x_1 + x_2 + x_3 \leq 4$

$$x_1, x_2, x_3 \geq 0$$

Can be converted to

$$x_1 + x_2 + x_3 + x_4 = 4$$

$$x_1, x_2, x_3, x_4 \geq 0$$

then x_4 is called slack variable.

(ii) Surplus Variables : The positive variables which are subtracted from the left hand side of the constraints to convert them into equalities are called surplus variables.

Ex. $x_1 + x_2 \geq 200$

Can be converted to

$$x_1 + x_2 - x_3 = 200$$

then x_3 is called surplus variable.

(iii) Artificial Variables : Artificial variable are added to those constraint with equality (=) and greater than or equal to ($\geq$) sign. An artificial variable is added to the constraint to get an initial solution to an LP problem.

Q.4 A carpenter has 90, 80 and 50 running feet respectively of teak, plywood and rosewood. Product A requires 2, 1 and 1 running feet of teak, plywood and rosewood respectively. Product B requires 1, 2 and 1 running feet of teak, plywood and rosewood respectively. If A would sell for Rs. 48 and B would sell for Rs. 40 per unit, how much of each should he make and sell in order to obtain the maximum gross income out of his stock of wood? Give a mathematical formulation to this linear programming problem.

[R.T.U. 2017]

Ans.

Product	A	B
Teak	2	1
Plywood	1	2
Rosewood	1	1
Selling price per unit	Rs. 48	Rs. 40

Advanced Engineering Mathematics

The supply of plastic is for 1500 dolls only,

$$x + y \leq 1500$$

Also fancy dress required for doll of type B is available for 600 dolls per day only,

$$y \leq 600$$

The profit on doll of type A is Rs. 3 and that on doll of type B is Rs. 5.

$$\therefore \text{The total profit } Z = 3x + 5y$$

Thus, mathematical formulation of LPP is

$$\text{Maximum } Z = 3x + 5y$$

$$\text{subject to } x + 2y \leq 2000,$$

$$y \leq 600,$$

$$x, y \geq 0$$

Q.10 Consider the following problem faced by a production planner of a soft drink plant. He has two bottling machines A and B. A is designed for 8-ounce bottles and B for 16-ounce bottles. However each can also be used for both types of bottles with some loss of efficiency. The following data is available.

Machine	8-ounce Bottles	16-ounce Bottles
A	100/bottles	40/minutes
B	60/minutes	75/minutes

The machine can be run for 8 hours per day, 5 days per week. The profit on a 8-ounce bottle is 15 paise and on a 16-ounce bottle is 25 paise. Weekly production of the drink cannot exceed 3,00,000 bottles and the market can absorb 25,000 eight ounce bottles and 7,000 sixteen ounce bottles per week. The planner wishes to maximize his profit, of course, to all the production and marketing restrictions. Formulate this as linear programming problem.

[R.T.U. 2013]

Ans. Let x_1 units of 8 ounce bottles and x_2 units of 16 ounce bottles be produced. Total profit of production planner is given by $P = 0.15x_1 + 0.25x_2$. Since machine A and B both work 8 hours per day and 5 days per week, the total working time for machine A and B will become 2400 minutes per week. Therefore, time conditions will be

$$\frac{x_1}{100} + \frac{x_2}{40} \leq 2400 \quad (\text{for machine A})$$

$$\frac{60}{75}x_1 + \frac{x_2}{45} \leq 400 \quad (\text{for machine B})$$

$$\text{Restriction of total weekly production will be}$$

$$8x_1 + 16x_2 \leq 300000$$

AEEM-33

Market consumption is restricted by

$$0 \leq x_1 \leq 25000$$

Hence the problem can be put in the form

$$\text{max. } P = 0.15x_1 + 0.25x_2$$

$$\text{subject to } \frac{x_1}{100} + \frac{x_2}{40} \leq 2400$$

$$\frac{60}{75}x_1 + \frac{x_2}{45} \leq 400$$

$$8x_1 + 16x_2 \leq 300000$$

$$0 \leq x_1 \leq 25000$$

$$0 \leq x_2 \leq 7000$$

$$x_1, x_2 \geq 0$$

PART-C

Q.11 Explain Engineering application of optimization techniques.

[R.T.U. Dec. 2016]

OR
Write 12 applications of Optimization techniques in Engineering.

[R.T.U. 2018]

OR
Define optimization techniques and write its various engineering applications.

[R.T.U. 2017]

OR
Explain the term 'optimization'. Discuss briefly the applications of optimization techniques in engineering field.

[R.T.U. 2017]

OR
Define optimization techniques and write its various engineering applications.

[R.T.U. 2016]

OR
Explain applications of optimization techniques in Engineering.

[R.T.U. 2019]

Ans. Optimization Techniques : In the simplest case, optimization means solving problems in which one seeks to minimize or maximize a real function by systematically choosing the values of real or integer variables from an allowed set. This formulation, using a scalar, real-valued objective function, is probably the simplest example. More generally, it means finding "best available" values of some objective functions given a defined domain, including a variety of different types of objective functions and different types of domains.

AEEM-34

Optimization is :

- The act of obtaining the best result under the given circumstances.

- Defined as the process of finding the conditions that gives the minimum or maximum value of a function, where the function represents the effort required or the desired benefit.

- Design, construction and maintenance of engineering systems involve decision making both at the managerial and the technological level.

Goals of such decisions are :

- to minimize the effort required or
- to maximize the desired benefit

Applications of Optimization in Engineering Fields

- In finding the optimal trajectories of space vehicles and missiles.
- For the optimal design of the control systems.
- For the optimal design of the electrical networks, rockets, with regard to their weight and consumption of fuel, their speed, atmospheric effects of temperature, pressure variations etc.
- In the designing of frames and structures with regard to their strength and the quantity of material used in their designing.
- In the designing of new tools and machines with regard to the new dimensions added in their working.
- In the designing of pumps, turbines, heat transfer, equipment etc.
- In the designing of scooters, cars, heavy and light goods carrying vans.
- In designing of foundations of big buildings, bridges, towers, dams, chimneys etc.
- In designing the optimal pipe line network.
- In solving the travelling salesman problem.
- In controlling the idle time and waiting time in queues of production lines to reduce the costs.

Other applications of optimization technique in engineering

Some of the typical applications from different engineering disciplines are given below :

- General
- Optimal production planning, controlling a scheduling.

B.Tech. (III Sem.) C.S. Solved Papers

- Inventory control.
- Optimum design of control systems.
- Shortest route taken of salesperson visiting various cities during one tour.
- Optimum design of decimal machinery such as motors, generator and transformer.
- Optimum design of electrical network.
- Optimum design of chemical processing equipment and plants.
- Design of optimum pipe network for processes industries.
- Optimum design of linkages CAMS, gears, machine tools and other mechanical components.
- Design of civil engineering structure such as frames, fundamental bridges, towers, chimneys and dams for minimum cost.
- Finding the optimal targets of spare vehicles.
- Optimal plastics design of structures.

II. Maximization

- Allocation of resources at services among several activities to maximize the benefit.
- Planning the best strategy to obtain maximum profit in the process of a competitor.
- Design of water resources system for maximum benefit.
- Design of pumps, turbines and heat transfer equipment for maximum efficiency.

III. Minimization

- Planning of maintenance and replacement of equipment to reduce operating cost.
- Controlling the vesting and ideal times and query in production line to reduce the cost.
- Design of aircraft and aerospace structure for minimum weight.
- Minimum weight design of structure for enter quake wind and other types of random loading.
- Selection of machining conditions in metal cutting process for minimum production cost.
- Design of material handling equipment such as conveyors trucks and cranes for minimum cost.

IV. Management

- Deciding the locations site and industry.
- Planning the best strategy in a competitive market and devising ways of increasing the earnings of company.

Advanced Engineering Mathematics

- Finding the shortest route which a salesman should adopt so as to minimize the cost and time spent in visiting a number of cities.
- Inventory control to know when to purchase. How much to purchase to meet the future demands which will arise.
- In the probability and statically fields.

Q.12 Write down a short note on "classification of optimization problem" based on following :

- Classification based on the nature of the design variables
- Classification based on the nature of the equations involved
- Classification based on the permissible values of the design variables
- Classification based on the number of objective functions.

OR

Explain the classification of optimization problem based on

- Nature of junctions involved
- Nature of decision variables involved.

[R.T.U. 2009]

OR

Explain classification and formulation of optimization problem.

[Ref. Univ. 2008, 2006]

Briefly discuss six different basis for classification of optimization problems.

[R.T.U. 2014]

Ans.(a) Classification Based on the Nature of Design Vectors : Based on the nature of design vectors, optimization problems can be classified into two categories :

- Static (parameter) optimization problem
 - Dynamic (trajectory) optimization problem
- (2) Dynamic Optimization Problem : In this category, the problems are to evaluate a set of design parameters which makes the prescribed objective function of these parameters minimum or maximum subjects to the constraints.
2. **Dynamic Optimization Problem** : In this case, the objects is to obtain a set of design parameters which are all continuous functions of some other parameters

CLASSICAL OPTIMIZATION USING DIFFERENTIAL CALCULUS

4

CHAPTER IN A NUTSHELL

OPTIMIZATION

Optimization is the act of solving the problems and obtaining the best result under the given circumstances by minimizing the effort required or by maximizing the desired benefit.

Statement of an Optimization Problem :

To find, X which maximizes $f(X)$ subject to the constraints

$$g_i(X) \leq 0, i = 1, 2, \dots, m$$

$$f_j(X) = 0, j = 1, 2, \dots, p$$

where,

X is an n -dimensional vector called the design vector $f(X)$ is called the objective function, and

$g_i(X)$ and $f_j(X)$ are known as inequality and equality constraints, respectively.

CLASSIFICATION OF OPTIMIZATION PROBLEMS

(A) Classification Based on the Existence of Constraints

- Constrained Optimization Problems
- Unconstrained Optimization Problems

(B) Classification Based on the Nature of the Design Variables

- Static Optimization
- Dynamic Optimization

(C) Classification Based on the Nature of the Equations Involved

- Linear Programming Problem
- Nonlinear Programming Problem
- Geometric Programming Problem
- Quadratic Programming Problem

(D) Classification Based on the Permissible Values of the Decision Variables

- Integer Programming Problem
- Real-valued Programming Problem

(E) Classification Based on Deterministic Nature of the Variables

- Deterministic Programming Problem
- Stochastic Programming Problem

(F) Classification Based on the Number of Objective Functions

- No Objective Function
- Single-Objective Programming Problem
- Multi-Objective Programming Problem

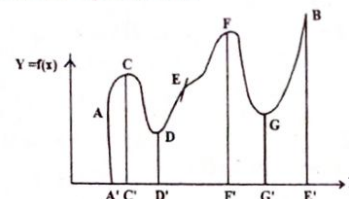
APPLICATION OF OPTIMIZATION

Optimization theory finds ready application in all branches of engineering in four primary areas :

- Design of components or entire systems
- Planning and analysis of existing operations
- Engineering analysis and data reduction
- Control of dynamic systems

SINGLE VARIABLE OPTIMIZATION WITH AND WITHOUT CONSTRAINTS

Single Variable Optimization



Let $y = f(x)$ be a differentiable function having the graph as shown above.

We note the following :

Function has local maximas at C and F.

It has two local minimas at D and G.

E is a point on the graph where the curve changes its concavity with respect to x axis such a point is called a point of inflexion.

Advanced Engineering Mathematics

Function has the highest value at B and lowest at D. The following results are easily obtained analytically.

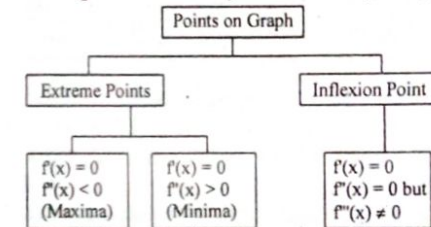


Fig. : Single Variable Optimization

MULTIVARIABLE OPTIMIZATION WITH AND WITHOUT CONSTRAINTS

Case 1 : (With no constraints)

Necessary Condition

If $x = x_0$ then

$$\frac{\partial f(x_0)}{\partial x_1} = \frac{\partial f(x_0)}{\partial x_2} = \dots = \frac{\partial f(x_0)}{\partial x_n}$$

Let $Z = f(x) = f(x_1, \dots, x_n)$ be a function of n variables

$x_1, x_2, \dots, x_n$

Let $P(x)$ and $Q(x + \delta x)$ be two neighbouring points on the n dimensional surface. Expanding by Taylor's expansion.

$$Z + \delta Z = f(x + \delta x)$$

$$= f(x) + \delta x \cdot \nabla f + \frac{1}{2} (\delta x)^T H(x) (\delta x) + \dots$$

If $(\delta x)^T$ is the transpose of vector δx , written as a row

vector, taking δx to be the column vector $\begin{pmatrix} \delta x_1 \\ \vdots \\ \delta x_n \end{pmatrix}$ and

$$H(x) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \dots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \dots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \dots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

is called the Hessian matrix of $f(x)$.

or we can represent as

$$H = \left[\frac{\partial^2 f}{\partial x_i \partial x_j} \right]_{n \times n}$$

Other necessary condition is (not required in one dimensional case) :

$$rt - s^2 > 0$$

If P be a maxima then r must be negative.

Sufficient Conditions for Maxima and Minima :

If r is negative, there is maxima at P .

If r is positive, there is minima at P .

If $rt - s^2 \leq 0$, then the point is called saddle point.

PREVIOUS YEARS QUESTIONS

PART-A

Q1 Determine the maximum and minimum values of the function
 $f(x) = x^3 - 5x^2 + 5x - 1$ [R.T.U. 2018]

Ans. $f(x) = x^3 - 5x^2 + 5x - 1$

$$\frac{df}{dx} = 3x^2 - 10x + 5 = 0$$

$$5x^2(x^2 - 2x + 1) = 0$$

$$5x^2(x - 1)^2 = 0$$

$$5x^2(x - 1)(x - 1) = 0$$

$$x = 0, 1, 1$$

$$\frac{d^2f}{dx^2} = 6x - 10 = 6(0) - 10 = -10 < 0$$

Case I: For $x = 0$

$$\left(\frac{\partial^2 f}{\partial x^2}\right)_{x=0} = -10 < 0 \text{ so we must investigate the next derivative.}$$

$$\frac{\partial^2 f}{\partial x^2} = 60x^2 - 120x + 30$$

At $x = 0$

$$\frac{\partial^2 f}{\partial x^2} = 30, \text{ Hence } x = 0 \text{ is neither a maximum nor a minimum point.}$$

Case II: For $x = 1$

$$\left(\frac{\partial^2 f}{\partial x^2}\right)_{x=1} = -10 < 0, \text{ Hence } x = 1 \text{ is a relative maximum and } f_{\max} = f(1) = 0.$$

Case III at $x = 3$

$$\left(\frac{\partial^2 f}{\partial x^2}\right)_{x=3} = 90 > 0, \text{ Hence } x = 3 \text{ is a relative minimum and } f_{\min} = f(3) = 243 - 405 + 135 - 1 = -28$$

Q2 Show that $f(x) = 4x^3 - 18x^2 + 27x - 7$ is never optimal in a given interval except at its end points. [R.T.U. 2014]

Ans. We have $\frac{df}{dx} = 12x^2 - 36x + 27 = 3(2x - 3)^2$

$$\frac{d^2f}{dx^2} = 24x - 36 = 24(2x - 3) = 24$$

$$\frac{d^2f}{dx^2} = 0 \text{ at } x = \frac{3}{2}$$

But for this value of x , $\frac{d^2f}{dx^2} = 0$ and $\frac{d^3f}{dx^3} \neq 0$

So it is not an extremum point but a point of inflection.

Further $\frac{d^2f}{dx^2} = 3(2x - 3)^2$, is always positive except at the point of inflection $x = \frac{3}{2}$. So f is an increasing function and will be maximum at one of the end points of the interval in which it is defined.

Q3 Minimize $f = 9 - 8x_1 - 6x_2 - 4x_3 - 2x_1^2 + 2x_1x_2 + x_3^2 + 2x_1x_3$
 Subject to $x_1 + x_2 + 2x_3 = 3$
 by constrained variation method. [R.T.U. 2014]

Ans. Here we have three variable and constant i.e. $n = 3$, $m = 1$. We have to choose the dependence and two independent variable so that.

$$\left(\frac{\partial f}{\partial x_1}\right)_{x_2, x_3} = 9 - 8 - 2x_1 = 1$$

$\therefore x_2, x_3$ can be taken as independent variable

$$\left(\frac{\partial f}{\partial x_2}\right)_{x_1, x_3} = 0 \text{ and } \left(\frac{\partial f}{\partial x_3}\right)_{x_1, x_2} = 0 \text{ respectively}$$

$$-6 + 4x_1 + 2x_3 - 8 + 4x_1 + 2x_3 + 2x_1 = 0$$

$$x_1 - x_2 + x_3 = 1 \quad \dots(1)$$

$$-4 + 2x_1 + 2x_3 - 8 + 4x_1 + 2x_3 + 2x_1 = 0$$

$$3x_1 - 2x_2 + x_3 = 6 \quad \dots(2)$$

$$\text{and } g(x) = 0 \Rightarrow x_1 + x_2 + 2x_3 = 3 \quad \dots(3)$$

PART-B

Solving (1), (2) and (3)
 $\Rightarrow x_1 = \frac{4}{3}, x_2 = \frac{7}{9}, x_3 = \frac{4}{9}$

Q4 State Kuhn-Tucker conditions. Use them to minimize $f(x, y, z) = x^2 + y^2 + z^2 + 20x + 10y$
 s.t. $x \geq 40$
 $x + y \geq 80$
 $x + y + z \geq 120$ [R.T.U. 2016]

OR

Use Kuhn-Tucker conditions to Min $f(x, y, z) = x^2 + y^2 + z^2 + 20x + 10y$
 s.t. (a) $x \geq 40$
 (b) $x + y \geq 80$
 (c) $x + y + z \geq 120$ [R.T.U. 2016, 2011, 2010, 2008]

Ans.

$$Z = f(x), x = (x, y, z)$$

$$g_j(x) \geq 0, j = 1, 2, 3 \quad \dots(1)$$

$$L = f(x, y, z) - \sum_{j=1}^3 \lambda_j (g_j(x) - s_j^2) \quad \dots(2)$$

$$L = x^2 + y^2 + z^2 + 20x + 10y + \lambda_1(x - 40 - s_1^2) + \lambda_2(x + y - 80 - s_2^2) + \lambda_3(x + y + z - 120 - s_3^2) \quad \dots(3)$$

$$\frac{\partial L}{\partial x} = 0, \frac{\partial L}{\partial y} = 0, \frac{\partial L}{\partial z} = 0 \quad \dots(4)$$

$$\lambda_1 s_1 = 0 \quad \dots(5)$$

$$\lambda_2 s_2 = 0 \quad \dots(6)$$

$$\lambda_3 s_3 = 0 \quad \dots(7)$$

$$\lambda_1, \lambda_2, \lambda_3 \leq 0 \quad \dots(8)$$

$$\lambda_1, \lambda_2, \lambda_3 \geq 0 \quad \dots(9)$$

$$\lambda_1, \lambda_2, \lambda_3 \leq 0 \quad \dots(10)$$

$$\lambda_1, \lambda_2, \lambda_3 \geq 0 \quad \dots(11)$$

$$20 + 2x + \lambda_1 + \lambda_2 + \lambda_3 = 0 \quad \dots(12)$$

$$10 + y + \lambda_1 + \lambda_2 + \lambda_3 = 0 \quad \dots(13)$$

$$z + \lambda_1 + \lambda_2 + \lambda_3 = 0 \quad \dots(14)$$

$$\lambda_1(x - 40) = 0 \quad \dots(15)$$

$$\lambda_2(x + y - 80) = 0 \quad \dots(16)$$

$$\lambda_3(x + y + z - 120) = 0 \quad \dots(17)$$

$$x \geq 40 \quad \dots(18)$$

$$x + y \geq 80 \quad \dots(19)$$

$$x + y + z \geq 120 \quad \dots(20)$$

$$x - 40 = 0, \quad \dots(21)$$

$$x + y - 80 = 0, \quad \dots(22)$$

$$x + y + z - 120 = 0, \quad \dots(23)$$

From (14), (15), (16) if $\lambda_1 \neq 0, \lambda_2 \neq 0, \lambda_3 \neq 0$.

$$\lambda_1, \lambda_2, \lambda_3 \leq 0$$

$$x - 40 = 0, \quad \dots(24)$$

$$x + y - 80 = 0, \quad \dots(25)$$

$$x + y + z - 120 = 0, \quad \dots(26)$$

$$x = 40, y = 40, z = 40$$

$$\lambda_2 = -80 - 10 - (-80) = -10$$

$$\lambda_3 = -20 - 80 - (-10) - (-80) = -10$$

$$\lambda_1 \leq 0 \text{ are also satisfied here.}$$

$$\text{Thus the conditions } \lambda_j \leq 0 \text{ are also satisfied here.}$$

$$\text{So } x_1 = x_2 = x_3 = 0$$

$$Z_{\min} = 3 \times 40^2 + (20 + 10) \times 40 = 6000$$

Q5 Solve the problem by constrained variation method

$$\text{Min } f(x) = \frac{1}{2} \left(\frac{2}{x} + \frac{2}{x} + \frac{2}{x} \right)$$

$$g_1(x) = x_1 = x_2$$

$$\text{Sub. To } g_2(x) = x_1 + x_2 + x_3 = 1. \quad \dots(1)$$

Ans. Here no. of variables $n = 3$ and no. of constraints $m = 2$

$$\text{Now } J \left(\frac{g_1, g_2}{x_1, x_2} \right) = \begin{vmatrix} \frac{\partial g_1}{\partial x_1} & \frac{\partial g_1}{\partial x_2} \\ \frac{\partial g_2}{\partial x_1} & \frac{\partial g_2}{\partial x_2} \end{vmatrix}$$

$$= \begin{vmatrix} 1 & -1 \\ 1 & 1 \end{vmatrix} = 1 + 1 = 2 \neq 0$$

Therefore x_3 can be taken as independent variable

$$\text{Again, } J \left(\frac{f, g_1, g_2}{x_1, x_2, x_3} \right) = 0 \text{ gives}$$

$$\begin{vmatrix} x_3 & x_1 & x_2 \\ 0 & 1 & -1 \\ 1 & 1 & 1 \end{vmatrix} = 0$$

$$\text{or } x_3(1+1) - x_1(0+1) + x_2(-1) = 0$$

ing.

At $x = \frac{a}{\sqrt{3}}, y = \frac{b}{\sqrt{3}}, z = \frac{c}{\sqrt{3}}$, above equation becomes,

$$\frac{2a}{\sqrt{3}a^2} dx + \frac{2b}{\sqrt{3}b^2} dy + \frac{2c}{\sqrt{3}c^2} dz = 0$$

$$\Rightarrow \frac{dx}{a} + \frac{dy}{b} + \frac{dz}{c} = 0$$

$$\Rightarrow dz = -c \left(\frac{dx}{a} + \frac{dy}{b} \right)$$

Substituting the value of dz in equation (6), we get

$$\begin{aligned} d^2F &= -\frac{8abc}{\sqrt{3}} \left[\left(\frac{dx}{a} \right)^2 + \left(\frac{dy}{b} \right)^2 + \left(\frac{dx}{a} + \frac{dy}{b} \right)^2 \right] \\ &\quad + \frac{16}{\sqrt{3}} \left[-acd \left(\frac{dx}{a} + \frac{dy}{b} \right) - bcd \left(\frac{dx}{a} + \frac{dy}{b} \right) + cdx dy \right] \\ &= -\frac{16abc}{\sqrt{3}} \left[\left(\frac{dx}{a} \right)^2 + \left(\frac{dy}{b} \right)^2 + \frac{dxdy}{ab} \right] \\ &\quad + \frac{16abc}{\sqrt{3}} \left[-\frac{dxdy}{ab} - \left(\frac{dy}{b} \right)^2 - \left(\frac{dx}{a} \right)^2 - \frac{dxdy}{ab} + \frac{dxdy}{ab} \right] \\ &= -\frac{32abc}{\sqrt{3}} \left[\left(\frac{dx}{a} \right)^2 + \left(\frac{dy}{b} \right)^2 + \frac{dxdy}{ab} \right] \\ &= -\frac{32abc}{\sqrt{3}} \left[\left(\frac{dx}{a} \right)^2 + \frac{dxdy}{ab} + \frac{1}{4} \left(\frac{dy}{b} \right)^2 + \frac{3}{4} \left(\frac{dy}{b} \right)^2 \right] \\ &= -\frac{32abc}{\sqrt{3}} \left[\left(\frac{dx}{a} + \frac{1}{2} \frac{dy}{b} \right)^2 + \frac{3}{4} \left(\frac{dy}{b} \right)^2 \right] \end{aligned}$$

which is negative definite,

$\therefore V$ is maximum at $\left(\frac{a}{\sqrt{3}}, \frac{b}{\sqrt{3}}, \frac{c}{\sqrt{3}} \right)$ and maximum

value is $\frac{8abc}{3\sqrt{3}}$.

Q.16 (a) Obtain the extreme points of

$$f = 20x_1 + 26x_2 + 4x_1x_2 - 4x_1^2 - 3x_2^2$$

(b) Determine the optimal solution for the following NLPP and check whether it maximizes or minimizes the objective function

$$\text{Optimize } Z = x_1^2 - 10x_1 + x_2^2 - 6x_2 + x_3^2 - 4x_3$$

Subject to $x_1 + x_2 + x_3 = 7$

$$x_1, x_2, x_3 \geq 0$$

[R.T.U. 2013]

Ans.(a) The given function is

$$f = 20x_1 + 26x_2 + 4x_1x_2 - 4x_1^2 - 3x_2^2$$

So, the necessary conditions for the given function f to have extreme points are

$$\frac{\partial f}{\partial x_1} = 0 \text{ and } \frac{\partial f}{\partial x_2} = 0$$

$$\text{Therefore, } \frac{\partial f}{\partial x_1} = 20 + 4x_2 - 8x_1 = 0$$

$$\text{and } \frac{\partial f}{\partial x_2} = 26 + 4x_1 - 6x_2 = 0$$

On solving both the equations, we get

$$x_1 = 7$$

$$x_2 = 9$$

$$x^* = (7, 9)$$

The point $(7, 9)$ will be an extreme point if the Hessian matrix H at this point is either positive or negative definite. But

$$\frac{\partial^2 f}{\partial x_1^2} = -8$$

$$\frac{\partial^2 f}{\partial x_1 \partial x_2} = 4$$

$$\frac{\partial^2 f}{\partial x_2^2} = -6$$

So, the Hessian matrix of f is

$$H = \begin{bmatrix} -8 & 4 \\ 4 & -6 \end{bmatrix}_{(7,9)}$$

We note that

$$H_1 = |-8| = -8 < 0$$

$$H_2 = \begin{vmatrix} -8 & 4 \\ 4 & -6 \end{vmatrix} = 32 > 0$$

Since $H_1 < 0$ and $H_2 > 0$ are of opposite signs, H is negative definite, so $x^* = (7, 9)$ is a point of maxima and the maximum value is $f_{\max} = f(7, 9) = 187$

Ans.(b) The objective function as well as the constraint are differentiable with respect to x_1, x_2 and x_3 and the Lagrangian function can be formed as

$$L(x, \lambda) = x_1^2 - 10x_1 + x_2^2 - 6x_2 + x_3^2 - 4x_3$$

$$- \lambda(x_1 + x_2 + x_3 - 7)$$

The necessary conditions for Z to be maximum or minimum are

$$\frac{\partial L}{\partial x_1} = 2x_1 - 10 - \lambda = 0$$

$$\frac{\partial L}{\partial x_2} = 2x_2 - 6 - \lambda = 0$$

$$\frac{\partial L}{\partial x_3} = 2x_3 - 4 - \lambda = 0$$

$$\frac{\partial L}{\partial \lambda} = -(x_1 + x_2 + x_3 - 7) = 0$$

The resulting solution is $x_1 = 4, x_2 = 2, x_3 = 1$ and $\lambda = -2$. To determine whether this solution results in maximization or minimization, $(n-1) = 3 - 1 = 2$ principal minors D_3 and D_4 of the determinants of order 3 and 4 are solved.

$$D_3 = \begin{vmatrix} 0 & \frac{\partial h}{\partial x_1} & \frac{\partial h}{\partial x_2} \\ \frac{\partial h}{\partial x_1} & \frac{\partial^2 f}{\partial x_1^2} - \lambda \frac{\partial^2 h}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} - \lambda \frac{\partial^2 h}{\partial x_1 \partial x_2} \\ \frac{\partial h}{\partial x_2} & \frac{\partial^2 f}{\partial x_2 \partial x_1} - \lambda \frac{\partial^2 h}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} - \lambda \frac{\partial^2 h}{\partial x_2^2} \end{vmatrix}$$

$$= \begin{vmatrix} 0 & 1 & 1 \\ 1 & 2 & 0 \\ 1 & 0 & 2 \end{vmatrix}$$

$$= -1(2) + 1(-2)$$

$$= -4$$

$$D_4 = \begin{vmatrix} 0 & 1 & 1 & 1 \\ 1 & 2 & 0 & 0 \\ 1 & 0 & 2 & 0 \\ 1 & 0 & 0 & 2 \end{vmatrix} = -1 \begin{vmatrix} 0 & 0 \\ 1 & 2 & 0 \\ 1 & 0 & 2 \end{vmatrix} + 1 \begin{vmatrix} 1 & 2 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 2 \end{vmatrix} - 1 \begin{vmatrix} 1 & 2 & 0 \\ 1 & 0 & 2 \end{vmatrix} + 1 \begin{vmatrix} 1 & 2 & 0 \\ 1 & 0 & 2 \end{vmatrix}$$

$$= -1\{1(4)\} + 1\{1(0) - 2(2)\} - 1\{1(0) - 2(-2)\}$$

$$= -4 - 4 - 4$$

$$= -12$$

Since the principal minors D_3 and D_4 are negative, the solution $x_1 = 4, x_2 = 2, x_3 = 1$ minimizes the objective function, and

$$Z_{\min} = 16 - 40 + 4 - 12 + 1 - 4 = -35$$

Q.17 (a) Find the extreme points of the function

$$f(x, y) = x^3 + 2y^3 + 3x^2 + 12y^2 + 24 \text{ and determine their nature also.}$$

(b) Solve the following :

$$\text{Min } f(x) = x_1 x_2 x_3$$

$$\text{Subject to } g = x_1 + x_2 + x_3 - 1 \quad [\text{R.T.U. 2013}]$$

Ans.(a) $f(x, y) = x^3 + 2y^3 + 3x^2 + 12y^2 + 24$

The necessary conditions for existence of extreme points are

$$\frac{\partial F}{\partial x} = 3x^2 + 6x$$

$$= x(3x + 6) = 0$$

$$\Rightarrow x = 0, -2$$

$$\frac{\partial F}{\partial y} = 6y^2 + 24y$$

$$= y(6y + 24) = 0$$

$$\Rightarrow y = 0, -4$$

LINEAR PROGRAMMING

5

CHAPTER IN A NUTSHELL

□ LINEAR PROGRAMMING PROBLEM (L.P.P.)

Linear programming defines a particular class of optimization problems in which the constraints of the system can be expressed as linear equations or inequality and the objective function is a linear function of design variable.

A linear programming problem in n variables, with some given restrictions can be stated as follows:

Find $x_1, x_2, \dots, x_n$ which

Optimizes $Z = c_1x_1 + c_2x_2 + \dots + c_nx_n$

Subject to the condition

$a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n (\leq \text{or } = \text{or } \geq) b_1$

$a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n (\leq \text{or } = \text{or } \geq) b_2$

$a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n (\leq \text{or } = \text{or } \geq) b_m$

and non-negative restrictions

$x_j \geq 0, j = 1, 2, \dots, n$

where a_{ij}, b_j and c_j are constants and x_j are variables.

□ MATHEMATICAL FORMULATION OF AN L.P.P.

First of all we must recognize a problem which can be solved by linear programming technique and then we must formulate its mathematical model according to the shape described as above.

1. Graphical Representation of Linear Programming Problem

If the objective function of an L.P.P. is a function of two variables then the problem can be solved graphically. Since in such a case each constraint is a linear inequality (or equality) in two variables, so first we draw straight lines corresponding to each constraint and determine their common solution space which is called the feasible region. It is generally a bounded or unbounded polygonal region whose boundaries are portions of the lines which correspond to the given constraints. If a L.P.P. has an optimal solution then it is given by some corner point (extreme point) of its feasible region. So by making use of this fact we decide the point which optimizes the objective function. The point so obtained satisfies

all the given constraints and optimizes Z and so it gives the required optimal solution of the given L.P.P.

2. Solution of L.P.P. by Simplex Method

Standard Form of an L.P.P.: In general procedure for solving an L.P.P. we shall use its standard (or simplified) form. The characteristic of this form are:

(i) All the constraints are in the form of equations, except the non-negative restrictions.

(ii) The components of the requirement vector b are positive.

(iii) All variables are non-negative.

(iv) The objective function is of the maximization type.

The minimization of a function $f(x)$ is equivalent to the maximization of $-f(x)$

The constraints of an L.P.P. may involve any one of the three $\leq, =, \geq$. In order to convert them into inequalities, we add slack variables to inequalities containing $\leq$ sign and introduce surplus variables to those which contain $\geq$ sign.

L.P.P. can be expressed in the matrix form as follows:

Maximize $Z = CX$ (Objective function)

such that $AX = B$; (Constraint equations)

and $X \geq 0$ (Non-negative restrictions)

□ COMPUTATIONAL PROCEDURE OF SIMPLEX METHOD

We now describe the necessary steps of the simplex method to obtain the optimal solution of a general L.P.P.

Step I : Convert the Problem into that of Max

Step II : Make all the y 's +ve

Step III : Convert Constraints into Equations

Step IV : Introduce Artificial Variables, if needed

Step V : Prepare the initial Simplex Table

Step VI : Remove the Artificial Vectors

Step VII : Test the Condition of Optimality

Step VIII : Fix the Entering Vector

Step IX : Fix the Departing Vector

Step X : Prepare New Simplex Table

SAEM.54

There are basically two simplex methods are used.

Simplex Method

Two-Phase Method

(i) **Two-Phase Method**: This method is divided into two parts: Phase I and Phase II.

In first part we try to remove all the artificial vectors added in the beginning of the initial simplex table and so we eliminate artificial variables one by one from the table as soon as they become non-basic (i.e., zero). In phase-II which starts at the end of phase I. We solve the problem to get an optimal solution as usual.

(ii) **Big-M Method**: Express the linear programming problem in the standard form. Add non-negative variables to the left hand side of all the constraints of ($=$ or $\geq$) type. These variables are called artificial variables. The purpose of introducing artificial variable is just to obtain an initial basic feasible solution. However addition of these artificial variables causes violation of the corresponding equality constraints. Therefore we would like to get rid of these variables and would not allow them to appear in the final solution. To achieve this, these artificial variables are assigned a large penalty in the objective function. Continue with the regular steps of simplex method.

3. Duality

Every linear programming has associated with another linear programming problem. The original problem is called the "Primal Problem" and the other is called its "dual".

When the number of variables is less than the constraints in a L.P.P., its solution can be found more easily solving its dual instead of the problem.

Primary problem is to find a column vector maximized $Z_p = CX$ such that $Ax \leq b$ and $X \geq 0$.

Dual problem find a column vector minimized $Z_D = b^TW$ such that $A^TW \geq C^T$.

Here Z_D is linear function C, b, A are the same as above and W is a m column vector. The dual variables are not restricted to be non-negative.

Transportation Problem: Transportation problem is a special class of the linear programming problem which deals with the situation in which a commodity is transported from sources (origins) to destinations.

The objective is to determine the amount of commodity to be transported from each source to each destination so that the total transportation cost is minimum.

Mathematical Formulation

Let us consider the m origins as $O_1, O_2, \dots, O_m$ and the n destinations as $D_1, D_2, \dots, D_n$ respectively. Let $a_i \geq 0, i = 1, 2, \dots, m$, be the amount available at the i^{th} plant O_i . Let the amount required at the j^{th} destination D_j be $b_j \geq 0, j = 1, 2, \dots, n$.

B.Tech. (III Sem.) C.S. Solved Papers

Let the cost of transporting one unit from i^{th} origin to j^{th} destination be $C_{ij}, i = 1, 2, \dots, m, j = 1, 2, \dots, n$. If $x_{ij} \geq 0$ be the amount to be transported from i^{th} origin to j^{th} destination, then the problem is to determine x_{ij} so as to minimize

$$Z = \sum_{i=1}^m \sum_{j=1}^n x_{ij} c_{ij} \quad \dots(1)$$

Subject to the constraint

$$\sum_{j=1}^n x_{ij} = a_i, i = 1, 2, \dots, m. \quad \dots(2)$$

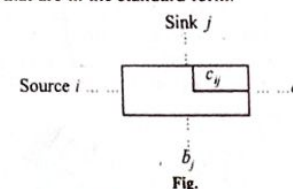
$$\sum_{i=1}^m x_{ij} = b_j, j = 1, 2, \dots, n. \quad \dots(3)$$

and $x_{ij} \geq 0$, for all i and j .

This LPP is called a **Transportation Problem**.

□ TABULAR FORM OF THE TRANSPORTATION PROBLEM

The simplex tableau serves as a very compact format for representing and manipulating linear programs. In the same spirit, we have a tableau representation for transportation problems that are in the standard form.



For a problem with m sources and n sinks, the tableau will be a table with m rows and n columns.

To \ From	D ₁	D ₂	...	D _n	Availability
O ₁	c_{11}	c_{12}	...	c_{1n}	a_1
O ₂	c_{21}	c_{22}	...	c_{2n}	a_2
⋮	⋮	⋮	...	⋮	⋮
O _m	c_{m1}	c_{m2}	...	c_{mn}	a_m
Demand	b_1	b_2	...	b_n	

Theorem The necessary and sufficient condition for the existence of a feasible solution to the transportation

problem is that $\sum_{i=1}^m a_i = \sum_{j=1}^n b_j$

□ UNBALANCED TRANSPORTATION PROBLEM

If Total supply $\neq$ Total demand, i.e. $\sum_{i=1}^m a_i \neq \sum_{j=1}^n b_j$ then transportation problem is known as an **unbalanced transportation problem**. There are two cases

PREVIOUS YEARS QUESTIONS

PART-A

Q.1 Determine the nature of the given matrix.

$$A = \begin{bmatrix} -1 & -1 & -1 \\ -1 & -2 & -2 \\ -1 & -2 & -3 \end{bmatrix}$$

[R.T.U. 2018]

Ans. $A_1 = \begin{bmatrix} -1 & -1 & -1 \\ -1 & -2 & -2 \\ -1 & -2 & -3 \end{bmatrix}$

$$A_1 = |a_{11}| = |-1| = -1$$

$$A_2 = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = \begin{vmatrix} -1 & -1 \\ -1 & -2 \end{vmatrix} = 1$$

$$A_3 = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = \begin{vmatrix} -1 & -1 & -1 \\ -1 & -2 & -2 \\ -1 & -2 & -3 \end{vmatrix} = -1$$

Since odd leading minors are negative and even leading minor are positive therefore A is negative definition.

Q.2 Write the dual of the given linear programming problem.

$$\text{Max } z = x_1 + 2x_2 - x_3$$

$$\text{s.t. } 2x_1 - 3x_2 + 4x_3 \leq 5$$

$$2x_1 - 2x_2 \leq 6$$

$$3x_1 - 3x_3 \geq 4$$

$$x_1, x_2, x_3 \geq 0$$

[R.T.U. 2018]

Ans. The given problem can be written as

$$\text{Max } Z_p = x_1 + 2x_2 - x_3$$

$$\text{s.t. } 2x_1 - 3x_2 + 4x_3 \leq 5$$

$$2x_1 - 2x_2 \leq 6$$

$$-3x_1 + 3x_3 \leq -4$$

$$x_1, x_2, x_3 \geq 0$$

The dual of the above problem

$$\text{Min } Z_D = 5w_1 + 6w_2 - 4w_3$$

$$\text{s.t. } 2w_1 + 2w_2 - 3w_3 \geq 1$$

$$-3w_1 - 2w_2 + 0w_3 \geq 2$$

$$4w_1 + 0w_2 + 3w_3 \geq -1$$

$$w_1, w_2, w_3 \geq 0$$

Q.3 Find the dual of the problem

$$\text{Min } Z = 2x_2 + 5x_3$$

$$\text{Subject to } x_1 + x_2 \geq 2,$$

$$2x_1 + x_2 + 6x_3 = 6,$$

$$x_1 - x_2 + 3x_3 \leq 4$$

$$\text{and } x_1, x_2, x_3 \geq 0$$

[R.T.U. 2017]

Ans. Standard form of the given problem is

$$\text{Max } z' = -2x_2 - 5x_3$$

$$\text{Subject to } x_1 - x_2 + 6x_3 \leq 6$$

$$-2x_1 - x_2 + 3x_3 \leq 4$$

$$\text{And } x_1, x_2, x_3 \geq 0$$

Let the dual variables are w_1, w_2, w_3 & w_4

Then dual of the given problem is

$$\text{Min } z' = -2w_1 + 6w_2 - 6w_3 + 4w_4$$

$$\text{Subject to } -w_1 + 2w_2 - 2w_3 + w_4 \geq -2$$

$$-w_1 + w_2 - w_3 - w_4 \geq -2$$

$$0.w_1 + 6w_2 - 6w_3 + 3w_4 \geq -5$$

$$\text{and } w_1, w_2, w_3, w_4 \geq 0$$

Q.4 Solve the following assignment problem.

	1	2	3	4
A	10	12	19	11
B	5	10	7	8
C	12	14	13	11
D	8	15	11	9

[R.T.U. 2014]

	1	2	3	4
A	10	12	19	11
B	5	14	7	8
C	12	14	13	11
D	8	15	11	9

(i) Subtracting smallest element of each row, from the other element of the row, we get

	1	2	3	4
A	0	2	9	1
B	0	5	2	3
C	1	3	2	0
D	0	7	3	1

(ii) Subtracting smallest element of each column, from the elements of the column and then make the assignment

	1	2	3	4
A	X	0	7	1
B	X	3	0	3
C	1	1	X	0
D	0	5	1	1

So the optimal assignment is

- A → 2
B → 3
C → 4
D → 1

PART-B

Q5. A company is spending Rs. 1000 on transportation of its units to four warehouses from three factories. What can be the maximum saving by optimal scheduling? Solve the following transportation problem:

Factory	Warehouses				Factory Capacity
	W ₁	W ₂	W ₃	W ₄	
F ₁	19	30	50	10	7
F ₂	70	30	40	60	9
F ₃	40	8	70	20	18
Requirement	5	8	7	14	34

[R.T.U. 2011, 2010]

OR
Find the initial basic feasible solution of the given transportation problem.

Warehouse → factory ↓	W ₁	W ₂	W ₃	W ₄	Factory Capacity
F ₁	19	30	50	10	7
F ₂	70	30	40	60	9
F ₃	40	8	70	20	18
Warehouse requirement	5	8	7	14	34

[R.T.U. 2010]

Find the optimum solution of the following transportation problem

	D ₁	D ₂	D ₃	D ₄	Capacity
O ₁	19	30	50	10	7
O ₂	70	30	40	60	9
O ₃	40	8	70	20	18
Demand	5	8	7	14	

[R.T.U. 2017]

OR
A company is spending Rs. 1000 on transportation of its units to four warehouses from three factories. What can be the maximum saving by optimal scheduling? Solve the following transportation problem to find optimal solution.

Factory ↓	← Warehouses →				Factory Capacity
	W ₁	W ₂	W ₃	W ₄	
F ₁	19	30	50	10	7
F ₂	70	30	40	60	9
F ₃	40	8	70	20	18
Warehouses Requirements	5	8	7	14	34

[R.T.U. 2016, 2014]

Ans. Maximum saving = Maximum profit - Transportation cost. Therefore, from the given data we have to find the maximum profit. Since the problem is of maximization, we need to convert it to minimization form by subtracting all the profits from the highest profit, i.e. 70. The problem now becomes to minimize the cost in:

Factory	Warehouses				Factory Capacity
	51	40	20	60	7
	0	40	30	10	9
	30	62	0	50	18
Requirement	5	8	7	14	34

Find initial BFS using Vogel's approximation method:

		Availability From Facilities							
		51	40	20	60	7	20	20	20
		0	40	30	10	9	10	-	-
		30	62	0	50	18	30	50	20
Requirement		5	8	7	14	34			

Thus the initial BFS is $x_{12}=7, x_{24}=9, x_{31}=5, x_{32}=1, x_{33}=7, x_{34}=5$.

The number of occupied cells is $4+3-1=6$, hence the solution is non-degenerate.

Optimality Test: Applying MODI method:

	D ₁	D ₂	D ₃	D ₄	Capacity
O ₁	19	30	50	10	7
O ₂	70	30	40	60	9
O ₃	40	8	70	20	18
Demand	5	8	7	14	

Since all cell evaluations are negative, we have obtained optimal solution:

$x_{12}=7, x_{24}=9, x_{31}=5, x_{32}=1, x_{33}=7, x_{34}=5$.
Therefore, the maximum profit is
 $Z = 7 \times 30 + 9 \times 60 + 5 \times 40 + 1 \times 8 + 7 \times 70 + 5 \times 20 = \text{Rs. } 1548$
and hence the maximum saving is $1548 - 1000 = \text{Rs. } 548$.

Q6. Solve the following LPP using simplex method:

Minimize $Z = x_1 - 3x_2 + 2x_3$
Subject to $3x_1 - x_2 + 3x_3 \leq 7$
 $-2x_1 + 4x_2 \leq 12$
 $-4x_1 + 3x_2 + 8x_3 \leq 10$
 $x_1, x_2, x_3 \geq 0$

And $x_1, x_2, x_3 \geq 0$
OR
Solve by simplex method.

Min $Z = x_1 - 3x_2 + 2x_3$
s.t. $3x_1 - x_2 + 3x_3 \leq 7$
 $-2x_1 + 4x_2 \leq 12$
 $-4x_1 + 3x_2 + 8x_3 \leq 10$
 $x_1, x_2, x_3 \geq 0$

[R.T.U. 2018]

Ans. First we convert the problem of minimization to maximization.

Let $Z' = -Z$

Consider the problem

Max. $Z' = -x_1 + 3x_2 - 2x_3 + 0x_4 + 0x_5 + 0x_6$
Subject to: $3x_1 - x_2 + 3x_3 + x_4 = 7$
 $-2x_1 + 4x_2 + x_5 = 12$
 $-4x_1 + 3x_2 + 8x_3 + x_6 = 10$
 $x_1, x_2, x_3, x_4, x_5, x_6 \geq 0$

Table-1

	C ₁	C ₂	C ₃	C ₄	C ₅	C ₆	Min. ratio
X ₄	3	-2	0	0	0	0	
X ₅	1	3	1	0	0	0	
X ₆	1	3	1	0	0	0	
X ₁	1	3	1	0	0	0	
X ₂	1	3	1	0	0	0	
X ₃	1	3	1	0	0	0	
X ₄	1	3	1	0	0	0	
X ₅	1	3	1	0	0	0	
X ₆	1	3	1	0	0	0	
X ₁	1	3	1	0	0	0	
X ₂	1	3	1	0	0	0	
X ₃	1	3	1	0	0	0	
X ₄	1	3	1	0	0	0	
X ₅	1	3	1	0	0	0	
X ₆	1	3	1	0	0	0	
X ₁	1	3	1	0	0	0	
X ₂	1	3	1	0	0	0	
X ₃	1	3	1	0	0	0	
X ₄	1	3	1	0	0	0	
X ₅	1	3	1	0	0	0	
X ₆	1	3	1	0	0	0	
X ₁	1	3	1	0	0	0	
X ₂	1	3	1	0	0	0	
X ₃	1	3	1	0	0	0	
X ₄	1	3	1	0	0	0	
X ₅	1	3	1	0	0	0	
X ₆	1	3	1	0	0	0	
X ₁	1	3	1	0	0	0	
X ₂	1	3	1	0	0	0	
X ₃	1	3	1	0	0	0	
X ₄	1	3	1	0	0	0	
X ₅	1	3	1	0	0	0	
X ₆	1	3	1	0	0	0	
X ₁	1	3	1	0	0	0	
X ₂	1	3	1	0	0	0	
X ₃	1	3	1	0	0	0	
X ₄	1	3	1	0	0	0	
X ₅	1	3	1	0	0	0	
X ₆	1	3	1	0	0	0	
X ₁	1	3	1	0	0	0	
X ₂	1	3	1	0	0	0	
X ₃	1	3	1	0	0	0	
X ₄	1	3	1	0	0	0	
X ₅	1	3	1	0	0	0	
X ₆	1	3	1	0	0	0	
X ₁	1	3	1	0	0	0	
X ₂	1	3	1	0	0	0	
X ₃	1	3	1	0	0	0	
X ₄	1	3	1	0	0	0	
X ₅	1	3	1	0	0	0	
X ₆	1	3	1	0	0	0	
X ₁	1	3	1	0	0	0	
X ₂	1	3	1	0	0	0	
X ₃	1	3	1	0	0	0	
X ₄	1	3	1	0	0	0	
X ₅	1	3	1	0	0	0	
X ₆	1	3	1	0	0	0	
X ₁	1	3	1	0	0	0	
X ₂	1	3	1	0	0	0	
X ₃	1	3	1	0	0	0	
X ₄	1	3	1	0	0	0	
X ₅	1	3	1	0	0	0	
X ₆	1	3	1	0	0	0	
X ₁	1	3	1	0	0	0	
X ₂	1	3	1	0	0	0	
X ₃	1	3	1	0	0	0	
X ₄	1	3	1	0	0	0	
X ₅	1	3	1	0	0	0	
X ₆	1	3	1	0	0	0	
X ₁	1	3	1	0	0	0	
X ₂	1	3	1	0	0	0	
X ₃	1	3	1	0	0	0	
X ₄	1	3	1	0	0	0	
X ₅	1	3	1	0	0	0	
X ₆	1	3	1	0	0	0	
X ₁	1	3	1	0	0	0	
X ₂	1	3	1	0	0	0	
X ₃	1	3	1	0	0	0	
X ₄	1	3	1	0	0	0	
X ₅	1	3	1	0	0	0	
X ₆	1	3	1	0	0	0	
X ₁	1	3	1	0	0	0	
X ₂	1	3	1	0	0	0	
X ₃	1	3	1	0	0	0	
X ₄	1	3	1	0	0	0	
X ₅	1	3	1	0	0	0	
X ₆	1	3	1	0	0	0	
X ₁	1	3	1	0	0	0	
X ₂	1	3	1	0	0	0	
X ₃	1	3	1	0	0	0	
X ₄	1	3	1	0	0	0	
X ₅	1	3	1	0	0	0	
X ₆	1	3	1	0	0	0	
X ₁	1	3	1	0	0	0	
X ₂	1	3	1	0	0	0	
X ₃	1	3	1	0	0	0	
X ₄	1	3	1	0	0	0	
X ₅	1	3	1	0	0	0	
X ₆	1	3	1	0	0	0	
X ₁	1	3	1	0	0	0	
X ₂	1	3	1	0	0	0	
X ₃	1	3	1	0	0	0	
X ₄	1	3	1	0	0	0	
X ₅	1	3	1	0	0	0	
X ₆	1	3	1	0	0	0	
X ₁	1	3	1	0	0	0	
X ₂	1	3	1	0	0	0	
X ₃	1	3	1	0	0	0	
X ₄	1	3	1	0	0	0	
X ₅	1	3	1	0	0	0	
X ₆	1	3	1	0	0	0	
X ₁	1	3	1	0	0	0	
X ₂	1	3	1	0	0	0	
X ₃	1	3	1	0	0	0	
X ₄	1	3	1	0	0	0	
X ₅	1	3	1	0	0	0	
X ₆	1	3	1	0	0	0	
X ₁	1	3	1	0	0	0	
X ₂	1	3	1	0	0	0	
X ₃	1	3	1	0	0	0	
X ₄	1	3	1	0	0	0	
X ₅	1	3	1	0	0	0	
X ₆	1	3	1	0	0	0	
X ₁	1	3	1	0	0	0	
X ₂	1	3	1	0	0	0	
X ₃	1	3	1	0	0	0	
X ₄	1	3	1	0	0	0	

Q.8 Solve the following assignment problem :

Jobs

Machines

	J ₁	J ₂	J ₃	J ₄	J ₅
M ₁	10	4	5	3	11
M ₂	13	11	9	12	10
M ₃	12	3	10	1	9
M ₄	9	1	11	4	8
M ₅	8	6	7	3	10

[R.T.U. 2017]

Ans. Step 1 : Find each row min. element and subtract from row.

	J ₁	J ₂	J ₃	J ₄	J ₅	
M ₁	7	1	2	0	8	(-3)
M ₂	4	2	0	3	1	(-9)
M ₃	11	2	9	0	8	(-1)
M ₄	8	0	10	3	7	(-1)
M ₅	5	3	4	0	7	(-3)

Step 2 : Subtract min. of element of each column from that column.

	J ₁	J ₂	J ₃	J ₄	J ₅	
M ₁	3	1	2	0	7	
M ₂	0	2	0	3	0	
M ₃	7	2	9	0	7	
M ₄	4	0	10	3	9	
M ₅	1	3	4	0	6	
	(-4)	(-0)	(-0)	(-0)	(-1)	

Step 3 : Make assignment in the opportunity cost table.

	J ₁	J ₂	J ₃	J ₄	J ₅	
M ₁	1	∞	[0]	[∞]	5	
M ₂	∞	3	∞	5	[0]	
M ₃	5	1	7	[0]	5	
M ₄	3	[0]	9	4	5	
M ₅	[0]	3	3	1	5	

Step 4 : No. of Assg. = 5, No. of rows = 5

	J ₁	J ₂	J ₃	J ₄	J ₅	
M ₁	1	∞	[0]	∞	5	
M ₂	∞	3	∞	5	[0]	
M ₃	5	1	7	[0]	5	
M ₄	3	[0]	9	4	5	
M ₅	[0]	3	3	1	5	

So, Optimal solution is

Work	Job	Cost
M ₁	J ₃	5
M ₂	J ₅	10
M ₃	J ₄	1
M ₄	J ₂	1
M ₅	J ₁	8
Total		25

Q.9 A department head has five subordinates and five jobs to be done. The subordinates differ in efficiency and jobs differ in their intrinsic difficulty. The estimate of the times each man would take to perform each job is given in effectiveness matrix. How should the tasks be allocated on one to one basis, so as to minimize the total man hours.

Subordinates

	I	II	III	IV	V
A	1	3	2	3	6
B	2	4	3	1	5
C	5	6	3	4	6
D	3	1	4	2	2
E	1	5	6	5	4

Jobs

[R.T.U. 2017]

Ans. Step 1 : Find min. element in each row and subtract it.

	I	II	III	IV	V	
A	0	2	1	2	5	(-1)
B	1	3	2	0	4	(-1)
C	2	3	0	1	3	(-3)
D	2	0	3	1	1	(-1)
E	0	4	5	4	3	(-1)

Step 2 : Find min. element in each column and subtract it.

	I	II	III	IV	V	
A	0	2	1	2	4	
B	1	3	2	0	3	
C	2	3	0	1	2	
D	2	0	3	1	0	
E	0	4	5	4	2	
	(-0)	(-0)	(-0)	(-0)	(-1)	

Step 3 : Make the assignment in the opportunity cost table.
Row wise and column wise assignment shown in table.

	I	II	III	IV	V	
A	∞	[0]	∞	∞	2	
B	3	3	3	[0]	3	
C	3	2	[0]	∞	1	
D	4	∞	4	1	[0]	
E	[0]	2	4	2	∞	

Q.15 Write the dual of the problem.

$$\text{Max } Z_p = 2x_1 + 4x_2$$

$$\text{Subject to } 2x_1 + 3x_2 \leq 48$$

$$x_1 + 3x_2 \leq 42$$

$$x_1 + x_2 \leq 21$$

$$x_1, x_2 \geq 0$$

[R.T.U. 2014]

Ans. This problem has from max.

$$Z_p = CX$$

$$AX \leq b$$

$$X \geq 0$$

...(1)

the dual of (1) is

$$\text{minimize } Z_D = b^T w$$

$$A^T w \geq C^T$$

$$w \geq 0$$

$$\text{Here } C = (2, 4), A = \begin{bmatrix} 2 & 3 \\ 1 & 3 \\ 1 & 1 \end{bmatrix}; b = \begin{bmatrix} 48 \\ 42 \\ 21 \end{bmatrix}$$

Hence dual of given problem

$$\text{minimize } Z_D = \begin{bmatrix} 48 \\ 42 \\ 21 \end{bmatrix}^T (w_1, w_2, w_3)$$

$$\text{Subjected to } \begin{bmatrix} 2 & 3 \\ 1 & 3 \\ 1 & 1 \end{bmatrix}^T (w_1, w_2, w_3) \geq (2, 4)^T$$

$$w_1, w_2, w_3 \geq 0$$

$$\text{Minimize } Z_D = 48w_1 + 42w_2 + 21w_3$$

$$\text{Subjected to } 2w_1 + w_2 + w_3 \geq 2$$

$$3w_1 + 3w_2 + w_3 \geq 4$$

$$w_1, w_2, w_3 \geq 0$$

...(2)

Equation (2) is dual given problem.

Q.16 Solve by Simplex method :

$$\text{Min } Z = x_1 - 3x_2 + 2x_3$$

$$\text{s.t. } 3x_1 - x_2 + 3x_3 \leq 7$$

$$-2x_1 + 4x_2 \leq 12$$

$$-4x_1 + 3x_2 + 8x_3 \leq 10$$

$$\text{and } x_1, x_2, x_3 \geq 0$$

[R.T.U. 2014, 08]

Ans. This is the problem of minimization so converting the objective function from minimization to maximization, we have

$$\text{Max } -Z = -x_1 + 3x_2 - 2x_3 = \text{Max } Z'$$

		C_j	-1	3	-2	0	0	0
Basic variables	C_B	X_B	x_1	x_2	x_3	x_4	x_5	x_6
x_4	0	7	3	-1	3	1	0	0
$\leftarrow x_5$	0	12	-2	4	0	0	1	0
x_6	0	10	-4	3	8	0	0	1
$x_1 = x_2 = x_3 = 0$	$Z' = 0$ $Z = 0$		1	-3	2	0	0	0

$$\text{Min Ratio } \left[\frac{12}{4}, \frac{10}{3} \right]$$

$\leftarrow x_4$	0	10	$\frac{5}{2}$	0	3	1	$\frac{1}{4}$	0
$\rightarrow x_2$	3	3	$-\frac{1}{2}$	1	0	0	$\frac{1}{4}$	0
x_6	0	1	$-\frac{5}{2}$	0	8	0	$-\frac{3}{4}$	1
$x_1 = x_3 = x_5 = 0$	$Z' = 9$ $Z = -9$		$-\frac{1}{2}$	0	2	0	$\frac{3}{4}$	0

$$\text{Min. Ratio } \left[\frac{10}{5/2} \right]$$

$\rightarrow x_1$	-1	4	1	0	$\frac{6}{5}$	$\frac{2}{5}$	$\frac{1}{10}$	0
x_2	3	5	0	1	$\frac{3}{5}$	$\frac{1}{5}$	$\frac{3}{10}$	0
x_6	0	11	0	0	11	1	$-\frac{1}{2}$	1
			0	0	$\frac{13}{5}$	$\frac{1}{5}$	$\frac{8}{10}$	0

All $Z_j - C_j > 0$

Optimal solution $x_1 = 4$

$$x_2 = 5$$

$$x_3 = 0$$

$$Z_{\max} = 4 \times (-1) + 5 \times 3 + 11 \times 0 = -11$$

$$\text{Min } Z = -11$$

Ans.

Q.17 Solve the following problem using duality :

$$\text{Minimize } Z = 300x_1 + 110x_2 \text{ subject to :}$$

$$30x_1 + 5x_2 \geq 6, 20x_1 + 10x_2 \geq 8, x_1, x_2 \geq 0$$

[R.T.U. 2014]

Ans. Given L.P.P. (Primal) is

$$\text{Min } Z = 300x_1 + 110x_2$$

$$\text{subjected to : } 30x_1 + 5x_2 \geq 6$$

$$20x_1 + 10x_2 \geq 8$$

$$x_1, x_2 \geq 0$$

AEM.64

The dual of the given problem is

$$\text{Max } Z' = 6w_1 + 8w_2$$

$$\text{subjected to : } 30w_1 + 20w_2 \leq 300$$

$$5w_1 + 10w_2 \leq 110$$

$$\text{and } w_1, w_2 \geq 0$$

The standard form of dual is

$$\text{Max } Z' = 6w_1 + 8w_2 + 0w_3 + 0w_4$$

$$\text{subjected to : } 30w_1 + 20w_2 + w_3 = 300$$

$$5w_1 + 10w_2 + w_4 = 110$$

$$w_1, w_2, w_3, w_4 \geq 0$$

Table-1

		C_j	6	8	0	0	Min. ratio
C_B	B	X_B	y_1	y_2	y_3	y_4	
0	w_3	300	30	20	1	0	$300/20 = 15$
0	w_4	110	5	10	0	1	$5/10 = 1/2 \rightarrow$
		$Z_j - C_j$	-6	-8	0	0	

$\uparrow$ $\downarrow$
 incoming outgoing

Table-2

		C_j	6	8	0	0	Min. ratio
C_B	B	X_B	y_1	y_2	y_3	y_4	
0	w_3	80	20	0	1	-2	$80/20 = 4 \rightarrow$
8	w_2	11	1/2	1	0	1/10	22
		$Z_j - C_j$	-2	0	0	8/10	

$\uparrow$ $\downarrow$
 incoming outgoing

Table-3

		C_j	6	8	0	0
C_B	B	X_B	y_1	y_2	y_3	y_4
6	w_1	4	1	0	1/20	-1/10
8	w_2	9	0	1	-1/40	3/20
		$Z_j - C_j$	0	0	1/10	3/5

$\therefore$ All $Z_j - C_j \geq 0$ so the optimal solution is
 $w_1 = 4, w_2 = 9$ with $\text{max } Z' = 24 + 72 = 96$

Q.18 Solve the following LPP :

$$\text{Maximize } Z = x_1 + 2x_2 + 3x_3 - x_4$$

$$\text{s.t. : } x_1 + 2x_2 + 3x_3 = 15, 2x_1 + x_2 + 5x_3 = 20$$

$$x_1 + 2x_2 + x_3 + x_4 = 10 \text{ and } x_1, x_2, x_3, x_4 \geq 0$$

[R.T.U. 2014]

Second Simplex Table

B	X _B	C _B	x ₁	x ₂	s ₁	s ₂
x ₁	21/13	-1	1	0	-7/13	1/13
x ₂	10/13	-1	0	1	1/13	-2/13
		c _j	-1	-1	0	0
		z _j	-1	-1	6/13	1/13
		Δ _j	0	0	-6/13	-1/13

Since all Δ_j ≤ 0, so the solution under test is optimal.
Optimal solution is

$$x_1 = \frac{21}{13},$$

$$x_2 = \frac{10}{13},$$

$$\max Z = -x_1 - x_2$$

$$= -\frac{21}{13} - \frac{10}{13} = -\frac{31}{13}$$

Or $\min Z = 31/13$

Q.21 Solve the following LPP by converting it into its dual :

Minimize $Z = x_1 + x_2$

Subject to

$$3x_1 + 2x_2 \geq 4$$

$$-x_1 + 3x_2 \geq 5$$

$$4x_1 + 2x_2 \geq 5$$

$$2x_1 + x_2 \geq 1$$

And $x_1, x_2 \geq 0$

[R.T.U. 2017]

Ans. Given problem

Minimize $Z = x_1 + x_2$

Subject to $3x_1 + 2x_2 \geq 4$

$$-x_1 + 3x_2 \geq 5$$

$$4x_1 + 2x_2 \geq 5$$

$$2x_1 + x_2 \geq 1$$

and $x_1, x_2 \geq 0$

The above problem is in standard form so it can be converted into matrix form as follows -

Min $Z = CX$

Subject to $AX \geq b$

$$X \geq 0$$

where $A = \begin{bmatrix} 3 & 2 \\ -1 & 3 \\ 4 & 2 \\ 2 & 1 \end{bmatrix}$, $X = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$,

and $b = \begin{bmatrix} 4 \\ 5 \\ 5 \\ 1 \end{bmatrix}$, $C = [1 \ 1]$

Duality of this problem will be as follows-

Maximize $Z_D = b^T W$

Subject to $A^T W \leq C^T$

and $W \geq 0$

where $b^T = [4 \ 5 \ 5 \ 1]$, $C^T = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$

$$A^T = \begin{bmatrix} 3 & -1 & 4 & 2 \\ 2 & 3 & 2 & 1 \end{bmatrix}$$

and $W = \begin{bmatrix} W_1 \\ W_2 \\ W_3 \\ W_4 \end{bmatrix}$

So dual problem of the given primal problem is -

Max. $Z_D = 4W_1 + 5W_2 + 5W_3 + W_4$

Subject to $3W_1 - W_2 + 4W_3 + 2W_4 \leq 1$

$$2W_1 + 3W_2 + 2W_3 + W_4 \leq 1$$

and $W_1, W_2, W_3, W_4 \geq 0$

Now we can solve it by simplex method to convert these inequalities into equations we add two slack variable W_5 and W_6 .

Max. $Z_D = 4W_1 + 5W_2 + 5W_3 + W_4 + 0.W_5 + 0.W_6$

Subject to

$$3W_1 - W_2 + 4W_3 + 2W_4 + W_5 + 0.W_6 = 1$$

$$2W_1 + 3W_2 + 2W_3 + W_4 + 0.W_5 + W_6 = 1$$

and $W_1, W_2, W_3, W_4, W_5, W_6 \geq 0$

So coefficient matrix

$$A = \begin{bmatrix} 3 & -1 & 4 & 2 & 1 & 0 \\ 2 & 3 & 2 & 1 & 0 & 1 \end{bmatrix}$$

$$= (\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6) \text{ (Let)}$$

here initial basis $B = (\alpha_5, \alpha_6) = I_2$

First Simplex Table :

			C _j →	4	5	5	1	0	0
C _B	B	W _B	b	y ₁	y ₂	y ₃	y ₄	y ₅	y ₆
0	α ₅	W ₅	1	3	-1	4	2	1	0
0	α ₆	W ₆	1	2	3	2	1	0	1
	(Z _j - C _j) →			-4	-5	-5	-1	0	0

here $Z_j - C_j = -5$ is minimum. So α₃ is entering vector.

and Min. $\left\{ \frac{1}{4}, \frac{1}{2} \right\} = \frac{1}{4}$, which is related to α₅, so α₅ is

departing vector.

Second Simplex Table :

			C _j →	4	5	5	1	0	0
C _B	B	W _B	b	y ₁	y ₂	y ₃	y ₄	y ₅	y ₆
5	α ₃	W ₃	1/4	3/4	-1/4	1	1/2	-	0
0	α ₆	W ₆	1/2	1/2	7/2	0	0	-	1
	(Z _j - C _j) →			-1/4	-25/4	0	3/2	-	0

here $Z_j - C_j = \frac{-25}{4}$ is Minimum. So α₂ is entering vector.

and Min. $\left\{ \frac{1/4}{1/4}, \frac{1/2}{7/2} \right\} = \frac{1}{7}$, which is related to α₆, so

α₆ is departing vector.

Third Simplex Table :

			C _j →	4	5	5	1	0	0
C _B	B	W _B	b	y ₁	y ₂	y ₃	y ₄	y ₅	y ₆
5	α ₃	W ₃	2/7	11/14	0	1	1/2	-	-
5	α ₂	W ₂	1/7	1/7	1	0	0	-	-
	(Z _j - C _j) →			9/14	0	0	3/2	-	-

In above table all $Z_j - C_j \geq 0$, So solution is optimal.

So $W_1 = 0, W_2 = 1/7, W_3 = 2/7,$
 $W_4 = 0, W_5 = 0, W_6 = 0$

$$\max. Z_D = 4.0 + 5 \times \frac{1}{7} + 5 \times \frac{2}{7} + 0$$

$$= 15/7$$

So Min. $Z = 15/7$

Q.22 Consider the linear programming problem :

Maximize $Z = 3x_1 + 5x_2 + 4x_3$

Subject to $2x_1 + 3x_2 \leq 8$

$$2x_2 + 5x_3 \leq 10$$

$$3x_1 + 2x_2 + 4x_3 \leq 15 \text{ and}$$

$$x_1, x_2, x_3 \geq 0$$

$$A = \begin{pmatrix} 1 & -3 & 4 & -4 \\ -1 & 3 & -4 & 4 \\ -2 & 2 & 0 & 0 \\ 0 & 2 & -1 & 1 \end{pmatrix}, B = \begin{pmatrix} 5 \\ -5 \\ 3 \\ 5 \end{pmatrix}$$

Therefore, the dual of the given LPP becomes

$$\text{Max. } Z_D = b^T w$$

$$\& \text{ t. } A^T w \geq C^T$$

$$\text{and } w \geq 0$$

$$\text{Max. } Z_D = (5 \ -5 \ 3 \ 5) \begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{pmatrix}$$

$$\text{s.t. } \begin{pmatrix} 1 & -1 & -2 & 0 \\ -3 & 3 & 2 & 2 \\ 4 & -4 & 0 & -1 \\ -4 & 4 & 0 & 1 \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{pmatrix} \leq \begin{pmatrix} 1 \\ 1 \\ 1 \\ -1 \end{pmatrix}$$

$$\text{and } w_1, w_2, w_3, w_4 \geq 0$$

$$\text{or Max. } Z_D = 5w_1 - 5w_2 + 3w_3 + 5w_4$$

$$\text{s.t. } w_1 - w_2 - 2w_3 \leq 1$$

$$-3w_1 + 3w_2 + 2w_3 + 2w_4 \leq 1$$

$$4w_1 - 4w_2 + 0w_3 - w_4 \leq 1$$

$$-4w_1 + 4w_2 + 0w_3 + w_4 \leq -1$$

$$\text{and } w_1, w_2, w_3, w_4 \geq 0$$

$$\text{or Max. } Z_D = 5w_1 + 3w_3 + 5w_4$$

$$\text{s.t. } w_1 - 2w_3 \leq 1$$

$$-3w_1 + 2w_3 + 2w_4 \leq 1$$

$$4w_1 + 0w_3 - w_4 \leq 1$$

$$-4w_1 + 0w_3 - w_4 \leq 1$$

$$\text{and } w_3, w_4 \geq 0, w_1 \text{ is unrestricted in sign.}$$

Q.25 (a) Solve the following LPP

$$\text{Maximize } z = -4x_1 - 3x_2 - 9x_3$$

$$\text{s.t. } 2x_1 + 4x_2 + 6x_3 \geq 15$$

$$6x_1 + x_2 + 6x_3 \geq 12$$

$$x_1, x_2, x_3 \geq 0$$

[R.T.U. Dec. 2016]

(b) Write the dual of the linear programming problem Maximize $z = x_1 - 2x_2 + 3x_3$

$$\text{s.t. } 2x_1 + 5x_3 \leq 16$$

$$5x_2 + 4x_3 \geq 18$$

$$x_1 + x_2 + x_3 = 10$$

$$x_1 \geq 0, x_2 \leq 0, x_3 \text{ unrestricted in sign}$$

[Note : For give condition replace $x_2 \leq 0$ with $x_2 \geq 0$]

[R.T.U. Dec. 2016]

Ans.(a) Standard form of given problem is

$$\text{Max } Z = -4x_1 - 3x_2 - 9x_3 + 0x_4 + 0x_5 - Mx_6 - Mx_7$$

$$\text{s.t. } 2x_1 + 4x_2 + 6x_3 - x_4 + x_6 = 15$$

$$6x_1 + x_2 + 6x_3 - x_5 + x_7 = 12$$

$$\text{and } x_1, x_2, x_3, x_4, x_5, x_6, x_7 \geq 0$$

where x_4 and x_5 are surplus variable and x_6 and x_7 are artificial variable.

$$\text{coefficient matrix } A = \begin{bmatrix} 2 & 4 & 6 & -1 & 0 & 1 & 0 \\ 6 & 1 & 6 & 0 & -1 & 0 & 1 \end{bmatrix}$$

Clearing initial basic feasible solution (BFS) is $B = (x_6, x_7)$

First Simplex Table

		C_j	-4	-3	-9	0	0	-M	-M
C_B	X_B	b	x_1	x_2	x_3	x_4	x_5	x_6	x_7
-M	x_6	15	2	4	6	-1	0	1	0
-M	x_7	12	6	1	6	0	-1	0	1
$Z_j - C_j$			-8M+4	-5M+3	-12M+9	M	M	0	0
			↑						

$\therefore Z_j - C_j$ is the most negative so x_1 will be entering vector.

$$\text{Min}_i \left(\frac{X_{Bi}}{y_{ii}}; y_{ii} \geq 0 \right) = \left\{ \frac{15}{2}, \frac{12}{6} \right\} = \frac{12}{6}$$

So 6 will be key element and x_7 will be the departing vector now

Second Simplex Table

		C_j	-4	-3	-9	0	0	-M
C_B	X_B	b	x_1	x_2	x_3	x_4	x_5	x_6
-M	x_6	11	0	11/3	4	-1	1/3	1 →
-4	x_1	2	1	1/6	1	0	-1/6	0
$Z_j - C_j$			0	$-\frac{11}{3}M + \frac{7}{3}$	-4M+5	M	$\frac{M}{3} + \frac{2}{3}$	0

$\therefore Z_j - C_j$ is the most negative so x_2 will be entering vector.

And on the similar way x_6 will be departing vector.

Third Simplex Table

		C_j	-4	-3	-9	0	0
C_B	X_B	b	x_1	x_2	x_3	x_4	x_5
-3	x_2	3	0	1	12/11	-3/11	1/11
-4	x_1	3/2	1	0	9/11	1/6	-2/11
$Z_j - C_j$			0	0	27/11	5/33	5/11

Here all $Z_j - C_j \geq 0$ so we have optimal solution

$$\text{i.e. } x_1 = 3/2; x_2 = 3 \text{ and } x_3 = 0$$

$$\text{So max } Z = -4x_1 - 3x_2 - 9x_3$$

$$= -4 \times \frac{3}{2} - 3 \times 3 - 9 \times 0$$

$$= -6 - 9$$

$$= -15$$

Ans.(b) First of all we convert the given L.P.P. in standard primal form. Here the problem is of maximization so all the inequalities must be in the form $\leq$.

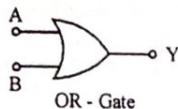
DE.4

Distributive law

$$A(B + C) = AB + AC$$

OR Operation

$$Y = A + B$$



AND Operation

$$Y = A \cdot B$$



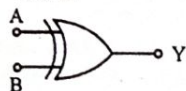
NOT Operation

I/P	O/P
0	1
1	0



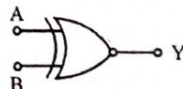
Ex-OR Operation

$$\bar{A}B + A\bar{B} = A \oplus B$$



Ex-NOR Operation

$$Y = A \odot B = \bar{A}\bar{B} + AB$$



NAND Operation

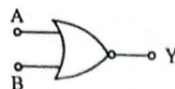
$$Y = \overline{A \cdot B}$$



B.Tech. (III Sem.) C.S. Solved Papers

NOR Operation

$$Y = \overline{A + B}$$



NAND and NOR are universal gates by which we can design logical expression.

De Morgan's Theorem

$$\overline{A + B} = \bar{A} \cdot \bar{B}$$

$$\overline{A \cdot B} = \bar{A} + \bar{B}$$

Duality Theorem

$$A(B + C) = AB + AC$$

$$A + (BC) = (A + B)(A + C)$$

Normal Relation

$$A + 0 \rightarrow B + A$$

$$A + (B + C) \rightarrow (A + B) + C$$

$$A(B + C) \rightarrow AB + AC$$

$$A + 0 = A$$

$$A + A = A$$

$$A + \bar{A} = 1$$

$$\bar{\bar{A}} = A$$

$$\overline{A + B} = \bar{A}\bar{B}$$

$$A + AB = A$$

$$A + \bar{A}B = A + B$$

Dual Relation

$$AB = BA$$

$$A(BC) = (AB)C$$

$$A + BC = (A + B)(A + C)$$

$$A.1 = A$$

$$A.A = A$$

$$A.\bar{A} = 0$$

$$\bar{\bar{A}} = A$$

$$\overline{AB} = \bar{A} + \bar{B}$$

$$A(A + B) = A$$

$$A(\bar{A} + B) = AB$$

Digital Electronics

Ans. $2x^1 + 3x^0 + 1.x^1 + 2x^0 = 1.x^2 + 0.x^1 + 1.x^0$
 $\Rightarrow 3x + 5 = x^2 + 1$
 $\Rightarrow x^2 - 3x - 4 = 0$
 $\Rightarrow (x - 4)(x + 1) = 0$
 $\Rightarrow x = 4 \text{ or } x = -1$

Q.4 Explain the reflected code.

[R.T.U. 2018]

Ans. Reflective Codes (or) Self Complementing Codes : A code is said to be a reflective code or gray code, if the code word of the 9's complement of N i.e. 9 - N can be obtained from the code word of N by interchanging all the 1's as 0's and 0's as 1's i.e., the code word for 9 is the complement for the code 0, 8 for 1, 7 for 2, 6 for 3 and 5 for 4. Note that 4221, 5211, 6, 4, 2, -3 and Ex-3 codes are reflective, whereas 8421 code is not.

Q.5 Find the 11's complement of following numbers :

(i) $(935)_{12}$

(ii) $(267)_{12}$

[R.T.U. 2017]

Ans.(i) $(935)_{12}$

The $(r - 1)$'s complement of any number N is defined as

$$(r^n - 1) - N \quad \dots(1)$$

Where $r = \text{base}$

$n = \text{number of bits in the number N}$

$N = \text{number}$

So 11's complement of $(935)_{12}$ is

$$(12^3 - 1) - (935)_{12}$$

$$= (1728)_{10} - (935)_{12} - 1$$

$$= (1000)_{12} - (935)_{12} - 1$$

$$\therefore (1728)_{10} = (1000)_{12}$$

$$= (287)_{12} - 1$$

$$= (286)_{12}$$

(ii) $(267)_{12}$

According to formula

$$(r^n - 1) - N$$

$$= (12^3 - 1) - (267)_{12} - 1$$

$$= (1728)_{10} - (267)_{12} - 1$$

$$= (1000)_{12} - (267)_{12} - 1$$

$$= (955)_{12} - 1$$

$$= (954)_{12}$$

Q.6 X and Y are successive digits in positional number system and $(XY)_r = (25)_{10}$ and $(YX)_r = (31)_{10}$. Determine the value of X, Y and r. [R.T.U. 2017]

DE.5

Ans. $(XY)_r = (25)_{10}$
 $X.r^1 + Y.r^0 = 25$... (i)

$(YX)_r = (31)_{10}$
 $Y.r^1 + X.r^0 = 31$... (ii)

Since X, Y are positional number

So $y = X + 1$... (iii)

After solving equation (i), (ii) & (iii)

$$X = 3, Y = 4 \text{ and } r = 7$$

Q.7 A register contains 2's complement 10010110. What will be the content of register if it is divided by 2. [R.T.U. 2017]

Ans. Register contain

1	0	0	1	0	1	1	0
---	---	---	---	---	---	---	---

If we do right shifting then the number will be divided by two and the given number is negative so one has to append. Therefore the contents of register will be

1	1	0	0	1	0	1	1
---	---	---	---	---	---	---	---

Q.8 Find the radix value if:

$$(23)_r + (12)_r = (101)_r \quad [R.T.U. 2013, Raj. Univ. 2007]$$

Ans. $(23)_r + (12)_r = (101)_r$

$$\Rightarrow 2r^1 + 3r^0 + 1.r^1 + 2r^0 = 1.r^2 + 0.r^1 + 1.r^0$$

$$\Rightarrow 3r + 5 = r^2 + 1$$

$$\Rightarrow r^2 - 3r - 4 = 0$$

$$\Rightarrow (r - 4)(r + 1) = 0$$

$$\Rightarrow r = 4 \text{ or } -1$$

Since the radix can not be negative, therefore radix $r = 4$.

Q.9 Simplify the expression using boolean algebra.

$$\overline{AB} + ABC + A(B + \bar{A}\bar{B})$$

[R.T.U. 2013, 2011, 2008, Raj. Univ. 2004]

Ans.(i) $\overline{AB} + ABC + A(B + \bar{A}\bar{B}) = (\overline{AB} + ABC) \cdot (A(B + \bar{A}\bar{B}))$

$$= (\overline{AB} + ABC) \cdot (A(B + \bar{A}\bar{B}))$$

Let $K = \overline{AB} + ABC$

$$= K \cdot (A(B + \bar{A}\bar{B})) = K \cdot (\bar{A} + \bar{B} + \bar{B}\bar{A})$$

$$= K \cdot (\bar{A} + 0 + \bar{A}\bar{B})$$

$$= K \cdot (\bar{A} + \bar{B}) = K \cdot \bar{A} = (\overline{AB} + ABC) \cdot \bar{A} \quad (\because \bar{A}\bar{A} = 0)$$

$$= \bar{A}\bar{B} + \bar{A}ABC = 0$$

PREVIOUS YEARS QUESTIONS

PART-A

Q.1 Convert $(D2.9)_{16} = (?)_8$ [R.T.U. 2018]

Ans. Convert $(D2.9)_{16}$ into four bit binary number and then making the groups of three bits.

Number $\rightarrow D2.9$

Four Bit $\rightarrow 011010010.100100$

322.44

Thus, the octal equivalent of hexadecimal number $(D2.9)_{16}$ is $(322.44)_8$.

Q.2 Find the sum $(y)_6 = (5.2)_6 + (4.5)_6$ [R.T.U. 2018]

Ans. $(y)_6 = (5 \times 6^0 + 2 \times 6^{-1}) + (4 \times 6^0 + 5 \times 6^{-1})$

$$= \left(5 + \frac{1}{3}\right) + \left(4 + \frac{5}{6}\right)$$

$$= \frac{61}{6}$$

$$= 10.1$$

Ans.

Q.3 Find the value of x

$$(23)_x + (12)_x = (101)_x$$

[R.T.U. 2018]

PART-B

Q.10 The two numbers represented in signed 2's complement form are $P = 11101101$ and $Q = 11100110$. Find the value obtained in signed 2's complement form for $P - Q$. [R.T.U. 2018, 2015]

Ans. Value obtained in signed 2's complement form for $P - Q$

Given that

$$P = 11101101$$

$$Q = 11100110$$

MSB of both numbers are 1, thus both are negative numbers. Now, we get

$$P = 11101101 = (-19)_{10}$$

$$\text{and } Q = 11100110 = (-26)_{10}$$

$$\therefore P - Q = (-19) - (-26) \\ = 7$$

Thus 7 signed 2's complement form is $(7) = 00000111$

Q.11 (a) What do you mean by sequential code, self complementing code, cyclic code and excess - 3 code? Give one example of each code.

(b) Using the postulates of Boolean algebra and the theorems, prove the following:

$$(i) a'b' + ab + a'b = a' + b$$

$$(ii) ab' + b'c' + a'c' = ab' + a'c' \quad [R.T.U. Dec. 2016]$$

Ans.(a) Sequential code : In case of seq. code, each succeeding code is one binary no. greater than the preceding code.

Ex - (i) 8421 code

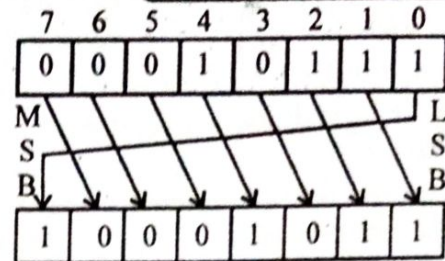
(ii) Excess - 3 code

Self complementing code : It is also as reflective code.

Ex - (i) 2421 code

(ii) Excess-3 code

Cyclic Code: A cyclic code is a block code, where the circular shifts of each code word gives another word that belongs to the code. They are error correcting codes that have algebraic properties that are convenient for efficient error detecting and correction.



Excess - 3 code :

Decimal No.	BCD	Excess - 3
0	0 0 0 0	0 0 1 1
1	0 0 0 1	0 1 0 1
2	0 0 1 0	0 1 0 1
3	0 0 1 1	0 1 1 0

Examples: (i) Unweighted code
(ii) 4-bit code

$$\text{Ans.(b)} \quad \overline{A}\overline{B} + AB + \overline{A}B = \overline{A}(\overline{B} + B) + AB$$

$$= \overline{A} + AB$$

$$\{B + \overline{B} = 1\}$$

$$= \overline{A} + AB$$

$$= (\overline{A} + A)(\overline{A} + B)$$

$$= 1(\overline{A} + B)$$

$$= \overline{A} + B$$

Hence proved

$$(ii) \quad \overline{A}\overline{B} + \overline{B}\overline{C} + \overline{A}\overline{C} = \overline{A}\overline{B} + \overline{B}\overline{C}(A + \overline{A}) + \overline{A}\overline{C}$$

$$= \overline{A}\overline{B} + \overline{A}\overline{B}\overline{C} + \overline{A}\overline{B}\overline{C} + \overline{A}\overline{C}$$

$$= \overline{A}\overline{C}(1 + \overline{B}) + \overline{A}\overline{B}(1 + \overline{C})$$

$$= \overline{A}\overline{C} + \overline{A}\overline{B}$$

Hence proved

Q.12 (a) For the integer with decimal representation 34567, give the corresponding bit vectors for BCD code and for excess - 3 code.

(b) Design a network using only XOR gates which performs the following function.

$$Z = \begin{cases} x_i & \text{if } C = 0 \\ \overline{x_i} & \text{if } C = 1 \end{cases}$$

[R.T.U. Dec. 2016]

Ans.(a) Represent the decimal no. 34567 in BCD code as

$$34567 \rightarrow \underbrace{0011}_3 \underbrace{0100}_4 \underbrace{0101}_5 \underbrace{0110}_6 \underbrace{0111}_7$$

Then the binary number is converted into octal number as follows:

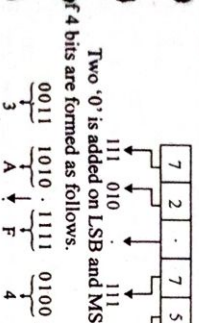
The binary bits are formed into groups of three and each group is converted into equivalent octal number

$$\begin{array}{r} 011 \ 111 \ 001 \ 001 \ 011 \ 000 \\ 3 \ 7 \ 1 \ 1 \ 3 \ 0 \end{array} \text{ Binary} \\ \text{Octal}$$

$$\text{Therefore, } (79.2C)_{16} = (371.130)_8$$

ii) $(72.75)_{10}$ into Hexadecimal

The given octal number is converted into binary as follows:



Two '0' are added on LSB and MSB sides, then groups of 4 bits are formed as follows:

$$\begin{array}{r} 0011 \ 1010 \ 1111 \ 0100 \\ 3 \ A \ F \ 4 \end{array}$$

$$\text{Therefore, } (72.75)_{10} = (3A.F4)_{16}$$

Ans.(c) Decimal equivalent of Excess-3 code (11001010.0101)

Binary number	1100	1010	0101
Its Decimal equivalent	12	10	5
Subtract 3 from each bit	-3	-3	-3
	9	7	2

Therefore, decimal equivalent of Excess-3 code (11001010.0101) is (97.2).

$$(11001010.0101)_2 \text{ is } (97.2)_{10}$$

1.14 (a) Subtract the following using 9's complement method

$$(i) \ 649 - 387$$

$$(ii) \ 891 - 786$$

(b) Minimize the following Boolean expression using Karnaugh maps of Boolean algebra:

$$(i) \ Y = AB + \overline{AC} + \overline{ABC} + C$$

$$(ii) \ Y = (AB + \overline{C}) + (\overline{A} + B + C)$$

Ans.(a) (i) Let $M = 649$

$$N = 387$$

9's complement of $N = 612$ [Subtract each digit of N from 9]

Adding this to M we get,

$$\begin{array}{r} 649 \\ + 612 \\ \hline 1261 \\ \text{end-around} \rightarrow +1 \\ \hline 1262 \\ \text{carry} \end{array}$$

$$= 262$$

Ans.

$$(ii) \text{ Let } M = 891$$

$$N = 786$$

9's complement of $N = 213$ [Subtract each digit of N from 9]

Adding this to M , we get,

$$\begin{array}{r} 891 \\ + 213 \\ \hline 1104 \\ \text{end-around} \rightarrow +1 \\ \hline 1105 \\ \text{carry} \end{array}$$

$$= 105$$

Ans.(b) (i) We have,

$$Y = AB + \overline{AC} + \overline{ABC} + C$$

$$= AB + \overline{AC} + 0 + \overline{ABC} \quad [\because B \cdot \overline{B} = 0 \text{ and } C \cdot C = C]$$

$$= AB + \overline{A} + \overline{C} + \overline{ABC}$$

$$= \overline{A} + B + \overline{C} + \overline{ABC}$$

$$= \overline{A} + \overline{ABC} + B + \overline{C}$$

$$= \overline{A} + \overline{B} + B + \overline{C}$$

$$= \overline{A} + C + B + \overline{C}$$

$$= \overline{A} + B + C + \overline{C}$$

$$= \overline{A} + B + 1$$

$$= 1$$

$$[\because C + \overline{C} = 1]$$

$$[\because 1 + x = 1]$$

$$(ii) \text{ We have,}$$

$$Y = (AB + \overline{C}) + (\overline{A} + B + C)$$

$$= (AB + \overline{C}) \cdot \{ \overline{A} + B + C \}$$

$$= (\overline{AB}) \cdot C \cdot (A + B) \cdot \overline{C}$$

$$= 0 \quad [\text{Since } \overline{C} \cdot C = 0]$$

$$[\text{Using De-Morgan's Theorem}]$$

$$\overline{AB} = \overline{A} + \overline{B}$$

$$A + B = \overline{\overline{A} + \overline{B}}$$

$$Q.15 (a) \text{ What is the importance of Gray code in engineering? Find equivalent binary Gray code of } (478)_{10}$$

(b) Convert the following

$$(i) \ (AB6)_{16} \text{ to Decimal}$$

$$(ii) \ (543.26)_{10} \text{ into Octal}$$

$$(iii) \ (247.36)_{10} \text{ into Hexa Decimal}$$

$$(iv) \ (AF9, BOD)_{16} \text{ into Binary.}$$

Ans. (a) Advantages of Gray Code

(1) This code is used in the shift position encoders.

(2) This code is useful for I/O devices, A/D converter and other peripheral devices.

(3) Very easy conversion process between binary and gray.

(4) This code is useful when other code produce undefined or ambiguous result during the transition from one number to next.

Gray Code of $(478)_{10}$: Gray code number differs from any adjacent number by a single bit.

To convert $(478)_{10}$ into Gray code, we will first find binary equivalent of $(478)_{10}$ i.e.

$$(478)_{10} = (1111011110)_2$$

Binary code to gray conversion is as follows:

Step-1: The first bit MSB of the gray code is the same as the first bit of the binary number.

$$\begin{array}{r} 1111011110 \\ \downarrow \\ 1110111110 \end{array} \text{ Binary}$$

Step-2: Ex-OR of 1^{st} and 2^{nd} bit yields 2^{nd} bit of gray code.

$$\begin{array}{r} 1110111110 \\ \oplus \\ 101111110 \end{array} \text{ Binary}$$

Step-3: Ex-OR of 2^{nd} and 3^{rd} bit yields 3^{rd} bit of gray code

$$\begin{array}{r} 101111110 \\ \oplus \\ 10111110 \end{array} \text{ Binary}$$

Step-4: Ex-OR of 3^{rd} and 4^{th} bit gives 4^{th} bit of gray code

$$\begin{array}{r} 111011110 \\ \oplus \\ 111011110 \end{array} \text{ Binary}$$

Step-5: Following the similar steps till the end of binary number gives us equivalent gray code i.e.

$$\begin{array}{r} 1001 \\ \oplus \\ 100110001 \end{array} \text{ Gray}$$

Ans. (b)(i) $(AB6)_{16} = A \times 16^2 + B \times 16^1 + 6 \times 16^0$

$$= 10 \times 16^2 + 11 \times 16 + 6 \times 1$$

$$= 2560 + 176 + 6$$

$$= (2742)_{10}$$

(ii) Integer Conversion:

$$\begin{array}{r} 8 \ 543 \\ 8 \ 67 \ 7 \\ 8 \ 8 \ 3 \\ 8 \ 1 \ 0 \\ 0 \ 1 \end{array}$$

Reading the remainders from bottom to top, the decimal number $(543)_{10}$ is equivalent to octal $(1037)_8$.

Multiplication

$$0.26 \times 8 = 2.08 \rightarrow 2$$

$$0.08 \times 8 = 0.64 \rightarrow 0$$

$$0.64 \times 8 = 5.12 \rightarrow 5$$

$$0.12 \times 8 = 0.96 \rightarrow 0$$

$$0.96 \times 8 = 7.68 \rightarrow 7$$

$$0.68 \times 8 = 5.44 \rightarrow 5$$

Clearly, reading from top to bottom we have,

$$(0.26)_{10} = (0.205075)_8$$

Thus, octal equivalent of $(543.26)_{10}$ is $(1037.205075)_8$

To represent $(247.36)_{10}$ into hexadecimal, we will first convert octal into decimal equivalent.

$$(247.36)_8 = 2 \times 8^2 + 4 \times 8^1 + 7 \times 8^0 + 3 \times 8^{-1} + 6 \times 8^{-2}$$

$$= 128 + 32 + 7 + 0.375 + 0.09375$$

$$= (167.46875)_{10}$$

Decimal to Hexadecimal

Integer conversion:

$$\begin{array}{r} 16 \ 167 \\ 16 \ 10 \ 7 \\ 0 \ 10(A) \end{array}$$

Reading from bottom to top gives the required equivalent hexadecimal number is $(A7)_{16}$.

Fractional Conversion:

Multiplication

$$0.46875 \times 16 = 7.5 \rightarrow 7$$

$$0.5 \times 16 = 8.0 \rightarrow 8$$

$$\text{Clearly, } (0.46875)_{10} = (.78)_{16}$$

Thus, hexadecimal equivalent of

$$(247.36)_8 = (A7.78)_{16}$$

Hexadecimal to binary conversion is done by just replacing each hexadecimal bit by its 4-bit binary equivalent.

$$(AF9.BOD)_{16} = A \ F \ 9 \ . \ B \ 0 \ D$$

The binary equivalent of given number $(AF9.BOD)_{16}$ is

$$(101011111001.101100001101)_2$$

Q16 Explain Duality theorem and De-Morgan's law.

Ans.

[R.T.U. 2013]

Ans. Duality Theorem : When one expression can be obtained from the other in each pair by replacing every 0 with 1, every 1 with 0, every (+) with (·) and every (·) with (+). Any pair of expression satisfying this property is called dual expression. This characteristic is called as principle of duality. Like

$$A + 0 = A$$

$$A \cdot 1 = A$$

$$A \cdot 0 = 0$$

$$A + A = A$$

$$A \cdot A = A$$

It states that every algebraic expression deducible from the postulates of boolean algebra remains valid if the operators and identity elements are interchanged. In the two-valued boolean algebra, the identity elements and the element of the set B are same : 1 and 0. The dual principle has many applications. If the dual of an algebraic expression is desired, we simply interchange OR and NAND operators and replace 1's by 0's and 0's by 1's.

De-Morgan's Theorems : Two theorems of Boolean algebra were proposed by De-Morgan.

(1) The first theorem states that the complement of a product is equal to the sum of the complements. That is, if the variables are A and B, then

$$\overline{A \cdot B} = \bar{A} + \bar{B}$$

(2) The second theorem states that, the complement of a sum is equal to the product of the complements. In equation form, this can be

$$\overline{A + B} = \bar{A} \cdot \bar{B}$$

In other words

(i) Replace the symbol (+) with symbol (·), the symbol (·) with (+) symbol.

(ii) Complement of each terms or variables.

Proof for the De-Morgan's theorem by truth table

A	B	$\bar{A}$	$\bar{B}$	$A + B$	$\overline{A + B}$	$\bar{A} \cdot \bar{B}$	$\overline{\bar{A} \cdot \bar{B}}$	$\bar{A} + \bar{B}$
0	0	1	1	0	1	1	1	1
0	1	1	0	1	0	0	1	1
1	0	0	1	1	0	0	1	1
1	1	0	0	1	0	0	0	0

Proof by algebraic method :

$$\overline{A \cdot B} = \bar{A} + \bar{B} \quad \dots(1)$$

Add AB both sides in (1)

$$A \cdot B + \overline{A \cdot B} = A \cdot B + \bar{A} + \bar{B}$$

$$A \cdot B + \bar{A} + \bar{B} = 1$$

[Absorption Law]

$$\bar{A} + B + \bar{B} = 1$$

$$\bar{A} + 1 = 1$$

$$1 = 1$$

Multiply AB to both sides in (1)

$$A \cdot B (\bar{A} + \bar{B}) = 0$$

$$A \cdot B \bar{A} + A \cdot B \bar{B} = 0$$

$$0 + 0 = 0$$

$$0 = 0$$

De-Morgan's first theorem is proved algebraically.

$$\overline{A + B} = \bar{A} \cdot \bar{B} \quad \dots(2)$$

Add A + B to both sides of (2)

$$A + B + \bar{A} \cdot \bar{B} = 1$$

$$A + B + \bar{B} = 1$$

$$A + 1 = 1$$

$$1 = 1$$

Multiply (A + B) to both sides of (2)

$$(A + B)(\bar{A} \cdot \bar{B}) = 0$$

$$\bar{A} \bar{B} A + B \bar{A} \bar{B} = 0$$

$$0 + 0 = 0$$

$$0 = 0$$

De-Morgan's second theorem is proved algebraically.

Q.17 Implement the XOR gate using minimal number of NAND gates. Show that the circuit drawn realizes the XOR gate. [R.T.U. 2009, Raj. Univ. 2005]

OR

Draw all logic gates with their truth tables. Generate X-OR gate with any one universal gate.

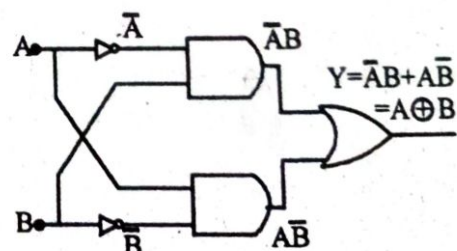
[R.T.U. 2013]

Ans. An exclusive OR gate is a gate with two or more inputs and one output. The output of a two input Ex-OR gate assumes a HIGH state. This is equivalent to saying that the output is HIGH, if either input A or input B is HIGH exclusively and low when both are 1 or 0 simultaneously.

The logic symbol for the Ex-OR gate is shown in fig. 1(a) and the truth table for the Ex-OR operation is given in table.



(a) Logic symbol



(b) Using AND-OR-NOT gates

Fig. 1 : Ex-OR gate

Table : Truth table

Input		Output
A	B	$Y = A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

DE.11

self complementing or reflective code. The reflective or self complementing property means a binary code if complemented, we get an opposite mirror reflection of that code.

Ans.(b) The ASCII code : The American Standard Code for Information Interchange (ASCII) pronounced as 'ASKEE' is a widely used alphanumeric code. This is basically a 7-bit code. Since the number of different bit patterns that can be created with 7 bit is $2^7 = 128$. The ASCII can be used to encode both the lowercase and uppercase characters of the alphabet (52 symbols) and some special symbols as well, in addition to the 10 decimal digits. It is used extensively for printers and terminals that interface with small computer systems. Many large systems also make provisions for its accommodation. Because characters are assigned in ascending binary numbers, ASCII is very easy for a computer to alphabetize and sort.

Table Shows the ASCII code groups. For case of presentation, they are listed with the three most significant bits of each group along the top row and the four least significant bits along the left column.

Table : The ASCII code

				MSBs							
				000	001	010	011	100	101	110	111
LSBs	0000	NUL	DEL	Space	0	@	P	>	p		
	0001	SOH	DC1	!	1	A	Q	a	q		
	0010	STX	DC2	u	2	B	R	b	r		
	0011	ETX	DC3	#	3	C	S	c	s		
	0100	EOT	DC4	\$	4	D	T	d	t		
	0101	END	NAK	%	5	E	U	e	u		
	0110	ACK	SYN	&	6	F	V	f	v		
	0111	BEL	ETB	'	7	G	W	g	w		
	1000	BS	CAN	(	8	H	X	h	x		
	1001	HT	EM	)	9	I	Y	i	y		
	1010	LF	SUB	*	:	J	Z	j	z		
	1011	VT	ESC	+	:	K	[	k	{		
	1100	FF	FS	,	<	L	\	l			
	1101	CR	GS	-	=	M	]	m	}		
	1110	SO	RS	.	>	N	^	n	~		
	1111	SI	US	/	?	O	_	o	DLE		

Q.19 (a) Consider the signed binary number are $A = 01000110$ and $B = 11010011$ where B is in 2's complement form. Find the value of following Mathematical expression.

(i) $A + B$

(ii) $A - B$

(iii) $B - A$

(iv) $-A - B$

(b) Why the gray code is also known as reflected code? Write a brief note on gray code and its applications.

[R.T.U. 2017]

DE.12

Ans.(a)

$A = 01000110$

$B = 11010011$

where B is in 2's complement form

(i) $A + B$

$$\begin{array}{r} 0100 \ 0110 \\ 1101 \ 0011 \\ \hline 0001 \ 1001 \end{array}$$

Since B is in 2's complement form therefore we discard the carry therefore

$A + B = 00011001$

Ans.

(ii) $A - B = A + (-B)$

$= A + 2's \text{ complement of } B$

$= 01000110 + 00101101$

$= 01110011$

Since there is no carry generated at MSB. Hence, we take the 2's complement of sum with -ve sign.

$(A - B) = (1000 \ 1101)$

Ans.

(iii) $B - A = B + (2's \text{ complement of } A)$

$= 11010011 + 10111010$

$= 10001101$

and carry discard since it is 2's complement.

$\text{Therefore } B - A = 10001101$

Ans.

(iv) $-A - B = (-A) + (-B)$

$= (2's \text{ complement of } A) + (2's \text{ complement of } B)$

$= 111.00111$

Ans.(b) Reflective Codes (or) Self Complementing Codes : Refer to Q.4.

Example : In 6423, the decimal 5 is represented by = 1011

$9 - N = 9 - 5 = 4$ decimal 4 is represented by = 0100
i.e. 9's complement of 5 is obtained just by complementing each bit in 5.

Necessary condition : A weighted code is said to be self complementing code if the sum of all weights is equal to '9'.

Totally there are 13 possible self complementing weighted codes which include positive and negative weights.

Example : 5211, 4221, 3321, 8421, etc.

Advantages of Gray Code : Refer to Q.15.

Applications of Gray Code

Since in the gray code only one bit changes between two successive numbers, therefore, it is also called the unit distance code. The unit distance code can be used in a shift encoder in which the position of a shift i.e. an analog quantity is represented digitally. The positional encoder wheel is attached to the shaft and the angular position of the shaft is represented with only four bit binary digits. The two angular position encoders are shown in fig. In the first, the binary coding is used and in other the gray coding is used. Since four

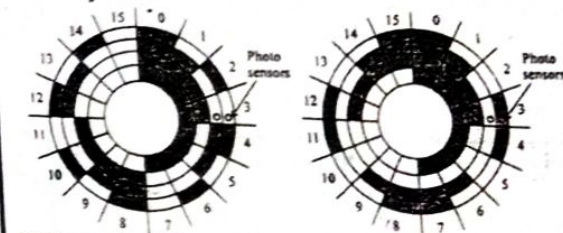
B.Tech. (III Sem.) C.S. Solved Papers

bits are used, so the angular position is given increments of

$$\frac{360}{2^4} = \frac{360}{16} = 22.5^\circ$$

Thus, the whole encoder is divided into $2^4 = 16$ sectors. Each sector shows a decimal digit from 0 to 15 in binary and gray codes. To determine the angular position i.e. sector, the fixed photo sensors are arranged in a line.

It can be used to detect the light and shaded or dark cells across the sector of the encoder wheel. The light cell represents the binary bit 1 and shaded or dark cell represents the binary bit 0.



(a) Conventional binary encoder (b) Gray code encoder

Fig. : Angular Position Encoders

When the position binary encoder moves or rotates from sector 3 in fig. (a), the code word must change from 0 0 1 1 to 0 1 0 0. Here, the code word must change from 0 0 1 0 to 0 1 1 0 where only second bit is changing i.e. a single bit is changing in the code word.

Q.20 (a) Convert the following-

(i) $(2F \ 9A)_{16} = ()_2$

(ii) $(1100100)_2 = ()_8$

(iii) $(247.05)_{10} = ()_2$

(iv) $(1010.011)_2 = ()_{10}$

(b) Represent decimal no (190)₁₀ in following

(i) Excess - 3 code

(ii) Gray code

(iii) BCD code

(c) Perform the following operations-

(i) $(11.1001)_2 - (01.110)_2$ using 2's complement

(ii) $(2AC)_{16} + (10F)_{16}$

(iii) $(32)_8 + (67)_8$

[R.T.U. Jan. 2016]

Ans.(a)(i) $(2F \ 9A)_{16}$

in binary

$2 \rightarrow 0010$

$F \rightarrow 1111$

$9 \rightarrow 1001$

$A \rightarrow 1010$

Hence

$(2F \ 9A)_{16} = (0010111110011010)_2$

Ans.

(ii) $(1100100)_2$

$= (001100100)_2$

$= (144)_8$

Ans.

Q.21 (a) Define the following with one example-

(i) Error correcting codes

(ii) Gray code

(iii) Hamming distance

(iv) Even and odd parity code

(b) Consider following codes-

Code A	Code B	Code C	Code D
0001	000	01011	000000
0010	001	01100	001111
0100	011	10010	110011
1000	010	10101	
	110		
	111		
	101		
	100		

Deform which of the following properties is satisfied by each above codes.

(i) Detects single error

(ii) Detects double error

(iii) Corrects single error

(iv) Corrects single error and detects double error.

[R.T.U. Jan. 2016]

Ans.(a)(i) Error Correcting Codes : By adding a single parity bit along with the information, or message being

transmitted, an error in single bit position can be detected. The parity check gives only information that the received message is incorrect. It can not locate the bit position in which error has occurred and, therefore, can not correct the error.

Let us consider a 4-bit binary word 0101 is transmitted along with an even parity bit. Due to transmission error in one bit position, the erroneous word received may be 00100, 00111, 00001, or 01101 depending on the error in bit position b_0 , b_1 , b_2 and b_3 , respectively.

Let us examine whether these erroneous words are possible with any other message being transmitted. If the word 01100 is transmitted, it results in 00100 at the receiving end due to an error in bit position b_3 . Similarly, the other erroneous words are received due to the transmission error in some other words being transmitted. This indicates that a minimum distance of two can not locate the bit position in the incorrectly received word. Therefore, for a code to be error correcting, its minimum distance must be more than two.

If the minimum distance is three, every error in single bit results in an invalid code word which is at a distance of one from the original code word and at a distance of two from any other valid code word. Therefore, a single error can be detected and located using this code. Once the error bit is located it can be inverted to correct the erroneously received message.

In general, a code is said to be error-correcting code if the correct code word can be deduced from the erroneous word. The efficiency of a code to be error detecting and/or error correcting can be determined from its minimum distance which is $2c + d + 1$. It can correct errors in upto c bits and detect errors in upto d additional bits.

(ii) Gray Code : Refer to Q.15.

(iii) Hamming Distance : It is the number of bits changed from one code word [data + parity bits] to another code word. It is defined by d_{min} .

Let 4 bit data is $D_4 D_3 D_2 D_1$.

And 3 parity bits are added $P_1 P_2 P_3$.

Then Hamming code will be

Sequence $\rightarrow$ 1 2 3 4 5 6 7

Arrangement $\rightarrow$ $P_1 P_2 D_4 P_3 D_3 D_2 D_1$

where $P_1 = 1$ or 0 so that position 1, 3, 5, 7 has even parity.

$P_2 = 1$ or 0 so that position 2, 3, 6, 7 has even parity.

$P_3 = 1$ or 0 so that position 4, 5, 6, 7 has even parity.

For example data is (1110) then

1 2 3 4 5 6 7
 $P_1 P_2 1 P_3 1 1 0$

where $P_1 =$ even parity of ($P_2, 1, 1, 0$) so $P_1 = 0$

$P_2 =$ even parity of ($P_1, 1, 1, 0$) so $P_2 = 0$

$P_3 =$ even parity of ($P_1, 1, 1, 0$) so $P_3 = 0$

so code word will be (0010110)

At the receiver code word is checked.

Step 1 : Position 1, 3, 5, 7 is checked for even parity, for even parity $P_1 = 0$ else $P_1 = 1$.

Step 2 : Position 2, 3, 6, 7 is checked for even parity, for even parity $P_2 = 0$ else $P_2 = 1$.

Step 3 : Position 4, 5, 6, 7 is checked for even parity, for even parity $P_3 = 0$ else $P_3 = 1$.

Step 4 : Find P_1, P_2, P_3 by writing 0 or 1. There will be error at position ($P_3 P_2 P_1$).

So complement that bit if ($P_3 P_2 P_1$) $\Rightarrow$ 000 then no error found.

(iv) Parity Code : The parity code is an error-checking code in which binary numbers are transmitted with an additional bit that is used only for error detection. In this code all data are transmitted either with an odd number of 1's (odd parity) or an even number of 1's (even parity). The odd-parity code is more commonly used, since it is not possible to transmit a number or word with all 0's. Table given below shows a display of data and the generation of a parity bit. Both odd and even parity codes are shown. Note that a parity bit is generated according to the number of 1's in a number. In the odd-parity code, a 0 parity bit is generated when the number of 1's in a number is odd. A 1 parity bit is generated when the number of 1's is even. The even-parity code is just the opposite of the odd code.

Table 1 : Odd parity and even parity codes table

Odd Parity	
Data	Parity bit
001001	1
110100	0
1011011	0
101010	0
110011	1

Table 2

Even parity	
Data	Parity bit
001001	0
1110110	1
101111	1
100110	1
100111	0

Ans.(b) We know that to detect single bit errors, minimum distance should be equal to 2, to detect double bit errors or detect single error and correct single error, minimum distance should be equal to 3, to detect triple bit errors or detect double error and correct single error, minimum distance should be equal to 4, and so on.

Considering Code A, it is observed that the minimum distance is 2, therefore it can detect single bit error.

Considering Code B, it is observed that the minimum distance is 1, therefore it can neither detect nor correct error bits.

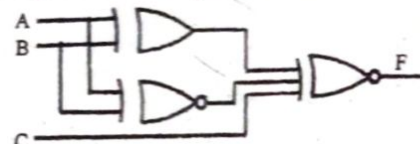
Considering Code C, it is observed that the minimum distance is 3, therefore it can detect double errors or detect single error and correct single error.

Considering Code D, it is observed that the minimum distance is 4, therefore it can detect triple bit errors or detect double error and correct single error.

Therefore:

- Detect single error: Code A, C and D.
- Detect double error: Code C and D.
- Correct single error: Code C and D.
- Correct single error and detect double error: Code D.

Q.22 (a) For the output 'F' to be '1' the logic circuit shown in figure find the values of A, B & C.

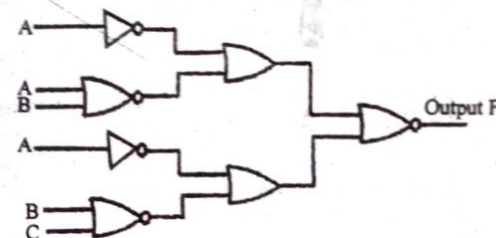


(b) Perform the following subtractions in 8421 code using the 10's complement method:

(i) $342.7 - 108.9$

(ii) $206.4 - 507.6$

(c) Draw the simplest possible logic diagram that implements the output of the logic diagram shown below:



[R.T.U. 2015]

Ans.(a)

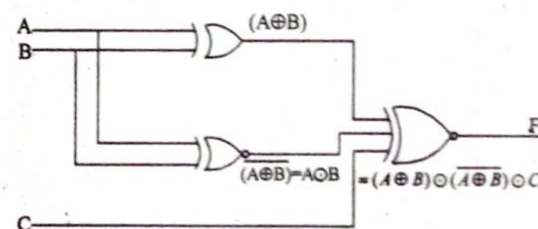


Fig.

PART-A

Q.1 Convert $Y = A + B\bar{C} + \bar{A}\bar{B} + A\bar{B}C$ into canonical form. [R.T.U. 2018]

Ans.

$$y = A + B\bar{C} + \bar{A}\bar{B} + A\bar{B}C$$

$$y = A(B + \bar{B})(C + \bar{C}) + (\bar{A} + A)\bar{B}C + A\bar{B}(C + \bar{C}) + A\bar{B}C$$

$$= A(BC + B\bar{C} + \bar{B}C + \bar{B}\bar{C}) + ABC + \bar{A}\bar{B}C + A\bar{B}C + A\bar{B}C$$

$$+ A\bar{B}C$$

$$= ABC + A\bar{B}C + A\bar{B}C + A\bar{B}C + A\bar{B}C + A\bar{B}C + A\bar{B}C$$

$$+ A\bar{B}C + A\bar{B}C$$

$$= ABC + A\bar{B}C + A\bar{B}C + A\bar{B}C + A\bar{B}C$$

$$y = \sum m(2, 4, 5, 6, 7)$$

Q.2 Find the minimum number of NAND gate required to realize $Y = A\bar{B}C$. [R.T.U. 2018]

Ans.

$$y = A\bar{B}C$$

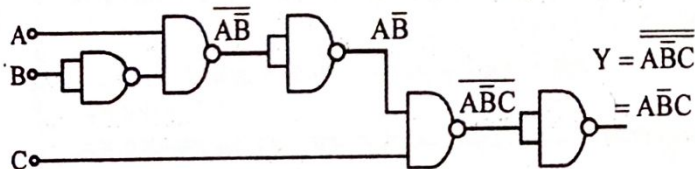


Fig.

Total of 5 gates are required to realize the equation.

Q.3 Simplify the following boolean function using K-map:

$$Y = (A + B)(A + \bar{C})(\bar{A} + \bar{B})(\bar{A} + C).$$

[R.T.U. 2017]

Ans. According to Boolean theorems

$$Y = (A + B + C\bar{C})(A + B\bar{B} + \bar{C})(\bar{A} + \bar{B} + C\bar{C})(\bar{A} + B\bar{B} + C)$$

$$Y = (A + B + C)(A + B + \bar{C})(A + B + \bar{C})(A + \bar{B} + C)$$

$$(\bar{A} + \bar{B} + C)(\bar{A} + \bar{B} + \bar{C})(\bar{A} + B + C)(\bar{A} + \bar{B} + C)$$

After removing common terms

$$\Rightarrow (A + B + C)(A + B + \bar{C})(A + \bar{B} + C)(\bar{A} + \bar{B} + C)$$

$$\Rightarrow (\bar{A} + \bar{B} + C)(\bar{A} + B + C)$$

$$\Rightarrow \Pi(0, 1, 3, 6, 7, 4)$$

$$\Rightarrow \Pi(0, 1, 3, 4, 6, 7)$$

	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
$\bar{A}$	0	1	0	1
A	0	1	0	0

$$Y = A\bar{B}C + \bar{A}\bar{B}C$$

Q.4 The logic gate G_1 and G_2 as shown in Fig. have propagation delay of 10 ns and 20 ns respectively. If input v_i makes an abrupt change from logic-0 to logic-1 at time $t = t_0$, then draw output waveform (V_o).

ns → nano second

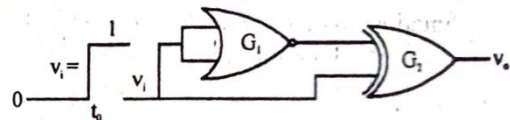


Fig.

[R.T.U. 2017]

Ans. Propagation delay of G_1 is 10 ns.

Propagation delay of G_2 is 20 ns.

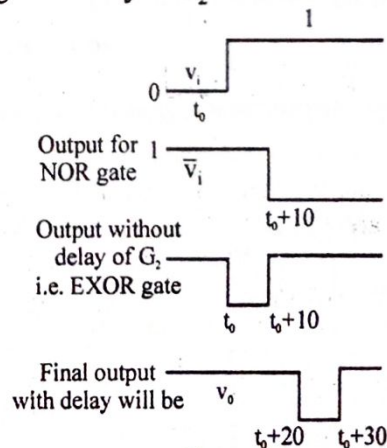


Fig.

Q.5 Minimize the following boolean function using K-map:

$$f(a, b, c, d) = \sum m(0, 1, 2, 8, 9). d(4, 10, 12)$$

[R.T.U. 2017]

DE.22

Ans. The given 4-variable function has minterms with two don't care terms. It can be minimized as follows :

cd \ ab	00	01	11	10
00	m ₀	m ₁	m ₃	m ₂
01	m ₄	m ₅	m ₇	m ₆
11	m ₁₂	m ₁₃	m ₁₅	m ₁₄
10	m ₈	m ₉	m ₁₁	m ₁₀

Now, according to given question

cd \ ab	cd	cd	cd	cd
(a b)00	1	1	0	1
(a b)01	d	0	0	0
(a b)11	d	0	0	0
(a b)10	1	1	0	d

Fig.

Here the don't terms are considered as 1 for simplification. If the don't care term helps to make a larger group, then it is included in the group, otherwise ignored. Above k-map has two quads (LOOP-1 and LOOP-2). So, the minimized boolean expression of given function is

$$f(a, b, c, d) = \bar{b}c + b\bar{d}$$

Q.6 Simplify the expression

$$F(A, B, C, D) = ACD + \bar{A}B + \bar{D}$$

[R.T.U. Dec. 2016]

Ans.

CD \ AB	00	01	11	10
00	1			1
01	1	1	1	1
11	1		1	1
10	1		1	1

$$F = \bar{A}B + AC + \bar{D}$$

Q.7 Reduce the expression using mapping and implement it in universal logic.

Im (2,8,9,10,11,12,14)

[R.T.U. 2013]

$$Ans. F(A, B, C, D) = \Pi m(2, 8, 9, 10, 11, 12, 14)$$

B.Tech. (III Sem.) C.S. Solved Papers

Enter 0 to the above mentioned cells and grouping adjacent cells to obtain the minimized expression.

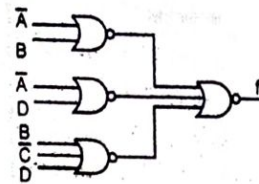
$$F(A, B, C, D) = (\bar{A} + B)(\bar{A} + D)(B + \bar{C} + D)$$

K-map :

CD \ AB	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	0	13	15	14
10	0	0	0	10

$$f = (\bar{A} + B)(\bar{A} + D)(B + \bar{C} + D)$$

Circuit diagram :



NOR logic Fig.

PART-B

Q.8 Simplify the following function and implement it using NOR gate :

$$F = \bar{A}\bar{B} + ABD + AB\bar{D} + \bar{A}\bar{C}\bar{D} + \bar{A}B\bar{C}$$

[R.T.U. 2018]

Ans.

$$F = \bar{A}\bar{B} + ABD + AB\bar{D} + \bar{A}\bar{C}\bar{D} + \bar{A}B\bar{C}$$

$$F = \bar{A}\bar{B}(C + \bar{C})(D + \bar{D}) + AB(C + \bar{C})D + AB(C + \bar{C})\bar{D} + \bar{A}(B + \bar{B})\bar{C}\bar{D} + \bar{A}B\bar{C}(D + \bar{D})$$

$$= \bar{A}\bar{B}CD + \bar{A}\bar{B}\bar{C}D + \bar{A}\bar{B}C\bar{D} + \bar{A}\bar{B}\bar{C}\bar{D} + ABCD + AB\bar{C}D + AB\bar{C}\bar{D} + \bar{A}B\bar{C}D + \bar{A}B\bar{C}\bar{D} + \bar{A}B\bar{C}D + \bar{A}B\bar{C}\bar{D}$$

The above function is for MIN terms, which results the output equation in SOP form, but here we are required to implement the function using NOR gate only, for that, converting the MIN term function into MAX term function.

$$\Sigma m(0, 4, 5, 8, 9, 10, 11, 12, 13, 15) = \Pi M(1, 2, 3, 6, 7, 14)$$

Now plot the function for MAX term using k-map and group them.

Digital Electronics

CD \ AB	00	01	11	10
00	0	0	0	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

The minimized equation in POS form is

$$F = (A + B + \bar{D})(A + \bar{C})(\bar{B} + \bar{C} + D)$$

Now, realization using NOR gate -

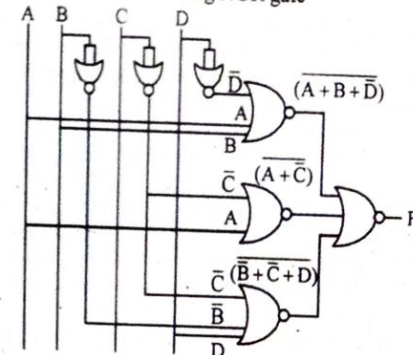


Fig. : NOR Realization

Q.9 Explain prime, essential and redundant implicant using suitable example.

[R.T.U. 2018]

Ans. First all the MIN terms are represented in the binary form

Table 1 : Binary representation of MIN terms - 1

MIN terms	8 A 2 ³	4 B 2 ²	2 C 2 ¹	1 D 2 ⁰
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
14	1	1	1	0
15	1	1	1	1

Now the minterms are arranged according to the increasing index.

Q.11 Simplify the following boolean function using tabulation method:

$$F = \sum m(0, 1, 2, 8, 10, 11, 14, 15). d(9, 12).$$

[R.T.U. 2017]

Ans. First, all the minterms are represented in the binary form as shown in table (1)

Table 1

Minterms	Variables			
	A	B	C	D
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
14	1	1	1	0
15	1	1	1	1

Now, the minterms are arranged according to the increasing index as shown in table (2).

All index minterms are separated by a horizontal line.

Table 2

Number of 1's (Index)	Minterms	Variables			
		A	B	C	D
0	0 ✓	0	0	0	0
1	1 ✓	0	0	0	1
	2 ✓	0	0	1	0
	8 ✓	1	0	0	0
2	9 ✓	1	0	0	1
	10 ✓	1	0	1	0
	12 ✓	1	1	0	0
3	11 ✓	1	0	1	1
	14 ✓	1	1	1	0
4	15 ✓	1	1	1	1

Each n^{th} index minterms are compared with each $(n+1)^{\text{th}}$ index minterms and two minterms which are differing from each other by one variable are combined to get the combinations of two minterms are shown in table (3). This variable is replaced by a dash.

Table 3 : Combination of two minterms

Combination	Variables			
	A	B	C	D
(0,1) ✓	0	0	0	-
(0,2) ✓	0	0	-	0

(0,8) ✓	-	0	0	0
(1,9) ✓	-	0	0	1
(2,10) ✓	-	0	1	0
(8,9) ✓	1	0	0	-
(8,10) ✓	1	0	-	0
(8,12) ✓	1	-	0	0
(9,11) ✓	1	0	-	1
(10,11) ✓	1	0	1	-
(10,14) ✓	1	-	1	0
(12,14) ✓	1	1	-	0
(11,15) ✓	1	-	1	1
(14,15) ✓	1	1	1	-

The minterms which are combined in the combination of two minterms are marked by (✓) sign as shown in table. Now the combination of two minterms are compared. If two combinations are varying by one variable and consist a dash on the same position they are combined to get the combinations of four minterms as shown in table (4).

Table 4 : Combination of four minterms

Combination	Variables			
	A	B	C	D
(0,1,8,9)	-	0	0	-
(0,2,8,10)	-	0	-	0
(0,8,1,9)	-	0	0	-
(0,8,2,10)	-	0	-	0
(8,9,10,11)	1	0	-	-
(8,10,12,14)	1	-	-	0
(8,12,10,14)	1	-	-	0
(9,11,8,10)	1	0	-	-
(10,11,14,15)	1	-	1	-
(10,14,11,15)	1	-	1	-

The combination of four minterms can be further combined to get the combination of eight minterms if the two combinations differ by one variable and two dashes are placed on the same position. Combination can be shown in table (5).

Table 5

Combination	Variables			
	A	B	C	D
(0,1,8,9)	-	0	0	-
(0,2,8,10)	-	0	-	0
(8,9,10,11)	1	0	-	-
(8,10,12,14)	1	-	-	0
(10,11,14,15)	1	-	1	-

All the unchecked terms also called prime implicants are shown in table (6). The crosses (X) are placed corresponding to the minterms. In the prime implicant table, don't care terms given in the function (9,12) are not written.

DE.26*Table 6 : Prime Implicant Table*

Prime Implicants	Minterms							
	0	1	2	8	10	11	14	15
(0,1,8,9)*	x	x		x				
(0,2,8,10)*	x		x	x	x			
(8,9,10,11)				x	x	x		
(8,10,12,14)				x	x		x	
(10,11,14,15)*					x	x	x	x
		✓	✓					✓

The tick mark (✓) is placed against each column which has a single cross (x). Now the (*) is placed on the prime implicants which corresponds to the minterms or the columns which have a single cross and are marked by a tick mark (✓). These star marked prime implicants are called essential prime implicants.

Since these essential prime implicants cover all the minterms given in the function, so the minimized boolean expression of the given function is

$$F = \overline{BC} + \overline{BD} + AC$$

Q.12 Compare k-map technique and quine-Mc-cluskey minimization technique.

[R.T.U. Dec. 2016]

Table : 3 Combine MIN Terms in group of 2

Combination	Variables			
	A	B	C	D
(0, 1)	0	0	0	—
(0, 8)	—	0	0	0
(1, 3)	0	0	—	1
(1, 9)	—	0	0	1
(8, 9)	1	0	0	—
(8, 10)	1	0	—	0
(9, 13)	1	—	0	1

Table : 4 Combine MIN Terms in group of four

Combination	Variables				Term
	A	B	C	D	
(0, 1, 8, 9)	—	0	0	—	$\overline{B}\overline{C}$
(0, 8, 1, 9)	—	0	0	—	

Since (0, 1, 8, 9) & (0, 8, 1, 9) are same, so only one combination will be taken into considerations.

Table 5 : Prime Implicant Table

PI	0 ✓	6	9 ✓	10	13
(0, 1, 8, 9)	x		x		
	✓		✓		

Thus, $F = \overline{B}\overline{C} + A\overline{B}\overline{D} + A\overline{C}D + \overline{A}BC\overline{D}$
Verifying the result using k-map.

$$F = \Sigma(0, 6, 9, 10, 13) + d(1, 3, 8)$$

AB \ CD	00	01	11	10
00	1 0	d 1	0 3	0 2
01	0 4	0 5	0 7	1 6
11	0 12	1 13	0 15	1 14
10	d 8	1 9	0 11	1 10

In grouping → 4 quads, 2 pairs, single term.

$$F = \overline{B}\overline{C} + A\overline{C}D + A\overline{B}\overline{D} + \overline{A}BC\overline{D}$$

Q.17 Write short note on Universal gate. [R.T.U. 2018]

Ans. Universal Gate : NAND and NOR gates are called universal gates or universal building blocks because both can be used to implement any gate like AND, OR and NOT gates or any combination of these basic gates.

(a) NAND Gate → NAND is a contraction of the NOT-AND gates. It has two or more inputs and only one output i.e. $y = \overline{A.B}$.

When all the inputs are HIGH, the output is LOW. If any one or both the inputs are LOW, then the output is HIGH. The

logic symbol for the NAND gate is as shown.

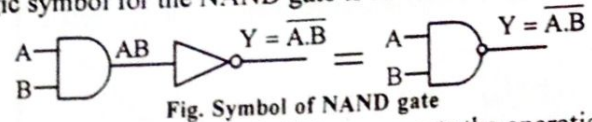


Fig. Symbol of NAND gate

The small circle or bubble represents the operation of inversion.

Truth table for NAND gate is as shown in table.

Inputs		Outputs
A	B	$y = A.B$
0	0	1
0	1	1
1	0	1
1	1	0

IC configuration of NAND gate is

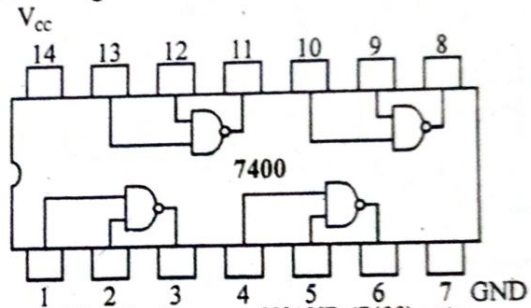


Fig. Pin diagram of NAND (7400) gate.

(b) NOR Gate → NOR is a contraction of NOT-OR gates.

It has 2 or more inputs and only one output, i.e. $y = \overline{A + B}$. The output is HIGH only when all the inputs are LOW. If any one or both the inputs are HIGH, then the output is LOW. The logic symbol is as shown

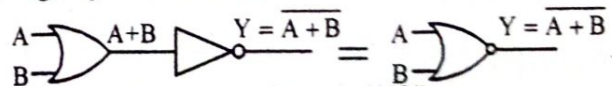


Fig. : Logic Symbol of NOR gate

The truth table of 2 input NOR gate is shown in table.

Inputs		Outputs
A	B	$y = A + B$
0	0	1
0	1	0
1	0	0
1	1	0

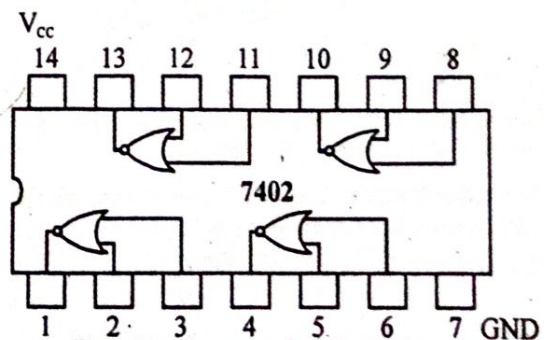


Fig. : Pin diagram of NOR (7402) gate

DE.30

NAND and NOR gate can be used to realize various logic gates.

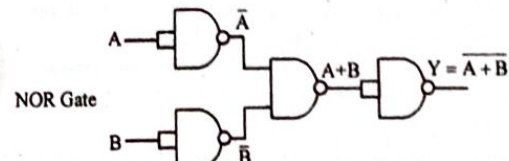
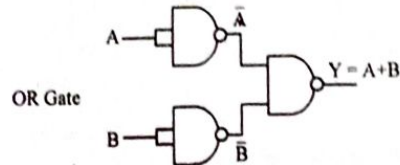
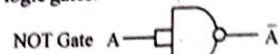


Fig. : Realization of NOT, AND, OR, NOR gates using NAND gates only.

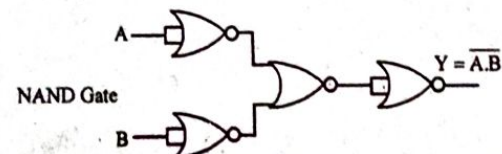
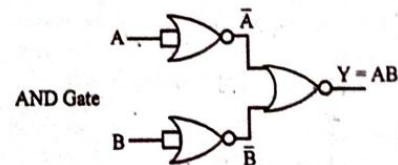
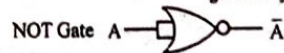


Fig. : Realization of NOT, OR, AND, NAND gates using NOR gates only.

Q.18 Simplify the following using the tabulation method:
 $F = \sum(1, 2, 3, 7, 8, 9, 10, 11, 14, 15)$

[R.T.U. Dec. 2016]

Ans. Firstly, these minterms are represented in the

Table (1) : Binary representation of minterms

Minterms	Variables			
	A	B	C	D
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
14	1	1	1	0
15	1	1	1	1

binary form as shown in Table (1). The above binary representation are grouped into a number of sections in terms of the number of 1s as shown in Table (2)

Table (2) : Group of minterms for different number of 1s

Number of 1s	Minterms	Variables			
		A	B	C	D
1	1 ✓	0	0	0	1
	2 ✓	0	0	1	0
	8 ✓	1	0	0	0
2	3 ✓	0	0	1	1
	9 ✓	1	0	0	1
	10 ✓	1	0	1	0
3	7 ✓	0	1	1	1
	11 ✓	1	0	1	1
	14 ✓	1	1	1	0
4	15 ✓	1	1	1	1

Any two numbers in these groups which differ from each other by only one variable can be chosen and combined, to get two-cell combinations, as shown in Table (3).

Table (3) : 2-Cell Combinations

Combination	Variables			
	A	B	C	D
(1, 3) ✓	0	0	-	1
(1, 9) ✓	-	0	0	1
(2, 3) ✓	0	0	1	-
(2, 10) ✓	-	0	1	0
(8, 9) ✓	1	0	0	-
(8, 10) ✓	1	0	-	0
(3, 7) ✓	0	-	1	1
(3, 11) ✓	-	0	1	1
(9, 11) ✓	1	0	-	1
(10, 11) ✓	1	0	1	-
(10, 14) ✓	1	-	1	0
(7, 15) ✓	-	1	1	1
(11, 15) ✓	1	-	1	1
(14, 15) ✓	1	1	1	-

From the two-cell combinations, one variable and a dash in the same position can be combined to form 4-cell combinations as shown in Table (4).

Table (4) : 4-cell combinations

Combination	A	B	C	D
(1, 3, 9, 11)	-	0	-	1
(2, 3, 10, 11)	-	0	1	-
(8, 9, 10, 11)	1	0	-	-
(3, 7, 11, 15)	-	-	1	1
(10, 11, 14, 15)	1	-	1	-

Note that the cells (1,3) and (9,11) form the same 4-cell combination as the cells (1,9) and (3,11). Thus, order in which the cells are placed in a combination does not have any effect. Thus, the (1,3,9,11) combination may be given as (1,9,3,11). Using table 1(d), the prime implicants table can be plotted as shown in Table (5).

The columns having only one cross mark correspond to essential prime implicants. A tick mark is put against every column which has only one cross mark. A star mark is placed against every essential primary implicant. The sum of the prime implicants gives the function in its minimal SOP form since all prime implicants are essential prime implicants.

$$\text{Therefore, } f = \bar{B}.D + \bar{B}.C + A.\bar{B} + C.D + A.C$$

Table (5) : Prime implicants table

Prime Implicants	Variables														
	1	2	3	7	8	9	10	11	14	15					
(1, 9, 3, 11)*	x		x			x		x							
(2, 3, 10, 11)*		x	x					x	x						
(8, 9, 10, 11)*					x	x	x	x							
(3, 7, 11, 15)*			x	x				x	x		x				
(10, 11, 14, 15)*							x	x	x	x					
	✓	✓		✓	✓					✓					

Q.19 (a) Minimize the logic expression with K-map

(i) $F(A, B, C, D) = \sum m(1, 2, 3, 7, 11, 15) + d(0, 4, 5)$

(ii) $F(AB, CD) = \prod m(1, 4, 6, 9, 10, 11) + d(13, 14, 15)$

(b) What is Qvine-McCluskey minimization technique? Explain for minimize the function.

$y(A, B, C, D) = \sum m(0, 1, 3, 4, 5, 7, 8, 9, 11, 15)$

[R.T.U. Jan. 2016]

n the binary
representation
the number

From the two-cell combinations, one variable and a dash in the same position can be combined to form 4-cell combinations as shown in Table 1 (d).

Table 1 (d) : 4-cell combinations

Combination	A	B	C	D
(1, 3, 9, 11)	—	0	—	1
(2, 3, 10, 11)	—	0	1	—
(8, 9, 10, 11)	1	0	—	—
(3, 7, 11, 15)	—	—	1	1
(10, 11, 14, 15)	1	—	1	—

Note that the cells (1,3) and (9,11) form the same 4-cell combination as the cells (1,9) and (3,11). Thus, order in which the cells are placed in a combination does not have any effect. Thus, the (1,3,9,11) combination may be given as (1,9,3,11). Using table 1(d), the prime implicants table can be plotted as shown in Table 1 (e).

The columns having only one cross mark correspond to essential prime implicants. A tick mark is put against every column which has only one cross mark. A star mark is placed against every essential primary implicant. The sum of the prime implicants gives the function in its minimal SOP form since all prime implicants are essential prime implicants.

$$\text{Therefore, } f = \bar{B}D + \bar{B}C + A\bar{B} + CD + A.C$$

Table 1 (e) : Prime implicants table

Prime implicants	Variables														
	1	2	3	7	8	9	10	11	14	15					
(1, 9, 3, 11)*	x		x			x		x							
(2, 3, 10, 11)*		x	x				x	x							
(8, 9, 10, 11)*					x	x	x	x							
(3, 7, 11, 15)*			x	x				x						x	
(10, 11, 14, 15)*							x	x		x	x				
	✓	✓		✓	✓					✓					

↑

Q.22 (a) Simplify using K-map.

$$f = \sum m(1, 3, 7, 11, 15) + d(0, 2, 5) \quad [\text{R.T.U. 2013}]$$

(b) Explain following with example.

(i) SOP and POS form

(ii) Minterm and Maxterm

[R.T.U. 2013]

OR

Name the two basic types of boolean expressions.

[R.T.U. 2010]

Ans. (a) Given that

$$f(A, B, C, D) = \sum m(1, 3, 7, 11, 15) + d(0, 2, 5)$$

Where m indicates the minterms representing and d indicates the don't care conditions.

Enter 1 to the cells 1,3,7,11,15

Enter x to the cells 0,2,5

$$F(A, B, C, D) = \bar{A}D + \bar{A}\bar{B} + CD$$

Note: you can group the don't care (x) with 1, but should not make grouping for only don't care. Grouping should always have at least one 1.

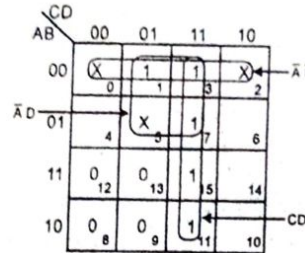


Fig.

Ans.(b) There are two basic types of boolean expression.

(i) **Sum of Products (SOP)** : The logical sum of two or more logical product terms, is called a sum of products expression. It is basically an OR operation of AND operated variables such as

$$(1) \quad Y = AB + BC + AC$$

$$(2) \quad Y = AB + \bar{A}C + BC$$

(ii) **Minterm** : A product term contains all the K-variable of the function in either complemented or uncomplemented form is called a minterm. A 2-variable function has four possible combination viz., $\bar{A}\bar{B}$, $\bar{A}B$, $A\bar{B}$ and AB . These product terms are called minterm or standard product or fundamental product. Each minterm can be obtained by the AND operations of all the variable of the function. In the minterm, a variable appears either in complemented form, if it possesses a value of 1 in the corresponding combination, or in complemented form, if it contain value 0. The main property of minterm is that it possesses a value 1 for only one combination of k input variable i.e. for a k-variable function of the 2^k minterm, only one minterm will have the value 1 while remaining $2^k - 1$ minterm will possess the value 0 for as arbitrary input combination. For e.g. as shown in table 1 for input combination 0/0 i.e. for $A = 0, B = 1$ and $C = 0$. Only the minterms $\bar{A}\bar{B}C$ will have the value 1, while the remaining seven minterm will have the value 0.

Table 1

A	B	C	Minterm
0	0	0	$\bar{A}\bar{B}\bar{C}$
0	0	1	$\bar{A}\bar{B}C$
0	1	0	$\bar{A}B\bar{C}$
0	1	1	$\bar{A}BC$
1	0	0	$A\bar{B}\bar{C}$
1	0	1	$A\bar{B}C$
1	1	0	$AB\bar{C}$
1	1	1	ABC

Canonical Sum of Product expression : It is defined as the logical sum of all the minterm device from the rows of truth table.

For which the value of the function is 1. It is also called a minterm canonical form. The canonical sum of product expression can be given in a compact form listing the decimal codes in correspondence will the minterm containing a function value of 1.

Product of Sum : The logical product of two or more logical sum terms, is called a product of sum expression. It is basically an AND operation of OR operated variable such as $y = (A + B).(B + C).(C + D)$

Maxterm : A sum term containing all the k-variable of the function in either complemented or uncomplemented form is called maxterm. A 2-variable function has four possible combinations viz.

$$A + B, A + \bar{B}, \bar{A} + B \text{ and } \bar{A} + \bar{B}$$

These sum are called maxterm. Each maxterm is obtained by the OR operation of all the variable of the function. In a maxterm, a variable appears either in uncomplemented or the complemented form it contain value 1.

The most important property of maxterm is that it possesses the value 0 for only one combination k-input variable i.e. for k variable function of the 2^k maxterms, only one maxterm will have the value 0, while all the remaining $2^k - 1$ maxterm will have the value, for an arbitrary input combination.

E.g., for input combination 101, i.e. for $A = 1, B = 0$ and $C = 1$. Only the maxterms $(\bar{A} + \bar{B} + \bar{C})$ will have the value 0, while the remaining seven maxterm will have the value 1.

Table 2

A	B	C	Maxterm
0	0	0	$A + B + C$
0	0	1	$A + B + \bar{C}$
0	1	0	$A + \bar{B} + C$
0	1	1	$A + \bar{B} + \bar{C}$
1	0	0	$\bar{A} + B + C$
1	0	1	$\bar{A} + B + \bar{C}$
1	1	0	$\bar{A} + \bar{B} + C$
1	1	1	$\bar{A} + \bar{B} + \bar{C}$

Canonical Product of Sum Expression

This is defined as the logical product of all the maxterms derived from the rows of truth table, for which the value of function is 0. It is also known as the maxterm canonical form. The canonical product of sum expression can be given in a compact form by listing the decimal codes corresponding to the maxterm contain a functional value of 0.

Low cost per gate. Lower Power drain increased circuit complexity	Very low speed, multiple power source needed. Reliability only fair
Very low power drain high noise immunity compatible to existing logic families	Very low speed, multiple power source needed
Speed power product constants low very high package density cheap	Low speed power noise margin.

Q.3 What do you understand by following properties of logic family :

- (i) Fan out
- (ii) Figure of merit
- (iii) Noise margin
- (iv) Current mode logic

[R.T.U. 2017]

Ans.(i) Fan-out : Fan-out is the maximum number of similar logic gates that a gate can drive without any degradation in voltage levels. Very often a gate will drive several other gates. Each driven gate requires a certain current which must be supplied by the driving gate. The driving gate must be capable of supplying this current while maintaining the required voltage level. In part this is a function of the output impedance of the driving gate and the input impedance of the driven gates. Usually, in a given logic family, gates drive others of the same type. If their output impedance is low while their input impedance is high, then one gate can often drive many others.

(ii) Figure of merit : The figure of merit of a digital IC is defined as that product of speed and power figure of merit : propagation time (ns) $\times$ power (mW).

(iii) Noise Immunity or Noise Margin : The term noise denotes an unwanted signal voltage, e.g. hum, transients and glitches.

Noise can sometimes cause the input voltage of a logic gate to drop below V_{IH} (min) or rise above V_{IL} (max), which leads to unreliable operations. Noise immunity is the maximum noise voltage that may appear at the input of a logic gate without changing the logical state of its output. A quantitative measure of noise immunity is called noise margin.

The difference between the operating input-logic voltage level and the threshold voltage is called the noise margin of the circuit. The manufacturer usually quotes the noise margin, which refers to the amplitude of the noise voltage that may cause the logic level to change. In the worst case, a TTL gate functions properly as long as the noise margin is kept less than 0.4V.

(iv) Current mode logic or Wired Logic : With some logic gates, the outputs of the gates are connected together so some output is obtained at this connection. This is known as wired logic. Let us consider the two open collectors. TTL NAND gates as shown in fig.

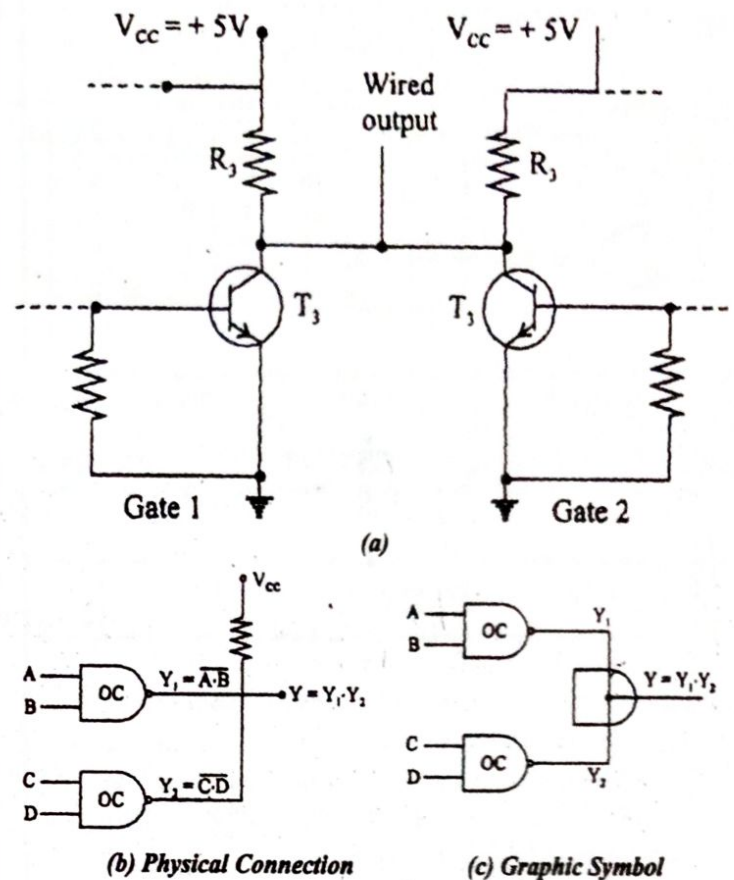


Fig. : Wired Logic

If the outputs are connected together, then the wired output is obtained at the collector of the parallel connection of the two output transistors. It is shown in fig.(a).

The wired output will be high if both the output transistors are in cut off. If atleast one transistor gets into saturation, the wired output will be low. So the output will be high if and only if the outputs of both the individual gates are high. It means the AND operation is performed by the wired connection. It is shown in fig (b). It shows the physical wiring with a common resistor. The graphic symbol for wired connection is shown in fig (c). The AND gate is implicit by the wired connection and is referred to as a wired AND gate. If the two TTL gates correspond to the boolean functions $\overline{AB}$ and $\overline{CD}$, then the wired output will be

$$Y = Y_1 \cdot Y_2$$

$$Y = \overline{AB} \cdot \overline{CD} = \overline{AB + CD}$$

The output can be referred to as an AND-OR-INVERT function. When two TTL NAND gates are connected in parallel, the effective resistance will be $R_3/2$. So the power dissipation increases when the output transistor T_3 goes into saturation.

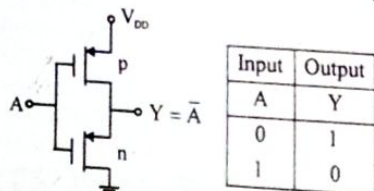


Fig. 3 : CMOS NOT Gate

In this circuit, p and n-channel MOSs are connected in series. The input A to the circuit is the common gate terminal and output Y is a common drain terminal. If the input A is at Logic 0 i.e. the voltage is 0V then both the MOSFETs are at zero potential. The gate to source voltage will be $-V_{DD}$ in the p-channel and 0V in the n-channel device. Therefore, the p-channel will be ON and n-channel will be OFF. The output y will be at logic 0 or low level i.e. 0V.

Now, according to given question, block 1 is inverter so its output is $\bar{A}$. Then the reduced circuit is NOR Gate.

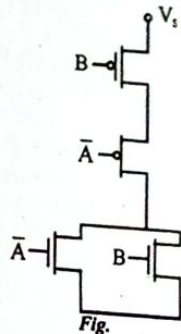


Fig.

$$\text{Therefore, } (\bar{A} + \bar{B}) = \overline{A \cdot B} = A \cdot B$$

- Q.5 (a) Explain the working of CMOS inverter.
(b) Discuss CMOS NAND and NOR gates.

[R.T.U. Dec. 2016]

Ans.(a) CMOS Inverter

When input is low:

$$V_{gs} \text{ for PMOS} = 0, V_g - V_s = 0 - V_{DD} = -V_{DD}$$

$$V_{gs} \text{ for NMOS} = 0 - 0 \text{ volts} = 0 \text{ volts}$$

So, PMOS will conduct and NMOS will turn OFF.

$$\text{So, } V_o = V_{DD}$$

When input is high

$$V_{gs} \text{ for PMOS} = V_{DD} - V_{DD} = 0 \text{ volts}$$

$$V_{gs} \text{ for NMOS} = V_{DD} - 0 = V_{DD}$$

So, PMOS will be turn off and NMOS will conduct

$$\text{So, } V_o = 0$$

Thus in either case one MOSFET is ON and other will be OFF. So the DC power dissipation of the CMOS circuit is

extremely low in the order of 10mW. Most of the power dissipation occurs when the input is changing its state (0 $\rightarrow$ 1 or 1 $\rightarrow$ 0)

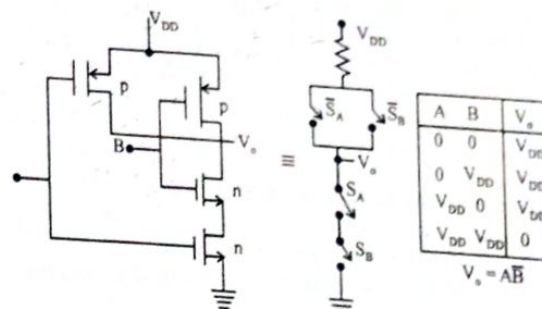


Fig.

Ans.(b) CMOS NAND GATE : Figure 1. shows a CMOS NAND gate with two inputs along with the arrangement of the complementary pairs (N channel and P channel MOSFET). The operation of the CMOS NAND gate is as follows.

Case 1 : When both inputs are LOW, Q_1 and Q_2 are ON and Q_3 and Q_4 are OFF. The output is pulled HIGH through on resistance of Q_1 and Q_2 in parallel.

Case 2 : When input A is LOW and input B is HIGH, Q_1 and Q_4 are ON, and Q_2 and Q_3 are OFF. The output is pulled HIGH through the on resistance of Q_1 .

Case 3 : When input A is HIGH and input B is LOW, Q_1 and Q_4 are OFF, and Q_2 and Q_3 are ON. The output is pulled HIGH through the on resistance of Q_2 .

Case 4 : When both inputs are HIGH, Q_1 and Q_2 are OFF, and Q_3 and Q_4 are ON. In this case, the output is pulled LOW through the on resistance of Q_3 and Q_4 in series to ground.

The table 1 shows the summarized operation of CMOS NAND gate.

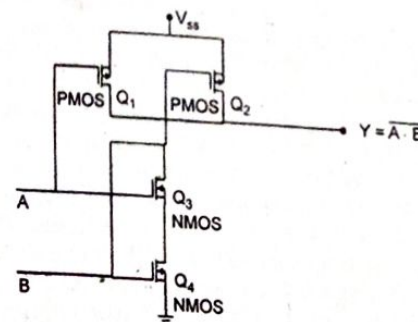


Fig. 1 : CMOS NAND GATE

Table 1 : Truth table of CMOS NAND GATE

Input		MOSFETs		Output	
P Channel	PMOS	NMOS			
A	B	Q_1	Q_2	Q_3	Q_4 - Y
0	0	ON	ON	OFF	OFF 1
0	1	ON	OFF	ON	OFF 0
1	0	OFF	ON	OFF	ON 0
1	1	OFF	OFF	ON	ON 0

L	L	ON	ON	OFF	OFF	1
L	H	ON	OFF	OFF	ON	1
H	L	OFF	ON	OFF	ON	1
H	H	OFF	OFF	ON	ON	0

CMOS NOR GATE

Figure 2 shows a CMOS NOR gate with two inputs. The operation of a CMOS NOR gate is as follows :

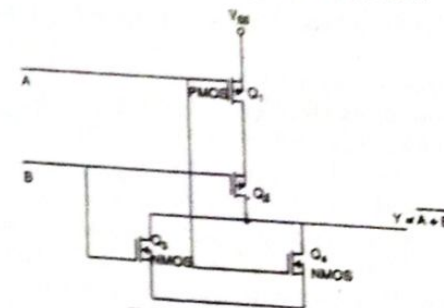


Fig. 2 : CMOS NOR GATE

Case 1 : When both inputs are LOW, Q_1 and Q_2 are ON, and Q_3 and Q_4 are OFF. The output is pulled HIGH through on resistance of Q_1 and Q_2 in series.

Case 2 : When input A is LOW and input B is HIGH, Q_1 and Q_3 are ON, and Q_2 and Q_4 are OFF. The output is pulled LOW through on resistance of Q_3 .

Case 3 : When input A is HIGH and input B is LOW, Q_1 and Q_3 are OFF, and Q_2 and Q_4 are ON. The output is pulled LOW through on resistance of Q_2 .

Case 4 : When both inputs are HIGH, Q_1 and Q_2 OFF, and Q_3 and Q_4 are ON. In this case, the output is pulled LOW through the on resistance of Q_3 and Q_4 in parallel to ground.

The table 2 shows the summarized operation of CMOS NOR gate.

Table 2 : Truth table of CMOS NOR GATE

Input		MOSFETs		Output Y	
P Channel	PMOS	NMOS			
A	B	Q_1	Q_2	Q_3	Q_4
0	0	ON	ON	OFF	OFF 1
0	1	ON	OFF	ON	OFF 0
1	0	OFF	ON	OFF	ON 0
1	1	OFF	OFF	ON	ON 0

Q.6 With the help of a neat diagram explain the working of:

- (a) a MOS inverter
(b) a two I/P MOS NAND gate, and
(c) a two input CMOS NOR gate.

[R.T.U. 2015]

Q.7 Explain a basic ECL NOR/OR gate circuit.

[R.T.U. 2012, Raj. Univ. 2003]

OR
With the help of a neat diagram, explain the working of a two-input ECL OR/NOR gate. [R.T.U. 2015]

OR
Explain the working of ECL NOR/OR gate. [R.T.U. 2011]

OR
Explain the working of ECL NOR gate. [R.T.U. 2008]

Ans. Emitter coupled logic (ECL): Emitter coupled logic is the fastest of all logic families and therefore is used in applications where very high speed is essential. High speeds have become possible in ECL because the transistors are never driven into saturation and thereby the storage time is eliminated. Here, rather than switching the transistors from ON to OFF and vice-versa, they are switched between cut-off and active regions. Propagation delays is less than 1 ns per gate have become possible in ECL.

Basically, ECL is realized using difference amplifiers in which the emitters of the two transistors are connected and hence it is referred to as emitter-coupled logic.

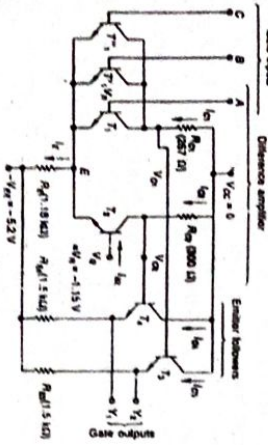


Fig. 1 : 3-input ECL OR/NOR Gate

A 3-input ECL gate is shown in fig. 1, which has three parts. The middle part is the difference amplifier which performs the logic operation.

Emitter followers are used for d.c. level shifting of the outputs, so that $V(0)$ and $V(1)$ are same for the inputs and the outputs. Note that two output Y_1 and Y_2 are available in this circuit which are complementary. Y_1 corresponds to OR logic and Y_2 to NOR logic and hence it is named as an OR/NOR gate.

Additional transistors are used in parallel to T_1 to get the required fan-in. There is a fundamental difference between all other logic families (including MOS logic) and ECL as far as the supply voltages is concerned.

In ECL, the positive end of the supply is connected to ground in contrast to other logic families in which negative end of the supply is grounded.

This is done to minimize the effect of noise induced in the power supply, and protection of the gate from an accidental short circuit developing between the output of a gate and ground. The voltages corresponding to $V(0)$ and $V(1)$ are both negative due to positive end of the supply being connected to ground. The symbol of an ECL OR/NOR gate is shown in fig. 2.



Fig. 2 : The Symbol for a 3-input OR/NOR gate

Verification

(i) Assume all inputs to be LOW.

Let us assume that the input transistors T_1 , T_1' , T_1'' are cut-off and T_2 is conducting in the active region.

The voltage at the common emitter is

$$V_E = V_{E2} - V_{BE2} = -1.15 - 0.7 = -1.85 \text{ V}$$

The current

$$I_E = \frac{V_E - (-V_{EE})}{R_E} = \frac{-1.85 + 5.2}{1.18} = 2.84 \text{ mA}$$

Since $I_{E2} < I_{C2}$, therefore $I_{C2} \approx I_E$

$$V_{O2} = -0.3 I_{C2} = -0.3 (2.84) = -0.852 \text{ V}$$

Transistor T_4 will be conducting and the output of

$$Y_1 = Y_{O2} - V_{BE4} = -0.852 - 0.7 = -1.55 \text{ V}$$

which is assumed to be $V(0)$.

Therefore, if all the inputs are at $V(0) = -1.55 \text{ V}$, then the base-to-emitter voltage of the input transistor is

$$V_{BE} = V_{I1} - V_E = -1.55 + 1.85 = 0.3 \text{ V}$$

which is less than the cut-in voltage (0.5 V) of the transistor and hence the input transistors are non-conducting, as was assumed above.

The base and collector of T_3 are effectively at the same potential, hence T_3 behaves as a diode. The current flowing through this diode is approximately 3 mA which corresponds to a voltage of about 0.75 V across the diode. Therefore, the voltage at $Y_2 = -0.75 \text{ V}$ which is assumed to be $V(1)$. This shows that Y_1 and Y_2 are complementary, i.e. $Y_2 = \bar{Y}_1$.

(ii) Assume at least one input to be HIGH. Corresponding to this the input transistor T_1 is assumed to be conducting and T_2 to be cut-off.

$$\text{Then } V_E = -V_{BE1} = -0.75 - 0.7 = -1.45 \text{ V}$$

$$\text{Hence, } V_{BE2} = V_{O2} - V_E = -1.15 + 1.45 = 0.3 \text{ V which}$$

verifies the assumption that T_2 is cut-off.

$$\text{The voltage } V_{O1} = -R_{C1} \times I_{C1}$$

$$\text{where } I_{C1} = \frac{V_E - (-V_{EE})}{R_E} = \frac{(-1.45 + 5.2)}{1.18} = 3.18 \text{ mA}$$

Since the collector current of T_1 is higher than the collector current of T_2 when it is conducting, hence $R_{C1} < R_{C2}$ to get the same voltage levels.

This gives voltage at $Y_2 = -1.55 = V(0)$. The voltage at $Y_1 = -0.75 = V(1)$. From (i) and (ii), we see that OR function is performed at Y_1 and NOR at Y_2 . Hence it is an OR/NOR gate. Its voltages corresponding to logic 0 and 1 are -1.55 V and -0.75 V respectively. The logic swing is 0.8 V .

Characteristics of ECL Family

There are following advantages and disadvantages of ECL families:

Advantages

1. Since the transistor works in active and cutoff region, speed of ECL is very high.
2. Since the input resistance is very high and the output resistance is very low, large fanout is possible.
3. Complementary outputs are also available.
4. The current flow in ECL circuit remains fixed so current switching spikes are not present.
5. Connecting to ECL circuit provides OR operation.

Disadvantages

1. The logic levels are 0.8 V for 1 and -1.70 V for logic 0. There is small logic swing (900 mV) available.
2. The noise margin of ECL is poor (1200 mV).
3. Since all the transistors work in active region, power dissipation is high.
4. Level shifter are required for interfacing with other families.
5. The speed is reduced by the heavy capacitive load.

Q.8 Explain the working of CMOS-NAND and NOR gates.

[R.T.U. 2013, 2011]

Ans. Refer to Q.5(b).

PART-C

Q.9 Draw a neat circuit of TTL (Transistor Transistor Logic) NAND gate with totem pole output and explain.

[R.T.U. 2018, Dec 2016]

Ans. Transistor-Transistor logic is so named because it is built of transistors only.

Physical Construction

Fig. (1) shows a TTL circuit configuration for a two input NAND gate. The inputs A and B comes into Q_1 which is the Multi Emitter Transistor. A unique feature of TTL is, inputs enter via a transistor which has got several emitters.

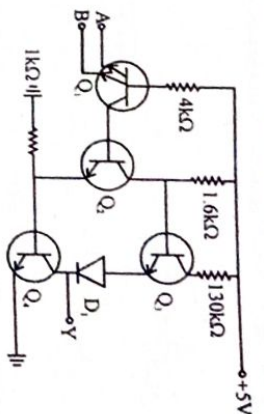


Fig. 1 : Two Input TTL NAND Gate

Q_1 : A Multi-emitter transistor performing AND function on inputs A and B.

Q_2 : A phase splitter

Q_3 and Q_4 : Output stage (Known as the Totem pole Stage)

The TTL output voltage is determined by Q_2 . The phase splitter transistor which drives Q_3 and Q_4 ; these are called totem pole transistor. Depending upon Q_2 is ON or OFF totem transistors changes the circuit output. The diode equivalent circuit is

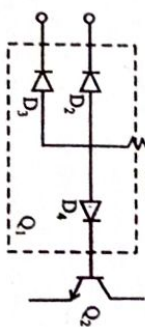


Fig. 2 : Diode Equivalent

Operation

The input voltages are either low or high. If either A or B or both are Low, the corresponding diode conducts and base of Q_1 is pulled down to 0.7V. This reduces the base voltage of Q_2 to almost zero. Therefore Q_2 cuts off. With Q_2 is off, no current enters the base of Q_3 and is turned off. Now the high collector voltage of Q_2 turns Q_3 ON. Since Q_3 acts as an emitter follower the Y output is pulled up to high voltage.

On the other hand when A and B both are high, the emitter diodes of Q_1 are reverse biased making them OFF. This causes the collector diode D_4 to go in to forward conduction. This forces the Q_2 base to go high with Q_2 is ON, it supplies the current to Q_4 , driving it in to saturation. At the same time the low collector voltage of Q_2 holds Q_3 OFF, which results in zero (low) output.

Table shows the I/P and O/P conditions

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

current for the low power Schottky TTL gate. But the CMOS driver can source 360 μ A current which is much higher than the high state input current for the power Schottky TTL. Hence, only one low power Schottky TTL can be connected to a CMOS driver (i.e. fanout = 1).

However, a CMOS can not drive a standard TTL. The worst case low state input current (i.e. I_{IL}) of a standard TTL gate is 1.6 μ A. But a CMOS driver with low output can sink only 360 μ A current. Hence, a standard TTL can not be connected to the output of a CMOS gate.

The above problems can be eliminated by connecting a CMOS buffer, a chip with larger output currents, directly to the CMOS driver output.

Q.17 Define the following characteristics of digital ICs.

- Propagation delay/Speed of operation/Gate delay** [R.T.U. 2014, 13, 11, 08, Raj. Univ. 2006, 2004]
- Power dissipation** [R.T.U. 2014, 13, 11, 08, Raj. Univ. 2006, 2004]
- Fan in** [R.T.U. 2014, 2010, Raj. Univ. 2006, 2004]
- Fan out** [R.T.U. 2014, 13, 11, 08, Raj. Univ. 2006, 04]
- Noise margin/Noise Immunity** [R.T.U. 2014, 13, 11, 10, 08, Raj. Univ. 2006, 2004]

Ans. Some of the important parameters or properties of various logic families are listed as follows:

- Speed of operation (Propagation delays)
- Power dissipation
- Fan-in
- Fan-out
- Noise immunity
- Operating temperature
- Power supply requirements

The comparison of performance of digital ICs may be made with reference to the above properties.

(i) Speed of Operation/Propagation Delay/Gate Delay

The speed of operation of an IC is expressed in terms of propagation delay. Propagation delay is defined as the time taken for the output of a gate to change after the inputs have changed.

A logic signal always experiences a delay in going through a circuit. The two propagation delay times shown in Fig. 1 are defined as follows:

t_{PLH} : It is the propagation delay time in going from logical LOW (0 state) to logical HIGH (1 state).

t_{PHL} : It is the propagation delay time in going from logical HIGH (1 state) to logical LOW (0 state).

It is evident that t_{PHL} is the delay in the output response as it goes from LOW state to a HIGH state, and vice versa for t_{PLH} . The delay times are measured between the 50% voltage level of the input and output waveforms. In general, the two delays t_{PHL} and t_{PLH} are not necessarily equal and

will vary depending on load conditions. The values of propagation time are a measure of the relative speed of logic circuits. The average of the above two propagation delays $(t_{PLH} + t_{PHL})/2$ is called the average propagation delay and is used to rate the circuit. It is a function of the switching time of the individual transistors or MOSFETs in the circuit.

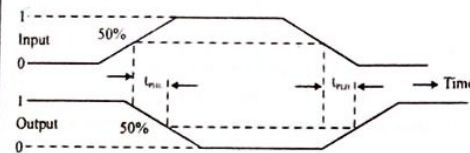


Fig. 1 : Propagation Delays

(ii) Power Dissipation

Power dissipation is a measure of the power consumed by the logic gate when fully driven by all its inputs, and its is expressed in milliwatts or nanowatts. The d.c. or average power dissipation is the product of d.c. supply voltage and the mean current taken from that supply.

(iii) Fan-in

The fan-in of a gate is the number of inputs connected to the gate without any degradation in the voltage levels. For example, an eight-input gate requires one Unit Load (UL) per input. Its fan-in is 8. This parameter determines the functional capabilities of a logic circuit.

(iv) Fan-out : Refer to Q.3(i).

(v) Noise Immunity or Noise Margin : Refer to Q.3(iii).

(vi) Operating Temperature

All IC gates are semiconductor devices that are temperature sensitive by nature. The operating temperature ranges for an IC vary 0°C to +70°C for consumer and industrial application and from -55°C to +125°C for military applications.

(vii) Power Supply Requirements

The amount of power and supply voltage required by an IC are the main parameters to be taken into consideration while choosing a proper power supply.

(viii) Current and Voltage Parameters

The following currents and voltages are very important in designing digital systems. The values given below are for TTL gates only.

(a) High-level input voltage (V_{IH}) [$V_{in(1)}$] : It is the minimum voltage level required for a logical 1 at an input. Its minimum value is 2V

(b) Low-level input voltage (V_{IL}) [$V_{in(0)}$] : It is the maximum input voltage required for a logical 0 (Low) at an input. Its minimum value is 0.8V

(c) High-level output voltage (V_{OH}) [$V_{out(1)}$] : It is the minimum voltage required for a logical 1 state at the output. Its minimum value is 2.4V

(d) Low-level output voltage (V_{OL}) [$V_{out(0)}$] : It is the maximum voltage available at the circuit's output corresponding to the logical 0 state. Its maximum value is 0.4 V

(e) High-level input current (I_{IH}) [$I_{in(1)}$] : The current that flows through an input when a specified high-level voltage is applied to that input.

(f) Low-level input current (I_{IL}) [$I_{in(0)}$] : The current that flows through an input when a specified low-level voltage is applied to that input.

(g) High-level output current (I_{OH}) [$I_{out(1)}$] : The current that flows from an output in the logical 1 state under specified load conditions.

(h) Low-level output current (I_{OL}) [$I_{out(0)}$] : The current that flows from an output in the logical 0 state under specified load conditions.

(i) Current-Sourcing and Current-Sinking Logic
Logic families can be categorized depending upon the flow of current from the output of one logic circuit to the input of another. Current sourcing and current sinking logic gates are illustrated in Fig. 2(a) and (b) respectively.

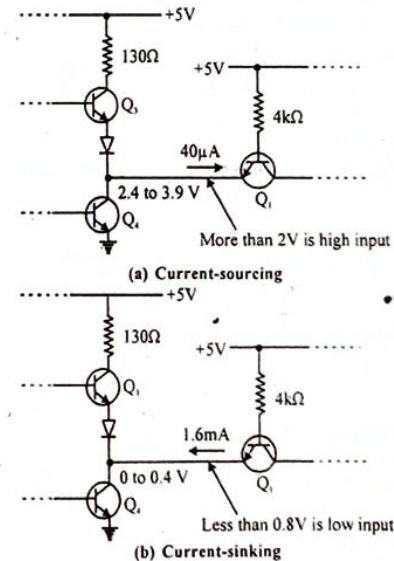


Fig. 2

When a standard TTL gate output is HIGH as shown in Fig. 2(a) a reverse emitter current of 40 μ A flows from transistor Q_1 of driver gate to the emitter of transistor Q_1 of load gate, and hence Q_1 acts as a current source.

When a standard TTL gate output is LOW as shown in Fig. 2(b), an emitter current of 1.6 mA flows from the emitter

of transistor Q_1 of load gate to the collector of transistor Q_1 of driver gate. As Q_1 is saturated, current flows through it to the ground and hence Q_1 acts as a current sink.

Q.18 Explain the following briefly:

- RTL
- ECL
- 3-state output logic [R.T.U. 2013, 2009, Raj. Univ. 2005]
- FET as a switch [R.T.U. 2013]

Ans.(i) Resistor Transistor Logic (RTL) : Refer to Q.15.

Ans.(ii) Emitter Coupled Logic (ECL) : Refer to Q.15.

Ans.(iii) Tristate gate

A very popular output connection that incorporates the benefits of totem-pole and open-collector in the single circuit is the tri-state output connection.

The tri-state (three state) output exhibits three possible output-state conditions as shown in table. Two of these states are the conventional logic 0 and logic 1. The third state is a high impedance (open circuit) state. This means, for all practical purposes, the circuit behaves as if the output is disabled. As a consequence, the output cannot affect or be affected by any external signal at its input terminals. The third state is controlled by a separate control input. A control (Select or Inhibit) input terminal is provided to allow the output to be switched into (or out off) its high impedance condition. When the control input C is 1, the gate behaves like a normal NAND gate providing states of 0 and 1. When the control input C is 0, the output is disabled irrespective of the values of the normal inputs. Tri-state gates are also available with the control input having a complementary effect, i.e. disabling the gate when the control input is 1 and enabling it when it is 0. The high-impedance state of a tri-state gate allows the possibility of making a direct wire connection from many outputs to a common bus line, in which only one output line will be enabled while all other outputs are disabled by their respective control inputs.

Table : Truth table for tri-state TTL NAND gate

A	B	C	Q ₁	Q ₂	Q ₃	Q ₄	Q ₅	Y
0	0	1	ON	OFF	OFF	ON	ON	1
0	1	1	ON	OFF	OFF	ON	ON	1
1	0	1	ON	OFF	OFF	ON	ON	1
1	1	1	OFF	ON	ON	OFF	OFF	0
X	X	0	ON	OFF	OFF	OFF	OFF	Open-circuit

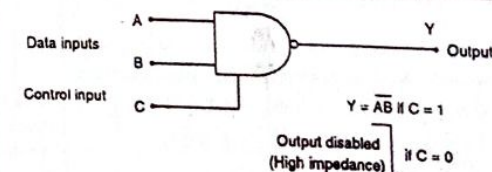


Fig. 1 : Symbol for tri-state NAND gate

Q.4 Find the output 4 : 1 multiplexer as shown in Fig.

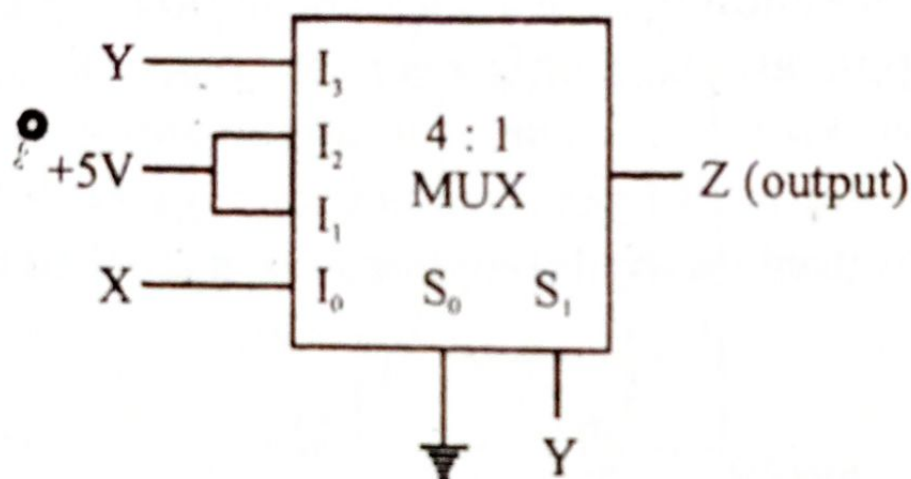


Fig.

[R.T.U. 2017]

Ans.

$$Z = \bar{S}_1 \bar{S}_0 I_0 + \bar{S}_1 S_0 I_1 + S_1 \bar{S}_0 I_2 + S_1 S_0 I_3$$

Here S_0 is always zero.

$$= \bar{S}_1 I_0 + S_1 I_2$$

$$= \bar{Y} X + Y \cdot 1 = Y + X \bar{Y}$$

$$= (Y + \bar{Y})(X + Y)$$

Output

$$\boxed{Z = X + Y}$$

DE.70

Implement a 6×64 decoder using 3×8 line decoders :

The 6×64 decoder will have 6 inputs lines and 64 output lines. It can be implemented using Nine (9) 3×8 decoders with enable input. Each 3×8 decoder has three input lines, eight output lines and an enable input E. If the enable input E is considered as fourth input line as connected to input D of the decoders as shown in fig. 4.

Since these are active high enable inputs, so when $D = 0$ is applied to input D, the first-4 decoder ($D_0 - D_3$) will get enabled and last -4 decoder ($D_4 - D_7$) will be disabled.

When $D = 1$ is applied, the first 4 decoder ($D_0 - D_3$) will get disabled and last-4 decoder ($D_4 - D_7$) will be enabled.

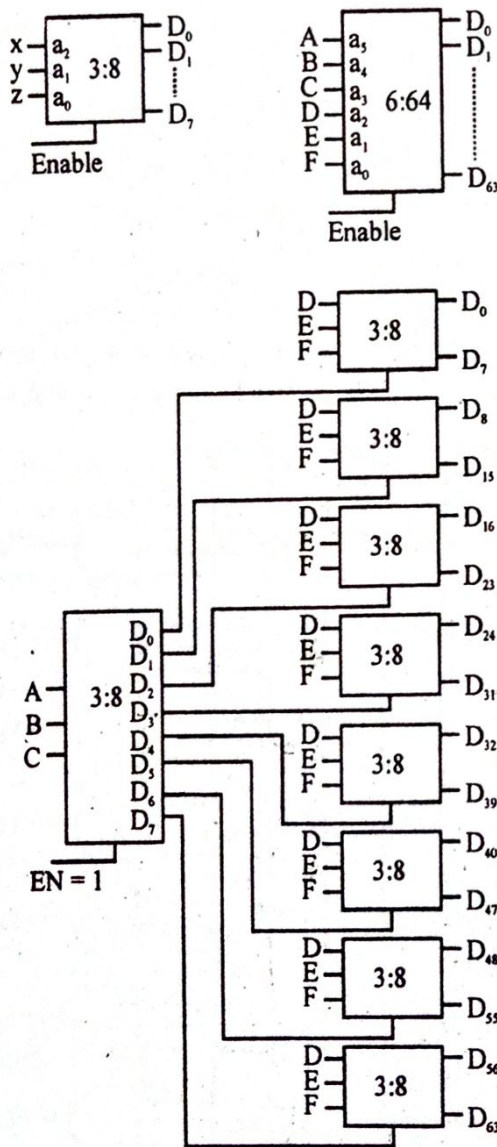


Fig. 4 : Total $3 \times 8 = 9$ Decoder

Q.8 Implement a full subtractor using two $4 : 1$ multiplexer. [R.T.U. 2017]

Ans.

Table : Truth Table of Full Subtractor

Inputs			Difference	Borrow
A	B	B_1	D	B_1
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

- Form the truth table of full subtractor B and B_1 inputs are connected to the select lines S_1 and S_0 respectively of $4 : 1$ multiplexer.
- A is used as data input line.
- The equation for difference (D) and borrow output (B_0) from the truth table of full subtractor is:

$$D = \sum m(1, 2, 4, 7),$$

(this is the same as that of sum equation)

$$B_0 = \sum m(1, 2, 3, 7)$$

- Design table for difference (D):

	D_0	D_1	D_2	D_3
$\bar{A}$	0	1	2	3
A	4	5	6	7

Input to MUX-1 (Difference D) $\rightarrow$ A $\bar{A}$ $\bar{A}$ A

Fig.

- Design table for borrow (B_0) :

	D_0	D_1	D_2	D_3
$\bar{A}$	0	1	2	3
A	4	5	6	7

Input to MUX-2 (Borrow B_0) $\rightarrow$ A $\bar{A}$ $\bar{A}$ A

Fig.

Implementing the circuit of full subtractor using two $4 : 1$ multiplexers.

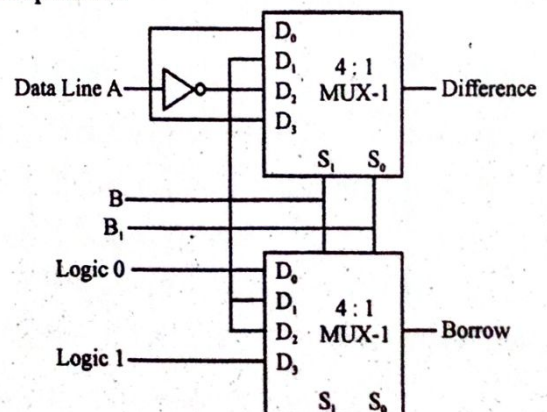


Fig. : Full Subtractor using Two $4 : 1$ Multiplexers

are capable of handling both analog and digital applications. In analog applications, multiplexers are made up of relays and transistor switches, whereas in digital applications, the multiplexers are built from standard logic gates. When the multiplexer is used for digital applications, it is called a digital multiplexer.

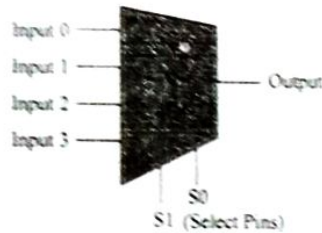


Fig.-Multiplexer

Multiplexer Types

Multiplexers are classified into four types.

- 2-1 multiplexer (1 select line)
- 4-1 multiplexer (2 select lines)
- 8-1 multiplexer (3 select lines)
- 16-1 multiplexer (4 select lines)

Q.10 (a) Draw gate level schematic of a 1-to-4 decoder as component realize a 1-to-16 decoder.

(b) Signals A, B, C, D and A are available. Using only one 8 to 1 MUX and no other gate, implement the expression.

$$F = (A, B, C, D) = BC + AB\bar{D} + \bar{A}\bar{C}D$$

[R.T.U. Dec. 2016]

Ans.(a)

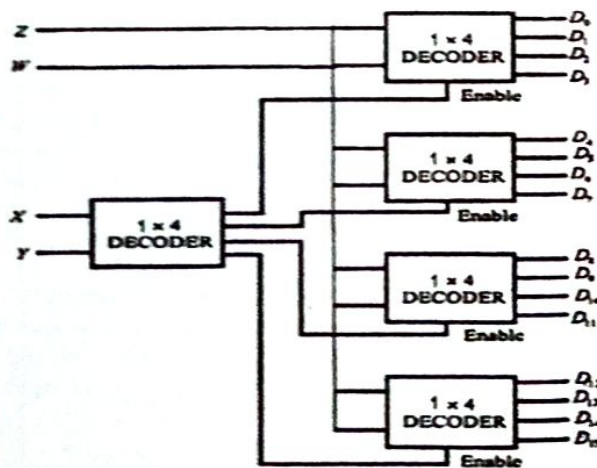


Fig. : 1-4 decod

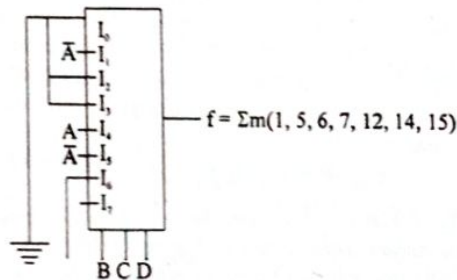
DE.72

Ans.(b)

	$\bar{A}\bar{B}$	$\bar{A}B$	$A\bar{B}$	AB
$\bar{C}\bar{D}$	0	1	3	2
$\bar{C}D$	4	5	7	6
CD	12	13	15	14
$C\bar{D}$	8	9	11	10

$$f(A, B, C, D) = \sum m(1, 5, 6, 7, 12, 14, 15)$$

	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
$\bar{A}$	0	1	2	3	4	5	6	7
A	8	9	10	11	12	13	14	15
$\bar{A}$	0	0	0	0	1	1	1	1



Q.11 Implement the following expression-

(i) $y = A\bar{B}C$ using 2 input NAND only

(ii) $y = \sum m(0, 2, 3, 6, 8, 9, 12, 14)$ using multiplexer

(iii) $y = \sum m(2, 3, 9, 11)$ using 4 to 16 line decoder

(iv) 16 x 1 mux using 2 x 1 mux [R.T.U. Jan. 2016]

Ans.(i) $y = A\bar{B}C$

$$= \overline{\overline{A\bar{B}C}}$$

$$= \overline{AC + \bar{B}}$$

$$y = \overline{AC + \bar{B}}$$

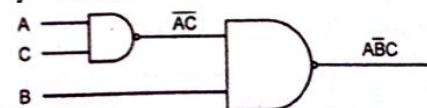


Fig.

(ii) $y = \sum m(0, 2, 3, 6, 8, 9, 12, 14)$

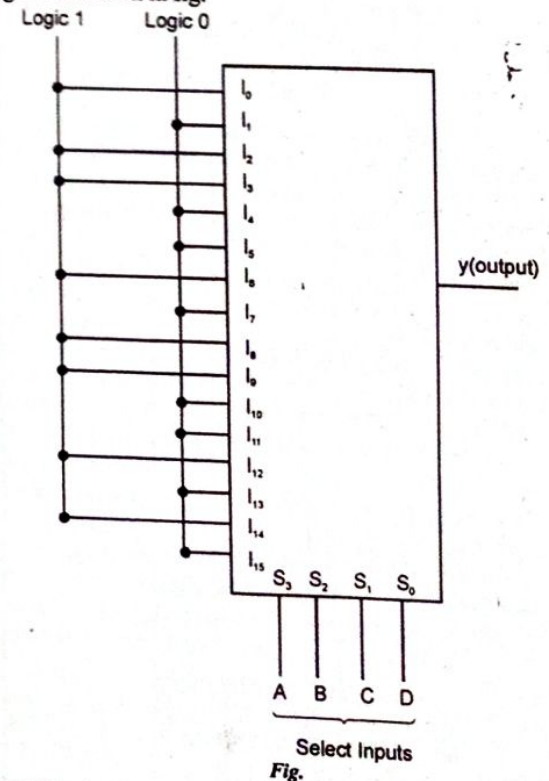
Since the given Boolean function has four variables. To implement it using a multiplexer 16:1 MUX is needed when type-0 method is used. The truth table of the given function is

B.Tech. (III Sem.) C.S. Solved Papers

Truth Table

Inputs				Output
A	B	C	D	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

In type-0 method, the input lines which are associated with minterms to logic 1 and remaining lines are connected to logic 0 as shown in fig.



Select Inputs
Fig.

(iii) The implementation of the function $y = \sum m(2, 3, 9, 11)$ using 4 to 16 line decoder is shown in fig.

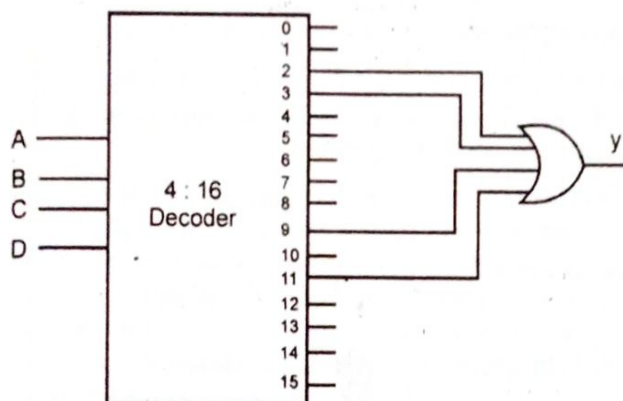


Fig.

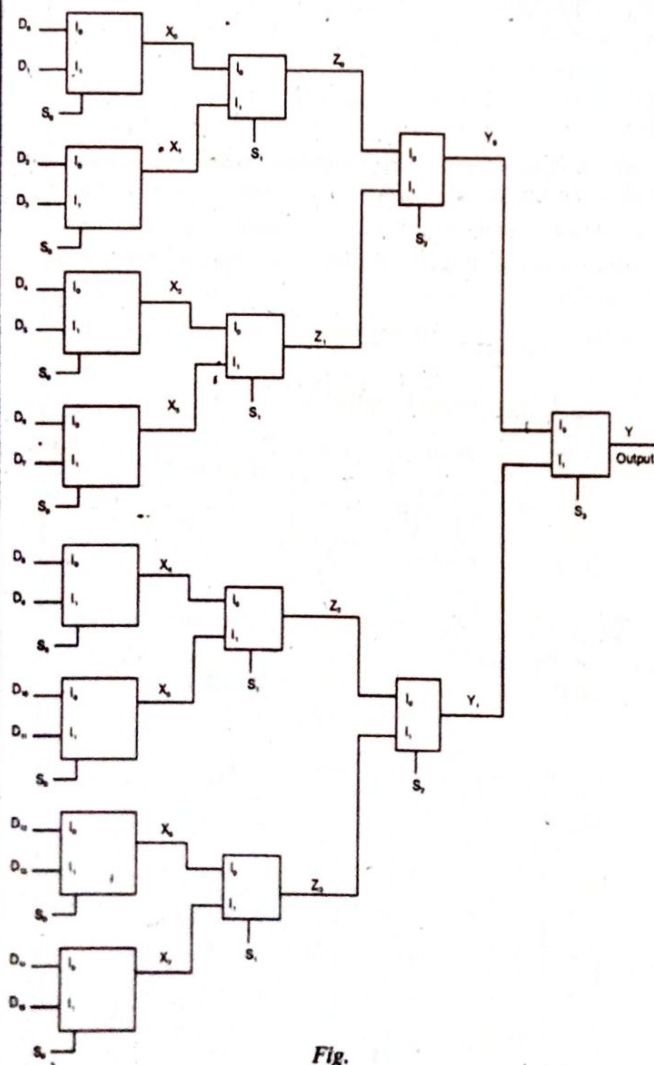
(iv) 16×1 mux using 2×1 mux

Fig.

Q.12 Design a Binary to Gray code converter.

[Raj. Univ. 2007, 2001]

OR

Design a 4 bit Binary to Gray code converter and draw the logic diagram. [R.T.U. 2015]

OR

Design and implement a 4-bit binary to gray code converter. [R.T.U. 2009, 2008]

Ans. A code converter is a logic circuit that changes data presented in one type of binary code to another type of binary code.

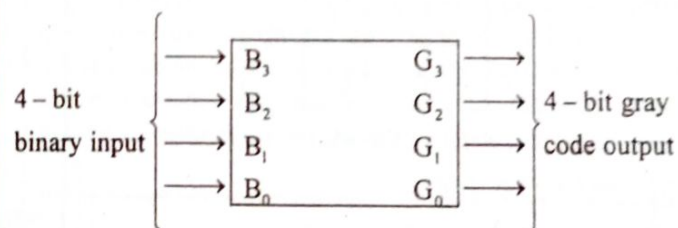


Fig. : Block Diagram of 4-bit Binary to Gray Code Converter

The BCD to 7 segment decoder is also a code converter. The block diagram of a 4 bit binary to gray code converter is shown in figure. It has four inputs ($B_3 B_2 B_1 B_0$) representing 4 bit binary numbers and four outputs ($G_3 G_2 G_1 G_0$) representing 4 bit gray code. The truth table is shown in table.

Table : Truth Table

Binary inputs				Gray code outputs			
B_3	B_2	B_1	B_0	G_3	G_2	G_1	G_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

From the truth table, the logic expressions for the gray code outputs can be written as:

$$G_3 = \sum_m(8, 9, 10, 11, 12, 13, 14, 15)$$

$$G_2 = \sum_m(4, 5, 6, 7, 8, 9, 10, 11)$$

$$G_1 = \sum_m(2, 3, 4, 5, 10, 11, 12, 13)$$

$$G_0 = \sum_m(1, 2, 5, 6, 9, 10, 13, 14)$$

The above expressions can be simplified using K-map methods as shown below.

$B_3 B_2$ $B_1 B_0$	00	01	11	10
00	0	0	1	1
01	0	0	1	1
11	0	0	1	1
10	0	0	1	1

(a) K-map for G_3

From the above K-map $G_3 = B_3$

$B_3 B_2$ $B_1 B_0$	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	0	1	0	1
10	0	1	0	1

(b) K-map for G_2

From the above K-map $G_2 = \bar{B}_3 B_2 + B_3 \bar{B}_2 = B_3 \oplus B_2$

$B_3 B_2$ $B_1 B_0$	00	01	11	10
00	0	1	1	0
01	0	1	1	0
11	1	0	0	1
10	1	0	0	1

(c) K-map for G_1

From the above K-map $G_1 = \bar{B}_2 B_1 + B_2 \bar{B}_1 = B_2 \oplus B_1$

$B_3 B_2$ $B_1 B_0$	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	0	0
10	1	1	1	1

(d) K-map for G_0

Fig. 1 : K-map

K-map simplification for binary to gray code converter

From the above k-map $G_0 = \bar{B}_1 B_0 + B_1 \bar{B}_0 = B_1 \oplus B_0$

Now the above expressions can be implemented using Ex-OR gates as shown in figure below :

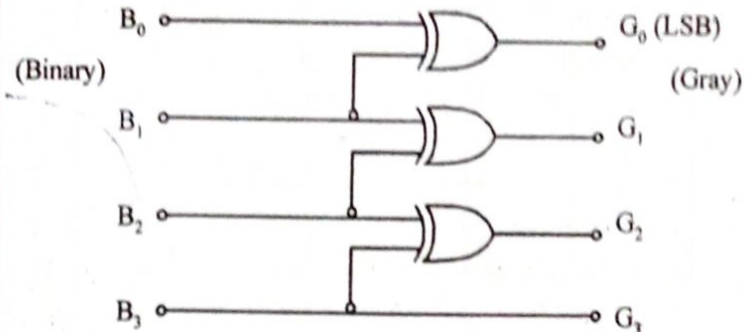


Fig. 2 : Logic Diagram of 4-bit Binary to Gray Code Converter

Q.13 (a) Design a excess - 3 to BCD code converter using 4 bit adder.

(b) Design a full adder using 4×1 multiplexer.

[R.T.U. 2014]

Ans.(a) Excess - 3 to BCD Code Converter using 4 Bit Adder : The excess-3 number can be converted to BCD number by adding 2's complement of 3 into excess-3 number as shown in the Fig.

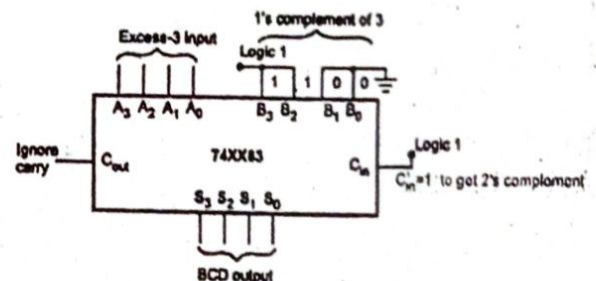


Fig.

The truth table for Excess-3 to BCD code converter is as shown in the Table 1.

Table : Truth table for Excess-3 to BCD Converter

K-map simplification

For B_0

E_3	E_2	E_1	E_0	B_3	B_2	B_1	B_0
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	1
0	1	0	1	0	0	1	0
0	1	1	0	0	0	1	1
0	1	1	1	0	1	0	0
1	0	0	0	0	1	0	1
1	0	0	1	0	1	1	0
1	0	1	0	0	1	1	1
1	0	1	1	1	0	0	0
1	1	0	0	1	0	0	1

Q.14 Explain Encoder and Decoder. [Raj. Univ. 2004]
OR

Describe encoders and decoders with suitable diagram. [R.T.U. 2013]

Ans. Encoder : Like decoder, encoder also provides the conversion of binary information from one form to another. Encoder performs the opposite function of a decoder. It gives information in more compact form. Encoder has more input lines than output lines or say output lines are less than the input lines. It has m input lines and n output lines. From m input lines, only one is activated at a time and gives an equivalent code on n output lines. The block diagram of encoder is shown in fig. 1. Encoder is a digital or combinational circuit that converts an active signal into a coded output signal. Encoders are used to reduce the number of bits needed to represent the given information. A practical use of encoders is for transmitting information in a digital system. Encoding is also useful if the information is to be stored for later use because fewer bits are required to be stored.

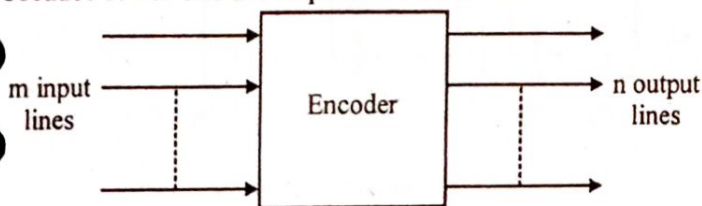


Fig. 1 : Block Diagram of Encoder

Decoder : Sometimes, digital information represented in binary form must be converted into some other form. This is obtained by multiple inputs-multiple outputs logic circuit known as decoder. The most commonly used decoder is n to 2^n line decoder. The block diagram of $(n : m)$ decoder is shown in fig. 1. It has n input lines and $m = 2^n$ output lines.

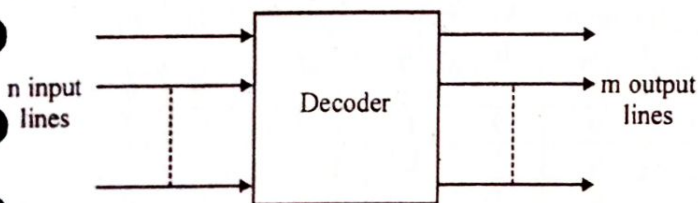


Fig. 2 : Block Diagram of $n : m$ Decoder

In a decoder, m output lines are greater than n input lines. But when input lines are equal to output lines then it is known as a converter. It converts one form of n bit binary information into another form of m bit binary information. For example BCD to excess-3 code converter, binary to gray code converter etc. In other case when m output lines are less than 2^n then this type of decoder is known as function specific decoder. For example, BCD to decimal decoder, BCD to seven segment decoder etc. The main applications of decoder are data demultiplexing, digital display, digital to analog converters and memory addressing.

Q.15 Construct 4-bit serial adder using shift registers and logic gates. Explain its operation.

Ans. Serial Binary Adder : The block diagram of a serial binary adder is shown in fig. 1. It is a synchronous circuit with two input terminals designated X_1 and X_2 which carry the two binary numbers to be added and one output terminal Z which represents the sum. The inputs and outputs consist of fixed-length sequences of 0s and 1s. The addition is performed serially, i.e. the least significant digits of the numbers X_1 and X_2 arrive at the corresponding input terminals at t_1 ; a unit time later the next significant digits arrive at the input terminals, and so on. The time interval between the arrival of two consecutive input digits is determined by the frequency of the circuit's clock.

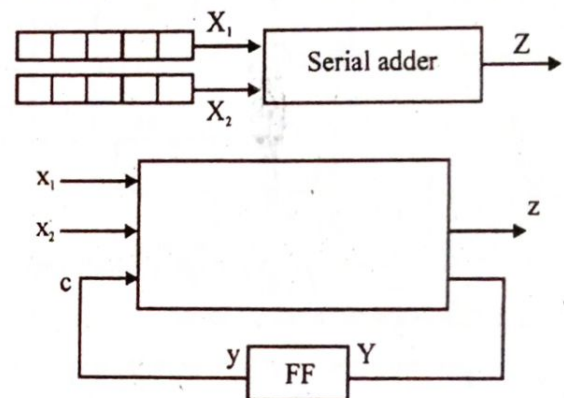
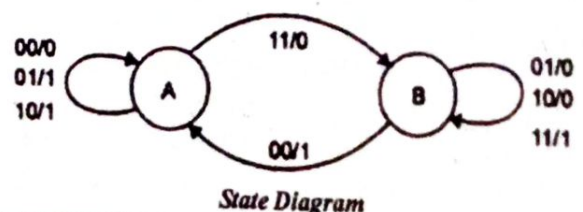


Fig. 1: Block diagram of the serial binary adder.

The delay within the combinational circuit is small with respect to the clock frequency and thus the sum digit arrives at the Z terminal immediately following the arrival of the corresponding input digits at the input terminals.

The output of the serial adder z_i at time t_i is a function of the inputs $x_i(t_i)$ and $x_2(t_i)$ at that time t_i and of a carry which had been generated at t_{i-1} . The carry which represents the past history of the serial adder may be a 0 or 1. So, one FF is required to store it and one state indicates that the carry from the previous addition is a 0, the other state indicates that the carry from the previous addition is a 1.

Let A designate the state of the serial adder at t_i if a carry 0 was generated at t_{i-1} , and let B designate the state of the serial adder at t_i if a carry 1 was generated at t_{i-1} . The state of the adder at the time when the present inputs are applied is referred to as the present state (PS), and the state to which the adder goes as a result of the new carry value is referred to as the next state (NS).



State Diagram

Table 1: Truth table of octal-to-binary encoder

Inputs								Outputs			
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	Y_2	Y_1	Y_0	
1	0	0	0	0	0	0	0	0	0	0	
0	1	0	0	0	0	0	0	0	0	1	
0	0	1	0	0	0	0	0	0	1	0	
0	0	0	1	0	0	0	0	0	1	1	
0	0	0	0	1	0	0	0	1	0	0	
0	0	0	0	0	1	0	0	1	0	1	
0	0	0	0	0	0	1	0	1	1	0	
0	0	0	0	0	0	0	1	1	1	1	

The truth table shows that (LSB of output code) must be 1 whenever the input D_1 OR D_3 OR D_5 OR D_7 is HIGH.

Thus,

$$Y_0 = D_1 + D_3 + D_5 + D_7$$

Similarly

$$Y_1 = D_2 + D_3 + D_6 + D_7$$

$$Y_2 = D_4 + D_5 + D_6 + D_7$$

Using the above expressions, the octal-to-binary encoder can be implemented using three 4-input OR gates as shown in fig. The circuit is designed in such a way that, when D_0 is HIGH, the binary code 000 is generated; when D_1 is HIGH, the binary code 001 is generated, and so on.

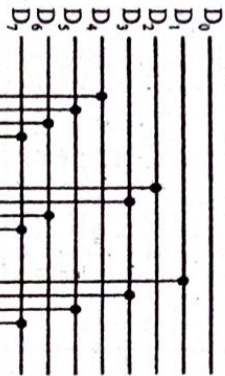


Fig. 3: Octal-to-binary encoder

The design is made simple by the fact that only eight out of the total of 2^3 possible input conditions are used.

Q18 (a) Write short note on Diode switching matrix. Design and explain the working of BCD to 7-segment decoder.

[R.T.U. 2014]

Ans.(a) Diode Switching Matrix

A encoder has n inputs and generates n bit code depending upon which of the inputs is excited. A general block diagram of an encoder is shown in fig.1.

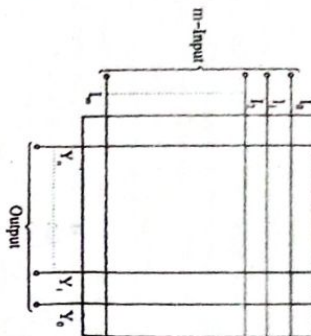


Fig. 1: Block diagram of diode switching matrix

It has array matrix of wires to design an encoder in which row lines and column lines will make the number of squares. Some of the squares are occupied by the diode depending on the circuit design. Let us take an example of 8 to 3 encoder whose truth table is shown in table.

Table 1: Truth table of octal to binary encoder

Inputs								Outputs			
I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7	Y_2	Y_1	Y_0	
1	0	0	0	0	0	0	0	0	0	0	
0	1	0	0	0	0	0	0	0	0	1	
0	0	1	0	0	0	0	0	0	1	0	
0	0	0	1	0	0	0	0	0	1	1	
0	0	0	0	1	0	0	0	1	0	0	
0	0	0	0	0	1	0	0	1	0	1	
0	0	0	0	0	0	1	0	1	1	0	
0	0	0	0	0	0	0	1	1	1	1	

The OR gates required in the expression of Y_0 , Y_1 , Y_2 can be implemented with diodes. The diode switching matrix for the above encoder is shown in fig. 2.

In Fig. 2 S_0 to S_7 are switches corresponding to each input line. So, each line is activated only when the respective switch of this line is switched on. In the diode matrix R-0 to R-7 are row lines and C-1 to C-3 are column lines. Each diode is connected within the square which is connected by one of the row and column line. These diodes are connected between R-1 and C-3, R-2 and C-2, R-3 and C-2, R-3 and C-3 and so on in the similar manner between some other rows and columns. Any line activated through the respective switch of the diode, if any, connected in this line will come into forward direction and +ve point of supply is connected to the column through which this diode is connected and so the current is flowing through this diode and resistance to which this column is connected.

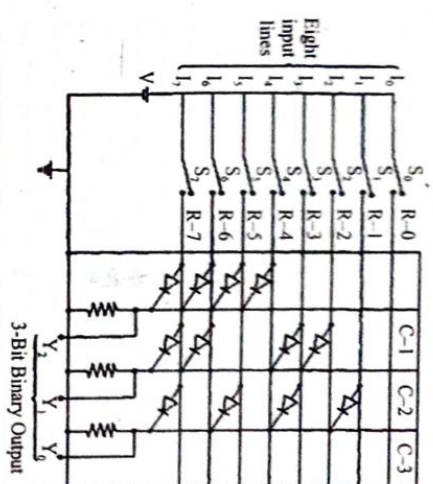


Fig. 2: Octal to binary encoder 3 bit-binary output

Since output lines are received through each column line i.e. Y_0 from C-3, Y_1 from C-2 and Y_2 from C-1, hence whenever the current is flowing in the resistance through respective columns, these outputs will be at high logic i.e. 1 otherwise 0 considering positive logic. Therefore, it is observed from the truth table that Y_0 will be high only when I_1 or I_3 or I_5 or I_7 is 1. Similarly Y_1 will be high only when I_2 or I_3 or I_6 or I_7 is high and Y_2 will be high only when I_4 or I_5 or I_6 or I_7 is high. Therefore, in Boolean notation, we can write Y_0 , Y_1 and Y_2 as follows:

$$Y_0 = I_1 + I_3 + I_5 + I_7$$

$$Y_1 = I_2 + I_3 + I_6 + I_7$$

$$Y_2 = I_4 + I_5 + I_6 + I_7$$

Output shows the ORing of inputs by use of diodes. So, by connecting the diodes in appropriate place in diode switching matrix, one can obtain any type of encoding. An encoder array shown in fig. 2 is called a rectangular diode matrix.

Similarly, decoder can also be made as a rectangular diode matrix.

Ans.(b) BCD to Seven-Segment Decoder/Driver

A seven-segment display is normally used for displaying any one of the decimal digits, 0 through 9. ABCD to-seven segment decoder accepts a decimal digit in BCD and generates the corresponding seven-segment code.

Figure 3(a) shows a seven-segment display composed of seven elements or segments. Each segment is made up of a material that emits light when current is passed through it. Most commonly used displays are LEDs and incandescent filaments. Note that letters a, b, c, d, e, f and g run clockwise from the top of each segment. For instance, to display a 1, the segments b and c have to be illuminated; to display a 0, the segments a, b, c, d, e and f have to be illuminated by properly forward biasing the LEDs in the selected segments.



(a) Seven-segment display

(b) Display of Decimal Digits is a seven segment display

Fig. 1

Design of BCD to - Seven Segment Decoder A BCD- to seven segment decoder can be designed using logic gates. A block diagram of BCD to- seven segment decoder with four BCD inputs (A, B, C and D) and seven outputs (a, b, c, d, e, f and g) corresponding to seven segments of a display, is shown in Fig.4. The truth table of the BCD to 7 segment decoder is shown in Table.

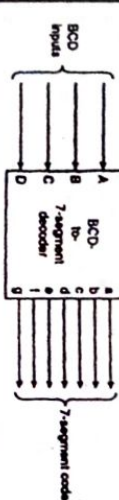


Fig. 2: Block diagram of BCD to-7-segment decoder

Table 1: Truth table of BCD to-7-segment decoder

BCD inputs				Seven segment outputs						
A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	1
0	0	0	1	1	1	1	0	1	1	1
0	0	0	1	1	1	1	1	0	1	1
0	0	1	0	1	1	0	1	1	1	1
0	0	1	1	1	1	0	1	1	1	1
0	1	0	0	1	0	1	1	1	1	1
0	1	0	1	1	0	1	1	1	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	0	1	1	1	1	1
1	0	0	0	1	1	1	1	1	1	1

Since only BCD inputs are valid combinations, the other input combination of four variables corresponding to 10, 11, 12, 13, 14 and 15 can be termed as don't care combinations to aid the simplification of logic expressions. Now the logic expressions corresponding to seven segment can be written from the truth table shown in table as follows.

$$a = \sum_m (0, 2, 3, 5, 6, 7, 8, 9) + \sum_d (10, 11, 12, 13, 14, 15)$$

$$b = \sum_m (0, 1, 2, 3, 4, 7, 8, 9) + \sum_d (10, 11, 12, 13, 14, 15)$$

$$c = \sum_m (0, 1, 3, 4, 5, 6, 7, 8, 9) + \sum_d (10, 11, 12, 13, 14, 15)$$

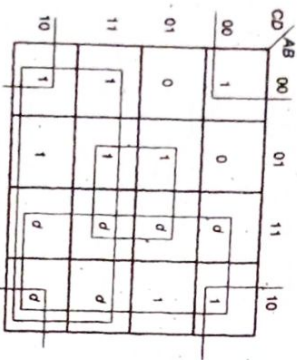
$$d = \sum_m (0, 2, 3, 5, 6, 8, 9) + \sum_d (10, 11, 12, 13, 14, 15)$$

$$e = \sum_m (0, 2, 6, 8) + \sum_d (10, 11, 12, 13, 14, 15)$$

$$f = \sum_m (0, 4, 5, 6, 8, 9) + \sum_d (10, 11, 12, 13, 14, 15)$$

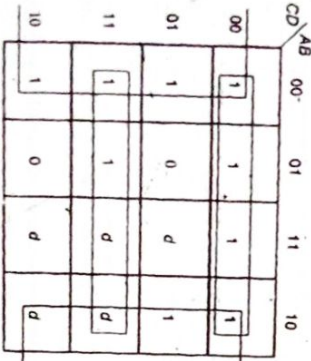
$$g = \sum_m (2, 3, 4, 5, 6, 8, 9) + \sum_d (10, 11, 12, 13, 14, 15)$$

The above expressions can be simplified using K-map method as shown in fig. 3.



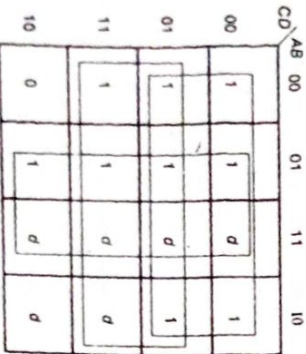
(a): K-map for 7-segment output 'a'.

From the above K-map
 $a = A + C + \overline{B}D + BD = A + C + B \oplus D$



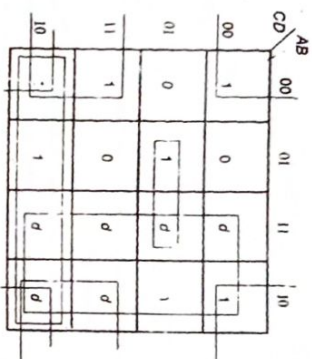
(b): K-map for 7-segment output 'b'.

From the above K-map
 $b = \overline{B} + CD + \overline{C}D = \overline{B} + C \oplus D$



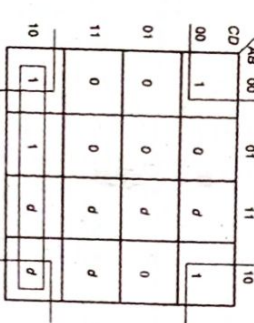
(c): K-map for 7-segment output 'c'.

From the K-map
 $c = B + \overline{C} + D$



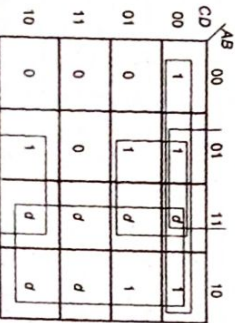
(d): K-map for 7-segment output 'd'.

From the above K-map
 $d = A + BD + \overline{C}D + \overline{B}C + B\overline{C}D$
 $= A + \overline{C}D + \overline{B}C + D + B\overline{C}D$
 $= A + \overline{C}D + B\overline{C} + D + B\overline{C}D$



(e): K-map for 7-segment output 'e'.

From the above K-map
 $e = BD + \overline{C}D = D(\overline{B} + C)$



(f): K-map for 7-segment output 'f'.

From the above K-map
 $f = A + \overline{C}D + \overline{B}C + BD = A + \overline{C}D + B(\overline{C} + D)$

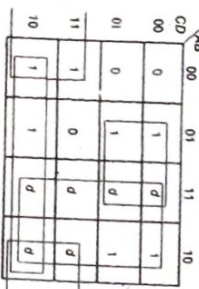


Fig. 3: K-map simplification for BCD to 7-segment decoder
 From the above K-map

$$g = A + \overline{C}D + \overline{B}C + BD = A + \overline{C}D + (B \oplus C)$$

Now, using the above simplified expressions for seven-segment output, the BCD to seven segment decoder can be implemented using logic gates as shown in fig.

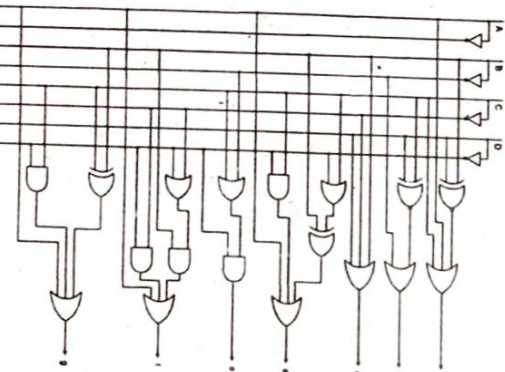


Fig. 4: Logic diagram of BCD-to-7-segment decoder

IC 7446/7447 and IC 7448/7449 - BCD-to-Seven-Segment Decoders. ICs: 7446 and 7447 are BCD to seven segment decoders with active LOW open collector outputs designed for driving common anode seven segment displays. ICs 7448 and 7449 are with active HIGH outputs for driving common cathode seven segment displays.

Fig. 5 shows a BCD to seven segment decoder/driver (TTL 7446 or 7447) used to drive a common anode 7-segment LED display. Each segment consists of one LED and the anodes of all LEDs are connected to +V_{cc} (5V). The cathodes of the LEDs are connected through current-limiting resistors to the appropriate outputs of the decoder/driver. By forward biasing different LEDs, the digits 0 through 9 can be displayed.

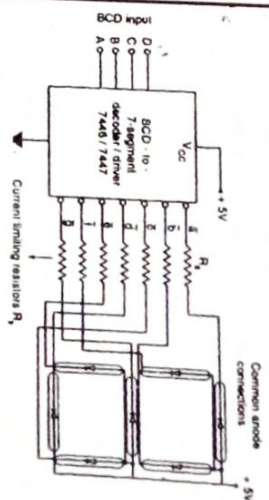


Fig. 5: BCD-to-7-segment decoder driving a common anode 7-segment LED Display

Q.18 (a) Explain full subtractor in detail. Realize it with two half-subtractors. OR [R.T.U. 2013]

Implement a full subtractor using half subtractors. [R.T.U. 2012; Raj. Univ. 2005]

(b) Explain 1:4 demultiplexer. Design 1:8 DEMUX using two 1:4 DEMUX [R.T.U. 2013]

Ans. (a) Half Subtractor: A half subtractor is a combinational circuit which is used to perform subtraction of two bits. It has two inputs, A (minuend) and B (subtrahend) and two outputs D (difference) and B_{out} (borrow out).

The logic symbol for a half subtractor is shown in fig. 1(a). The truth table for half-subtractor is shown in table 1.

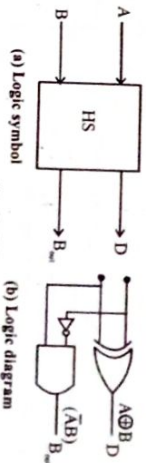


Fig. 1: Half subtractor
 Table 1: Truth Table for Half Subtractor

Inputs			Output
A	B	D	B _{out}
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

From the truth table, it is clear that the difference output is 0 if A = B and 1 if A ≠ B; the borrow output B_{out} is 1 whenever A < B, if A is less than B, then subtraction is done by borrowing 1 from the next higher order bit. Thus, D = A ⊕ B; B_{out} = A̅B

Full Subtractor : A full subtractor is a combinational circuit that performs subtraction involving three bits, namely: minuend bit, subtrahend bit and the borrow from the previous stage. The logic symbol for full subtractor is shown in Fig. 2(a). It has three inputs: A (minuend), B (subtrahend) and B_{in} (borrow from previous stage), and two outputs: D (difference) and B_{out} (borrow out). The truth table for the full subtractor is given in table. The full subtractor can be implemented using two half subtractors and an OR gate as in fig. 2(b).

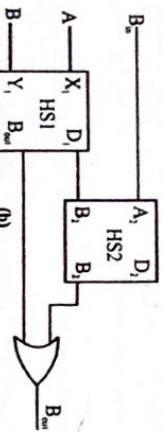
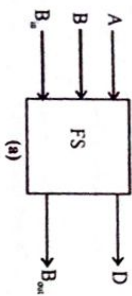


Fig. 2

Table 2 : The Truth Table of Full subtractors

Minuend bit	Subtrahend bit	Borrow in	Difference	Borrow out
A	B	B_{in}	D	B_{out}
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

From table, the sum of product expression for the difference (D) output can be written as:

$$D = \bar{A}\bar{B}B_{in} + \bar{A}B\bar{B}_{in} + A\bar{B}\bar{B}_{in} + AB B_{in}$$

Simplification of the above expression.

$$D = (\bar{A}\bar{B} + AB)B_{in} + (\bar{A}B + A\bar{B})\bar{B}_{in}$$

$$= (\bar{A} \oplus B)B_{in} + (A \oplus B)\bar{B}_{in}$$

$$D = A \oplus B \oplus B_{in}$$

Similarly the sum of product expression for B_{out} can be written from the truth table as:

$$B_{out} = \bar{A}\bar{B}B_{in} + \bar{A}B\bar{B}_{in} + \bar{A}B B_{in} + AB B_{in}$$

$$B_{out} = (\bar{A}\bar{B}B_{in} + \bar{A}B\bar{B}_{in}) + (\bar{A}B B_{in} + AB B_{in})$$

$$= (\bar{A}\bar{B}B_{in} + AB B_{in}) + (\bar{A}B\bar{B}_{in} + AB B_{in})$$

$$= \bar{A}B_{in}(\bar{B} + B) + \bar{A}B(\bar{B}_{in} + B_{in}) + BB_{in}(\bar{A} + A)$$

$$B_{out} = \bar{A}B_{in} + \bar{A}B + BB_{in}$$

K-map Simplification

Next step is to find out expressions for difference and borrow out using K-map:

AB	00	01	11	10
B_{in}	0	1	0	1
1	1	0	1	0

(a) K-map for D = $A \oplus B \oplus B_{in}$

AB	00	01	11	10
B_{in}	0	1	0	0
1	1	1	1	0

(b)

K-map for $B_{out} = \bar{A}B + \bar{A}B_{in} + BB_{in}$

Fig. 3

Logic Circuit

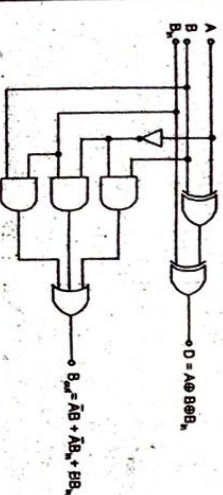


Fig. 4 : Logic Circuit for full subtractor

Implementation of Full Subtractor

A full subtractor can also be made by two half-subtractor and one OR gate. From the Karnaugh map, Expression B_{out} can be written as follows:

$$B_{out} = \bar{A}B + \bar{A}B_{in} + AB B_{in}$$

$$= \bar{A}B + B_{in}(A \oplus B)$$

So,

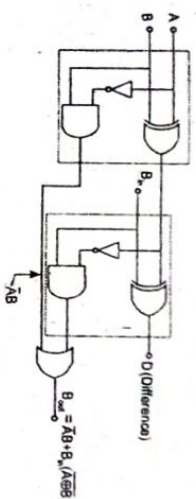


Fig. 5 : Full subtractor using two half subtractor

Ans.(b) 1 x 4 Demultiplexer

Logic circuit of 1 x 4 demultiplexer is shown in fig. There are two select line S_1 and S_0 one data line D and 4 output lines Y_0 to Y_3 .

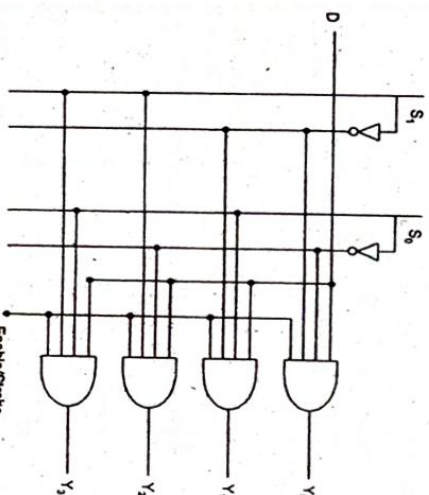


Fig. 1 : 1 x 4 Demultiplexer with enable
Truth table of demultiplexer is shown in table 3.

Table 1 : Truth table of demultiplexer 1 x 4

E	S_1	S_0	Y_3	Y_2	Y_1	Y_0
0	X	X	0	0	0	0
1	0	0	0	0	0	D
1	0	1	0	0	D	0
1	1	0	0	D	0	0
1	1	1	D	0	0	0

If enable (active high) is low, it will disable the multiplexer and output of all AND gate will be zero. Demultiplexer work normally only when $E = 1$.

If $S_1, S_0 = 00$, data D is transmitted to Y_0 . If $S_1, S_0 = 01$, data D is available on Y_1 and so on.

1:8 DEMUX using two 1:4 DEMUX

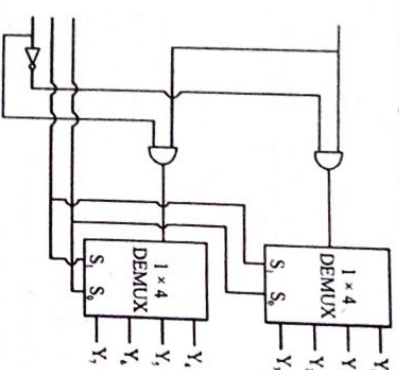


Fig.

Table 2

S_3	S_2	S_1	S_0	Y
0	0	0	0	Y_0
0	0	0	1	Y_1
0	0	1	0	Y_2
0	0	1	1	Y_3
0	1	0	0	Y_4
0	1	0	1	Y_5
0	1	1	0	Y_6
0	1	1	1	Y_7

[Note: By using two 1 x 4 DEMUX, 2-AND gate, 1-NOT gate, we can realize 1 x 8 DEMUX.]

Q.2(a) Design a 4-bit parallel adder using full adders.

(b) Implement the following boolean function using 8:1 MUX

$$F(A, B, C) = \sum m(1, 3, 5, 6) \quad [R.T.U. 2013]$$

Ans.(a) Binary Parallel Adder : A full adder can add two binary bits with carry. By using this Full-Adder we can add two binary numbers of n bit each, let the Binary Number are $A_3A_2A_1A_0$ and $B_3B_2B_1B_0$. So first add A_0 and B_0 then add A_1 and B_1 with previous carry generated (A_0 and B_0). Similarly, add all other bits, as shown in the Fig.1.

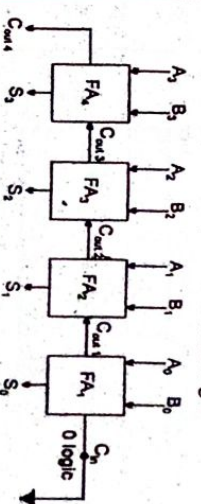


Fig. 1 : 4-bit parallel adder

Thus two bits of A and B are added in parallel, with the carry from previous adder. The input of full adder 1 is given logic '0'. You see that carry out of one adder is connected to the C_{in} of next adder. Such kind of carry propagations called 'ripple carry'.

As explained earlier that first A_0 and B_0 are as sum so S_0 is generated. Then A_1 and B_1 are added with C_{out1} and gives S_1 and C_{out2} , and so on. But the main drawback is that bits $A_3 A_2 A_1 A_0$ and $B_3 B_2 B_1 B_0$ are applied to the adder simultaneously and since an electronic circuit take some time to give result (called propagation delay) you will get correct S_{out2} after time t_{pd} (propagation delay). Then only F_{A2} works correctly. Thus to add 4 bit binary number you have to wait $4 t_{pd}$ time for getting correct sum although propagation delay is in the order of micro seconds (μs). Such propagation delay must be reduced when high-speed circuits are required. The resulting output sum bits are $S_3 S_2 S_1 S_0$.

For example if $A_3 A_2 A_1 A_0 = 1101$ and $B_3 B_2 B_1 B_0 = 0110$, us to have

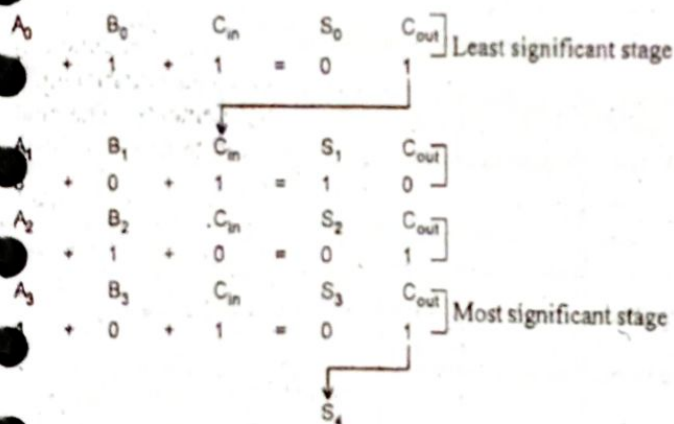


Fig. 2

We see that $S_3 S_2 S_1 S_0 = 0010$. Since the carry out from the most significant stage is a 1. We have an overflow, i.e. the sum (10010) must be expressed in 5 bits.

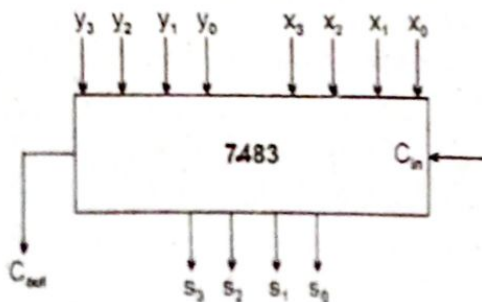


Fig. 3 : Logic symbol of IC 7483-4 bit parallel binary adder.

Ans.(b) Given that

$$F(A, B, C) = \sum m(1, 3, 5, 6)$$

It is the three variable function so it can be solved using 8:1 multiplexer. The given minterm is connected to the $1(+V_{cc})$. The remaining input lines are connected to the ground or 0 potential. The truth table for given function is as:

Truth Table

Inputs			Outputs
A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

The complete expression for output is :

$$Y = \bar{A}\bar{B}C + \bar{A}BC + A\bar{B}C + ABC$$

The 8:1 multiplex circuit implementation is given as :

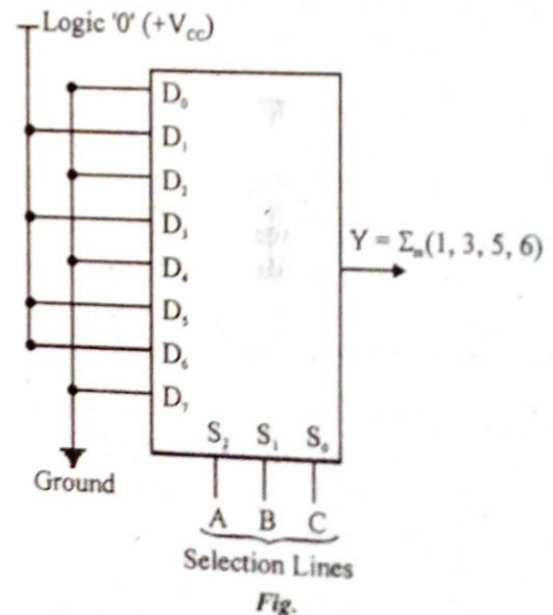


Fig.

Ans. $Q_A(t+1) = Q_A(t) \oplus x$

$Q_B(t+1) = Q_A(t) \oplus Q_B(t)$

$y(t) = \bar{x} + \bar{Q}_A(t) + \bar{Q}_B(t)$

State Table

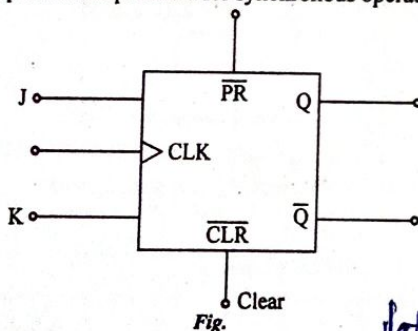
Present State		x	Next State		y
Q_A	Q_B		Q_A	Q_B	
0	0	0	0	0	1
0	0	1	1	0	1
0	1	0	0	1	1
0	1	1	1	1	1
1	0	0	1	1	1
1	0	1	0	1	1
1	1	0	1	0	1
1	1	1	0	0	0

Q.6 What is preset and clear in flip-flops? Explain it.
[R.T.U. 2012; Raj. Univ. 2006]

OR

Define the preset and clear terms with relation to flip flops.
[R.T.U. 2015]

Ans. The preset and clear in flip-flops are active LOW input. An active LOW level at the preset input will SET the flip-flop and an active LOW level on the clear input will RESET it. These asynchronous input as activated by a 0 level, as indicated by small bubbles on flip-flop symbol. Both the preset and clear input must kept HIGH for synchronous operation.



PART-B

Q.7 Explain the procedure for conversion of JK flip flop to RS flip flop.
[R.T.U. 2017]

OR

Convert JK flip flop into RS flip flop. Explain its process also.
[R.T.U. 2018]

Ans. Realization of J-K flip-flop using S-R flip-flop : The J-K flip-flop can be realized using SR flip-flop the block diagram of JK flip-flop using SR flip-flop is shown in figure. Here Y flip-flop is SR flip-flop and X flip-flop is J-K flip-flop which is to be realized.

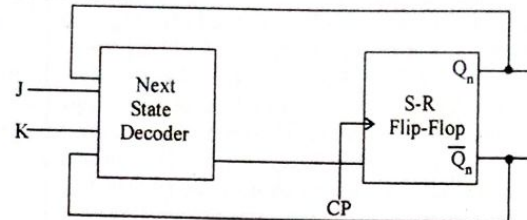


Fig. : Block Diagram of J-K using S-R Flip-flop

Step 1 : To realize the J-K flip-flop, the state table of J-K flip-flop will be made. There will be $2^3 = 8$ combinations of variables Q_n , J and K as shown in table. The next state i.e. Q_{n+1} is obtained by using the characteristic table of J-K flip-flop.

State table of J-K flip-flop

Present State	Input		NS
Q_n	J	K	Q_{n+1}
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Step 2 : Now from the excitation table of SR flip flop, the excitation inputs S and R are to be inserted in the state table mentioned in step 1 to obtain the excitation table shown in table below.

Excitation Table : Refer to Q. 3.

Step 3 : Now we will get the state equation of excitation inputs S and R in terms of inputs J and K and present state Q_n of S-R flip-flop using K-map minimization method. These are given below :

For S

Q_n	J_k			
	00	01	11	10
0	0	0	1	1
1	d	0	0	d

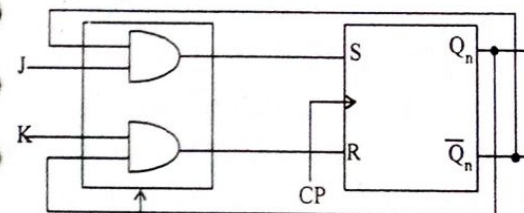
$S = J\bar{Q}_n$

For R

Q_n	J_k			
	00	01	11	10
0	d	d	0	0
1	0	1	1	0

$R = K\bar{Q}_n$

Step 4 : After getting the state equation of S and R, the logic circuit of J-K flip-flop using S-R is made as shown in figure. The next state decoder is replaced by the logic gates according to the input equations of S and R.



Next State Decoder

Fig. : J-K flip-flop using SR flip-flop

Q.8 (a) Draw a logic diagram of clocked S-R flip-flop and obtain its characteristic equation. Also show its excitation table.

(b) Write short note on the following:

- Asynchronous and synchronous counter.
- Sequential and non sequential counter.

[R.T.U. Dec. 2016]

Ans.(a) Clocked RS Flip-Flop

The basic flip-flop as it stands is an asynchronous sequential circuit. By adding gates to the inputs of the basic circuit, the flip-flop can be made to respond to input levels during the occurrence of a clock pulse. The clocked RS flip-flop shown in Fig. (a) consists of a basic NOR flip-flop and two AND gates. The outputs of the two AND gates remain at 0 as long as the clock pulse (abbreviated CP) is 0, regardless of the S and R input values. When the clock pulse goes to 1, information from the S and R input is allowed to reach the basic flip-flop. The set state is reached with $S=1$, $R=0$ and $CP=1$. To change to the clear state, the inputs must be $S=0$, $R=1$ and $CP=1$.

With both $S=1$ and $R=1$, the occurrence of a clock pulse causes both outputs to momentarily go to zero. When the pulse is removed, the state of the flip-flop is indeterminate, i.e. either state may result, depending on whether the set or the reset input of the basic flip-flop remains a 1 longer before the transition to 0 at the end of the pulse. The graphic symbol for the clocked RS flip-flop is shown in Fig. (b). It has three inputs : S, R and CP. The outputs of flip-flops are marked with

(ii) **Sequential and Non-Sequential Counter** : The counters in which the successive states are represented by the binary numbers in odometeric sequence are called sequential counters. In sequential counters the binary equivalent of subsequent states differ by binary 001.

The non-sequential counters may have the states in any random order. The consecutive states are not represented by binary numbers odometeric sequence.

The example of sequential counter is 000, 001, 010, 011, 100, 101, 110 and 111, whereas the example of non-sequential counter is 000, 100, 011, 101, 001, 010, 111 and 110.

Q.9 Design a 4-bit synchronous counter.

[R.T.U. Jan. 2016]

Ans. 4-bit Synchronous Counter : In a synchronous counter each FF is triggered at the same time by a common clock input and so all the FFs change state simultaneously. Four-bit synchronous binary counters are shown in Fig. There are two schemes of flip-flop control in a synchronous counter. One with ripple carry (Fig. 1) where the state change of i th stage is propagated to j th stage ($j > i$) through $(i + 1)$ th, $(i + 2)$ th, ..., j th stage. By contrast, in the other scheme, the state change of i th stage is parallelly propagated to all j th stages—this is referred to as parallel carry. A 4-bit synchronous counter with parallel carry is shown in Fig. (b). All the flip-flops of a counter can be reset by activating clear input. A counter can also be set to a specific state by presetting the corresponding flip-flops to '1'.

An n -bit counter having n flip-flops sequences through 2^n states. Counters can also be designed to have a number of states less than 2^n . A decade counter (modulo ten counter) has ten states; Thus a modulo k counter needs $\lceil \log_2 k \rceil$ number of flip-flops where $\lceil x \rceil$ denotes the smallest integer greater than or equal to x . Counters can be cascaded, to realize higher modulus operation. It realizes a divide by 40,000 counters by using four 4-bit counters and presetting the 16-bit counter hexadecimal value $63CO = 25,536 = 2^{16} - 40,000$. The RCO output of the rightmost counter goes high when the terminal value $2^{16} = 65,536$ is reached. This output is utilized to preset each of the 4-bit counters.

The maximum clock frequency a counter can count depends on propagation delay (T_p) of a flip-flop and propagation delay (T_g) from input to output of a control gating. In addition for ripple (or asynchronous) counters and synchronous counter with ripple carry the maximum clocking frequency will also depend on the number of flip-flop stages n . Table gives the formula for counting the maximum frequency that can be achieved for each type of counter.

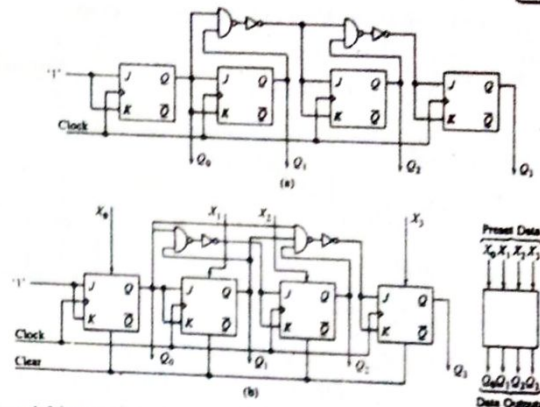


Fig. : 4-bit synchronous modulo 16 ripple/parallel carry counter: (a) Ripple carry; (b) Parallel carry

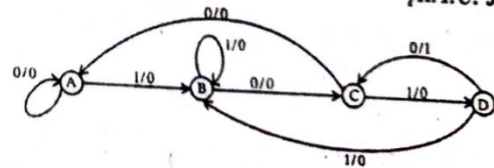
Table : Computation of Maximum Clock Frequency

Counter type	Maximum clock Frequency (f)
Ripple counter	$\frac{1}{f} \geq n(T_p) + T_g$
Synchronous counter with ripple carry	$\frac{1}{f} \geq (T_p) + (n - 2)T_g$
Synchronous counter with parallel carry	$\frac{1}{f} \geq T_p + T_g$

Q.10 (a) Draw the state diagram of logic circuit shown in fig. of Q.3(c).

(b) Design a logic circuit for the state diagram show in fig.

[R.T.U. Jan. 2016]



Ans.(a) $Q_A(t+1) = Q_A(t) \oplus x$

$$Q_B(t+1) = Q_A(t) \oplus Q_B(t)$$

$$y(t) = \bar{x} + \bar{Q}_A(t) + \bar{Q}_B(t)$$

State Table

Present State		x	Next State		y
Q_A	Q_B		Q_A	Q_B	
0	0	0	0	0	1
0	0	1	1	0	1
0	1	0	0	1	1
0	1	1	1	1	1
1	0	0	1	1	1
1	0	1	0	1	1
1	1	0	1	0	1

Expression for T_B

$Q_B Q_A$	00	01	11	10
0	1	0	1	3
1	1	4	5	7

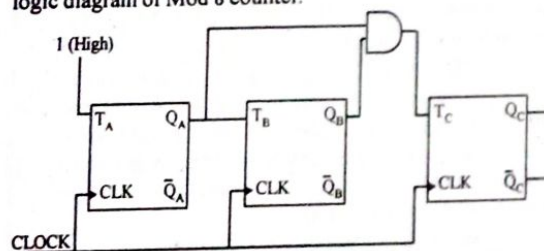
$T_B = Q_A$

Expression for T_C

$Q_B Q_A$	00	01	11	10
0	0	1	3	2
1	4	5	7	6

$T_C = Q_B Q_A$

Step-6: From the above expressions, construct the logic diagram of Mod 8 counter.



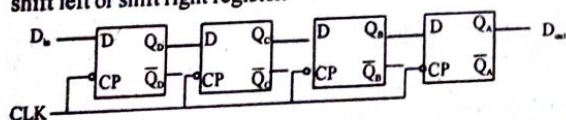
Q.12 Explain the different types of Register used in sequential circuits. [R.T.U. 2014]

Ans. Shift Register : A Shift register consists of an arrangement of a set of flip-flops. It is used to store & shift binary data (0's & 1's)

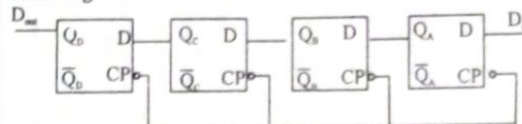
Classification of shift registers:

- (i) Serial in serial out (SISO)
- (ii) Serial in parallel out (SIPO)
- (iii) Parallel in serial out (PISO)
- (iv) Parallel in parallel out (PIPO)

(i) **Serial in Serial out :** This register accepts the data serially on a single i/p line and produces the stored information on its output in serial form. Data can be shifted from left side or right side. Based on the shift of data the register is called shift left or shift right register.



This is a shift register type SISO shift register in which data is shifted towards right serially entered into the left side of the register.



This is a shift left type SISO shift register. Hence the serial data is entered from right side and the output is taken from left most flip flop in serial from.

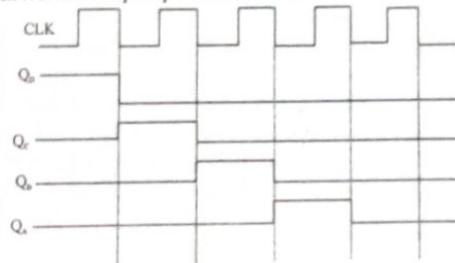


Fig. 1 : Timing Diagram for 4 bit serial in serial out shift register
(ii) Serial in - Parallel out Shift Register

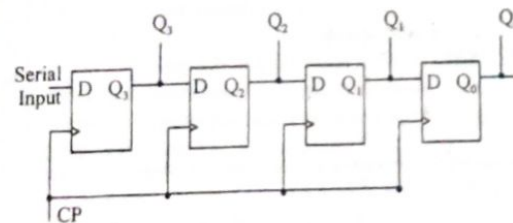


Fig. 2 : Serial in parallel out shift register

Serial-in-parallel-out register is shown in fig.2 which consists of one serial input. The outputs are taken from all the flip-flops simultaneously. Hence, it is known as serial in parallel out shift register. To shift the data out in parallel, data output must be available at the same time and input data is applied bit by bit. Data shifts on occurrence of each clock pulse.

(iii) **Parallel in - Serial out Shift Register**

In this register, the data inputs are applied or stored simultaneously. The output is taken bit by bit i.e. serially. The 4-bit parallel in serial out register is shown in fig.

It has a control input i.e. SHIFT/LOAD which controls the parallel inputs to enter and shifts the data serially. When this input is 0, the AND gates G_1 , G_2 and G_3 get enabled and allow to store the data inputs in the respective flip-flops. When the clock pulse is applied, the flip-flops with $D = 1$ will set and those with $D = 0$ will reset thereby storing all the four bits simultaneously. When it is 1, shift command is activated i.e. gates G_4 , G_5 and G_6 are enabled which shift the data

serially to the right from one stage to the next on each clock pulse. The OR gates allow either the normal shifting operation or parallel data entry operation depending on which AND gates are enabled by the level on the SHIFT/LOAD inputs.

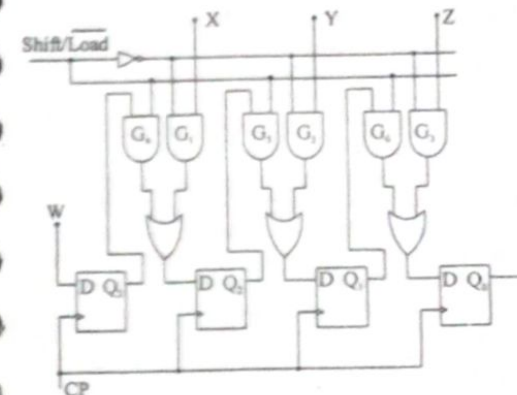


Fig. 3 : 4-bit Parallel in serial out Shift Register

(iv) **Parallel in Parallel out :** All the data appear simultaneously along with all the flip flop inputs and outputs.

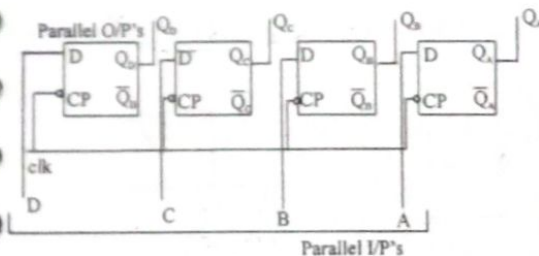


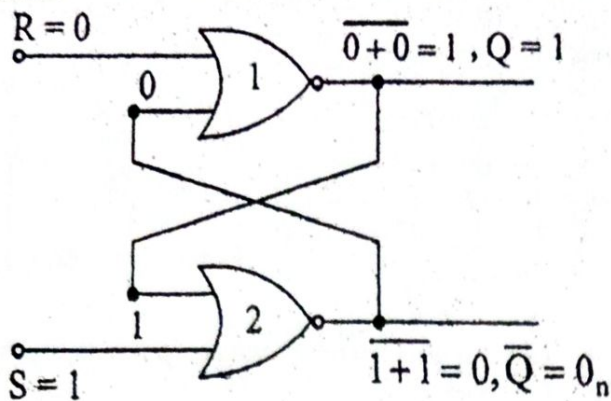
Fig. 4 : PIPO

Q.13 Make short notes on the following :

- (i) D-Flip Flop
- (ii) R-S latch
- (iii) Race around condition

[R.T.U. 2013]

Ans.(i) Delay Flip-Flop : To eliminate the condition of both inputs 1 (one) in SR flip-flop at the same time, the D flip-flop is used. A D flip-flop is derived from R-S flip-flop. This flip flop provides delay between digital circuits, therefore it is named as Delay or D flip-flop. An inverter is inserted between R and S. The input D goes directly to S input and its complement is applied to R input. Because of this inverter, R and S both cannot be 0 or 1 simultaneously and hence condition of Race does not occur.



This signal ($\bar{Q} = 0$) is fed back into the first NOR gate along with the $R = 0$ input causes the output of the first NOR gate i.e. Q to become 1. So, under this condition next states of output are $Q_{n+1} = 1$ and $\bar{Q}_{n+1} = 0$. So this state is called SET state as shown in fig.4.

4. When $S = 1, R = 1$: When both the inputs are 1 then the output of both the NOR gates becomes 0, which is not possible because Q and $\bar{Q}$ are the complement of each other. So they can never be same. It is called the indeterminate or forbidden state. The next state of this condition is taken as don't care output which is never used. This condition leads us to 'RACE' condition of the circuit.

Ans.(iii) Race-Around Condition in J-K Flip-flop : In S-R flip-flop both input are prevented to become 1, since this will lead to ambiguous output Q . This problem is solved by J-K FF. Where J works as S (Set) and K works as R (Reset) and J and K both can be at logic 1. Output Q will get toggle i.e. if Q is 0 it will become 1 and vice-versa.

I/P		O/P
J_n	K_n	Q_{n+1}
0	0	Q_n
1	0	1
0	1	0
1	1	$\bar{Q}_n$

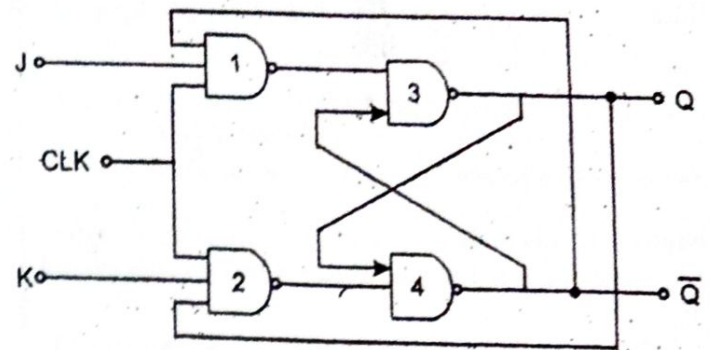
From table assume that the input do not change during the clock pulse ($CLK = 1$), which is not true because of the feedback connection.

Practically when $J = K = 1$, a problem is again created in the flip-flop. We redraw the J-K FF here to understand the problem.

Set both J and K at logic 1, and a clock of on time t_{ON} is applied to the FF. Assume initially $Q = 0$. Now Q will toggle and become 1.

Since a feedback path is available between output and input so the input gate 1, will be (0, 1, 1)

[i.e. $\bar{Q} =$ become 0 after toggle, $J = 1$, CLK is still high].



I/P of gate 2 = 1, 1, 1

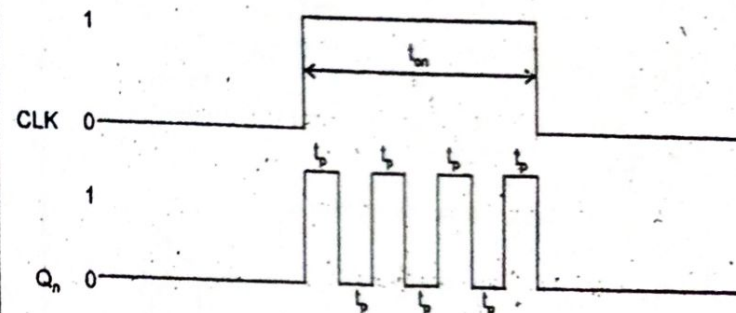
[$\because$ i.e. Q become 1, $K = 1$ CLK is High]

After the propagation time t_{p1} of NAND gate 1, and 2.

O/P of gate 1 = 1;

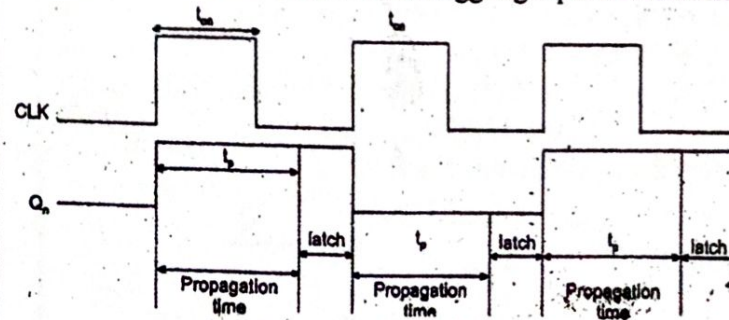
O/P of gate 2 = 0

So, after the propagation time t_{p2} of NAND gate 3, and 4 output Q again toggle and becomes 0. The process of toggling will continue from 0 to 1 and 1 till CLK is high as shown in the Fig.2.



Thus the output Q at the end of t_{ON} [CLK becomes low] can be 0 or 1 i.e. unpredictable. This is called as **Race-around condition** of J-K flip-flop.

This problem can be avoided by making $t_p > t_{ON}$ as shown in fig. 3, shown output can toggle only once and for second time CLK will be low. So no toggling is possible further.



The extra delay in propagation time can be added in order to avoid race-around problem. But since $CLOCK$ frequency can be variable and added extra delay is not a practical solution. The Race-around problem is avoided by using master-slave flip-flop.

State diagram of 4 bit ring counter using D flip-flops

Assuming that the ring counter is initially cleared to 0000, the state diagram for the 4-bit ring counter is drawn as shown in fig.

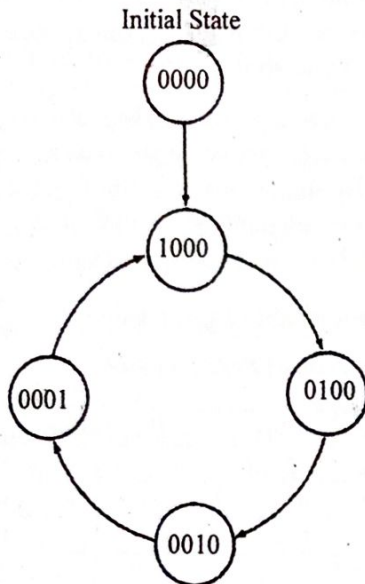


Fig. : State diagram of 4-bit ring counter

Q.21 (a) What is difference between flip-flop and latch ? Explain different types of flip-flops.

(b) Explain the construction and working of Master-slave JK flip-flop. [R.T.U. 2014]

Ans.(a) Difference between Latch and Flip-Flop: Refer to Q.18(a).

Clocked RS Flip Flop : Refer to Q.8(a)

D Flip-Flop: Refer to Q.13(i)

JK Flip-Flop: Refer to Q.18(a)

T-Flip-Flop: Refer to Q.18(a)

Ans.(b) Master-Slave JK Flip-Flop

A master-slave JK flip-flop is constructed by two J-K flip-flop. One flip-flop (first) works as master and second flip-flop works as slave as shown in the Fig.6. The master flip-flop is driven by positive clock while slave flip-flop is driven by negative clock pulse. Therefore when there is positive clock, master is active and works as simple J-K flip-flop and slave remains inactive. When there is negative clock master will become inactive and slave becomes active. Both master and slave are driven by same clock.

This is a level clocked flip-flop. When clock is high any change in J and K affects the output S and R. So the J and K are kept constant during the positive half of clock.

When clock is low the master is inactive and J and K input can be allowed to be changed. The different conditions are set, reset and toggle. The Race around condition is avoided because the feedback from slave to master and the slave being inactive during positive half of clock.

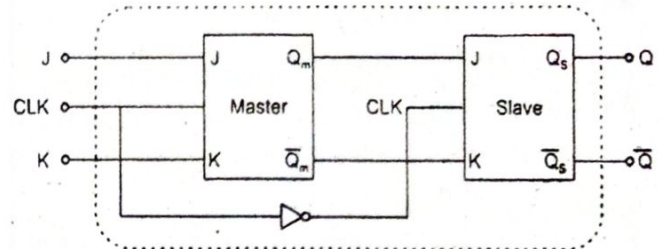


Fig. 1 : Block diagram of master slave J-K flip-flop

A master-slave JK flip-flop using NAND gates is shown below in Fig. 2.

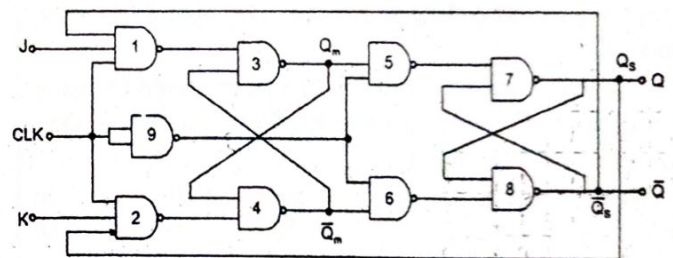


Fig. 2 : Master slave J-K flip-flop using NAND gates only

NAND gates 1, 2, 3 and 4 make master flip-flop and NAND gates 5, 6, 7 and 8 make slave flip-flop. NAND gate 9 works as an inverter $[\bar{A} \cdot \bar{A} = \bar{A}]$ since clock is directly connected to master flip-flop, it will work as simple +ve level flip-flop and since clock is inverted by NAND gate 9, slave is driven by -ve level of the clock [i.e. when CLK is low]

Let us see how master-slave works

(i) **J = 0, K = 0**

When CLK is high

O/P of master = Unchanged (i.e. $Q_m = 0, \bar{Q}_m = 1$)

O/P of slave = Unaffected

When CLK is low

O/P of master = Unaffected

O/P of slave = Unchanged (i.e. $Q_s = 0, \bar{Q}_s = 1$)

(ii) **J = 1, K = 0**

When clock is high

O/P of master = set ($Q_m = 1, \bar{Q}_m = 0$)

O/P of slave = Unaffected

When clock is low

O/P of master = Unaffected

O/P of slave = set ($Q_s = 1, \bar{Q}_s = 0$)

(iii) **J = 0, K = 1**

When clock is high

O/P of master = Reset ($Q_m = 0, \bar{Q}_m = 1$)

When clock is low

O/P of slave = Reset ($J = Q_m, K = \bar{Q}_m$)

$Q_s = 0, \bar{Q}_s = 1$.

(iv) $J = 1, K = 1, Q_s = 0$

When clock is high

O/P of master = toggle ($Q_m = 1, \bar{Q}_m = 0$)

When clock is low

O/P of slave = Set ($J = Q_m, K = \bar{Q}_m$)

i.e. $Q_s = 1$ (i.e. toggle $0 \rightarrow 1$)

The main thing is that although feedback is present between input and output, it is unidirectional due to clock i.e. when clock is high there will be no change in Q_s and hence Race-around problem is removed. When clock is low slave become active and this will toggle to Q_s . So there is only one time toggling can take place and hence race-around condition is removed.

The Race-around condition can be removed by using edge trigger JK flip-flop. So master-slave flip-flop is used when level triggering required.

Realisation of JK flip flop using D flip flop : To construct the JK flip flop using D flip flop, first we will make the block diagram of JK flip flop using D flip flops as shown in fig.3. Here Y is D flip flop and X is JK flip flop which is to be realised.

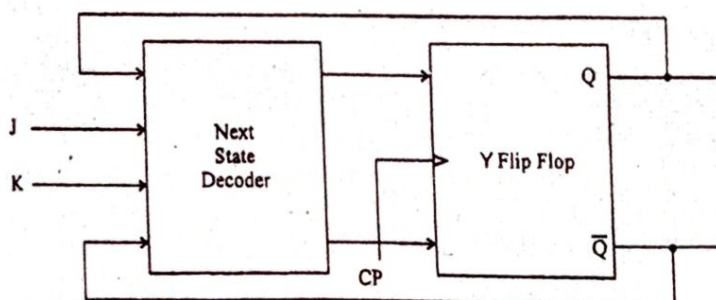


Fig. 3 : Block diagram of JK flip flop using D flip flop

Step 1 : Now we will make the state table of JK flip flop. The state table of JK flip flop is shown in table 1 :

Table 1 : State Table of JK flip flop

PS	Input		NS
Q_n	J	K	Q_{n+1}
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

There are eight combinations of variables of present state Q_n and inputs JK. The next state Q_{n+1} for each combination can be found from the characteristic table of JK flip flop. For example, when the inputs $JK = 0$ are applied, there will be no change in state so $Q_{n+1} = Q_n$. Similarly for other combinations the next state Q_{n+1} can be obtained which are shown in state table above.

Step 2 : Since JK flip-flop is to be realised using D flip flop, therefore, excitation inputs D are to be incorporated in the state table of JK flip flop. For D flip flop the next state Q_{n+1} is equivalent to the input of D flip flop i.e.

$$Q_{n+1} = D$$

This is shown in table 2 given below :

Table 2 : Excitation Table

PS	Input		NS	Excitation Inputs	
Q_n	J	K	Q_{n+1}		D
0	0	0	0		0
0	0	1	0		0
0	1	0	1		1
0	1	1	1		1
1	0	0	1		1
1	0	1	0		0
1	1	0	1		1
1	1	1	0		0

Step 3 : Now the state equation of Input D is obtained using K-map as follows :

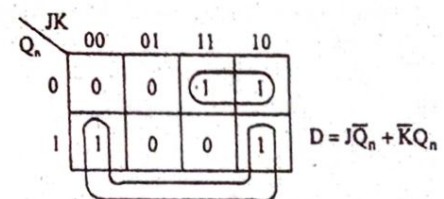


Fig. 4

Step 4 : The next state decoder in JK flip flop shown in fig.10 is replaced by logic gates which are connected in accordance with the state equation of input D obtained in step 3. The JK flip flop using D flip flop is shown in fig. 5.

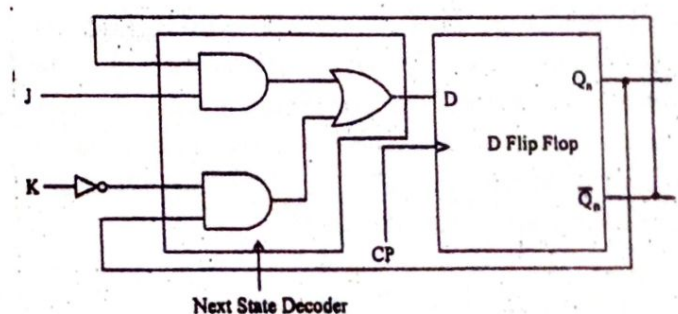


Fig. 5 : JK flip flop using D flip flop

DSA.2

- High-level programming languages, such as Pascal, C, etc., that provides support for recursion use stack for book keeping. Remember, in each recursive call, there is a need to save the current values of parameters, local variables and the return address (the address where the control has to return from the call).

Parenthesis Checker : Parenthesis Checker is a program that checks whether a mathematical expression is properly parenthesised. We will consider three sets of grouping symbols: the standard parenthesis "()", braces "{}", and

B.Tech. (III Sem.) C.S. Solved Papers

brackets "[]". For an input expression, it verifies that for each left parenthesis, brace or bracket, there is a corresponding closing symbol and the symbols are appropriately nested.

Some examples of valid inputs are

Valid Input
()
{ }
{ () }
{ ({ }) }
{ ({ () }) }

Some examples of invalid inputs are

Invalid Input
{ ({ }
{ ({ (}
{ ({ ({ }
{ ({ ({ (}

Inside parenthesis (braces or brackets), there can be any valid arithmetic expression. If the no. of symbols are even then it is valid input otherwise it is invalid input.

PREVIOUS YEARS QUESTIONS

PART-A

Prob.1 What are the applications of stack? [R.T.U. 2018]

Sol. Applications of Stacks : As mentioned earlier, stack is one of the most commonly used data structures. Some of its applications are :

- Stacks are used to pass parameters between functions. On a call to a function, the parameters and local variables are stored on a stack.
- High-level programming languages, such as Pascal, C, etc., that provides support for recursion use stack for book keeping. Remember, in each recursive call, there is a need to save the current values of parameters, local variables and the return address (the address where the control has to return from the call).

Prob.2 Write down the recursive algorithm to solve tower of Hanoi problem? [R.T.U. 2018]

Sol. Procedure : Tower (N, BEG, AUX, END)

This procedure gives a recursive solution to the Tower of Hanoi problem for N disks.

- If N = 1, then:
 - Write: BEG → END.
 - Return.

[End of If structure.]
- [Move N - 1 disks from peg BEG to peg AUX.]
Call TOWER(N - 1, BEG, END, AUX).

- Write: BEG → END.
- [Move N - 1 disks from peg AUX to peg END.]
Call TOWER(N - 1, AUX, BEG, END).
- Return.

So the Tower of Hanoi problem can be solved by using the recursion.

Prob.3 Convert following infix expression into postfix notation :

$$A + B - (C + D) / E * F - (G + h) / I$$

[R.T.U. Dec. 2016]

Sol. A + B - (C + D) / E * F - (G + h) / I

Symbol scanned	Stack	Postfix Expression
A	(	A
+	(+	A
B	(+	AB
-	(+ -	AB
(	(+ - (	AB
C	(+ - (	ABC
+	(+ - (+	ABC
D	(+ - (+	ABCD
)	(+ -	ABCD +
/	(+ - /	ABCD +
E	(+ - /	AB + D + E
*	(+ - *	AB + D + E /
F	(+ - *	AB + D + E / F
-	(+ - -	AB + D + E / F *
(	(+ - - (	AB + D + E / F *
G	(+ - - (	AB + D + E / F * G
+	(+ - - (+	AB + D + E / F * G
h	(+ - - (+	AB + D + E / F * G h

Data Structures and Algorithms

) (+ -- AB + D + E / F * G h +
/ (* -- / AB + D + E / F * G h +
! (* -- / AB + D + E / F * G h + I
) AB + D + E / F * G h + I / -- *
Postfix expression : AB + D + E / F * G h + I / -- *

Prob.4 Transform the following expression in Postfix Notation :

$$A * (B + D) / E - F * (G + H / K) \quad [R.T.U. Jan 2016]$$

Sol. A * (B + D) / E - F * (G + H / K)

$$= A * (BD +) / E - F * (G + HK /)$$

$$= ABD + * / E - F * GHK / +$$

$$= ABD + * / E - FG HK / + *$$

$$= ABD + * / EFG HK / + * -$$

$$= ABD + * EFG HK / + * - /$$

Prob.5 Convert $(a + (b * (c - d) + (e) - (f * g)))$ into postfix notation. [R.T.U. 2014]

Sol. Convert $(a + (b * (c - d) + (e) - (f * g)))$

$$p = (a + (b * (c - d) + (e) - (f * g)))$$

$$p = (a + (b * (cd -) + (e) - (fg *)))$$

$$p = (a + (bcd - * + (e) - (fg *)))$$

$$p = (a + (bcd - * e + fg * -))$$

$$p = abcd - * e + fg * - +$$

PART-B

Prob.6 Translate infix expression into equivalent postfix expression:

$$(a) (A - B) * (D/E)$$

$$(b) (A + B \uparrow D) / (E - F) + G \quad [R.T.U. 2018]$$

Sol.(a) $(A - B) * (D/E)$

$$(AB -) * (DE /)$$

$$\rightarrow AB - DE / *$$

(b) $(A + B \uparrow D) / (E - F) + G$

$$(A + (BD \uparrow)) / (EF -) + G$$

$$(ABD \uparrow +) (EF -) / + G$$

$$ABD \uparrow + EF - / G +$$

Prob. 3 What is STACK? Write algorithms to insert an element in STACK and delete an element from STACK with example. [R.T.U. Dec. 2016]

Sol. Stack : A stack is a list of elements in which an element may be inserted or deleted only at one end, called the top of the stack. This means, in particular, that elements are removed from a stack in the reverse order of that in which they were inserted into the stack.

Special terminology is used for two basic operations associated with stacks:

- (a) "Push" is the term used to insert an element into a stack.
- (b) "Pop" is the term used to delete an element from stack.

We emphasize that these terms are used only with stacks, not with other data structures.

Example

Suppose the following 6 elements are pushed, in order, onto an empty stack:

AAA, BBB, CCC, DDD, EEE, FFF

Figure shows three ways of picturing such a stack. For notational convenience, we will frequently designate the stack by writing:

STACK: AAA, BBB, CCC, DDD, EEE, FFF

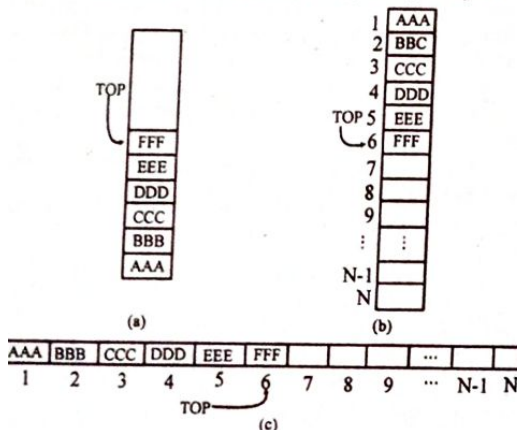


Fig. 1 : Stack representation

The implication is that the right-most element is the top element. We emphasize that, regardless of the way a stack described, its underlying property is that insertions and deletions can occur only at the top of the stack. This means

EEE cannot be deleted before FFF is deleted, DDD cannot be deleted before EEE and FFF are deleted, and so on. Consequently, the elements may be popped from the stack only in the reverse order of that in which they were pushed onto the stack.

Postponed Decisions

Stacks are frequently used to indicate the order of the processing of data when certain steps of the processing must be postponed until other conditions are fulfilled. This is illustrated as follows.

Suppose that while processing some project A we are required to move on to project B, whose completion is required in order to complete project A. Then we place the folder containing the data of A onto a stack, as pictured in figure 2(a), and begin to process B. However, suppose that while processing B we are led to project C, for the same reason. Then we place B on the stack above A, as pictured in figure 2(b), and begin to process C. Furthermore, suppose that while processing C we are likewise led to project D. Then we place C on the stack above B, as pictured in figure 2(c), and begin to process D.

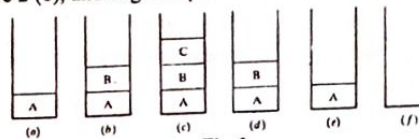


Fig. 2

On the other hand, suppose we are able to complete the processing of project D. Then the only project we may continue to process is project C, which is on top of the stack. Hence we remove folder C from the stack, leaving the stack as pictured in figure 2(d), and continue to process C. Similarly, after completing the processing of C, we remove folder B from the stack, leaving the stack as pictured in figure 2(e), and continue to process B. Finally, after completing the processing of B, we remove the last folder, A, from the stack, leaving the empty stack pictured in figure 2(f), and continue the processing of our original project A.

Observe that, at each stage of the above processing, the stack automatically maintains the order that is required to complete the processing. An important example of such a processing in computer science is where A is a main program and B, C and D are subprograms called in the order given.

Array Representation of Stacks

Stacks may be represented in the computer in various ways, usually by means of a one-way list or a linear array. Unless otherwise stated or implied, each of our stacks will be maintained by a linear array STACK; a pointer variable TOP, which contains the location of the top element of the stack; and a variable MAXSTK which gives the maximum number of elements that can be held by the stack. The

condition $TOP = 0$ or $TOP = NULL$ will indicate that the stack is empty.

Fig.3 pictures such an array representation of a stack. (For notational convenience, the array is drawn horizontally rather than vertically.) Since $TOP = 3$, the stack has three elements, XXX, YYY and ZZZ; and since $MAXSTK = 8$, there are rooms for 5 more items in the stack.

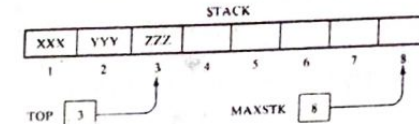


Fig. 3

The operation of adding (pushing) an item onto a stack and the operation of removing (popping) an item from a stack may be implemented, respectively, by the following procedures, called PUSH and POP. In executing the procedure PUSH, one must first test, whether there is room in the stack for the new item; if not, then we have the condition known as overflow. Analogously, in executing the procedure POP, one must first test, whether there is an element in the stack to be deleted; if not, then we have the condition known as underflow.

Procedure 1:

PUSH(STACK, TOP, MAXSTK, ITEM)

This procedure pushes an ITEM onto a stack.

1. [Stack already filled?]
If $TOP = MAXSTK$, then: Print: OVERFLOW, and Return.
2. Set $TOP := TOP + 1$. [Increases TOP by 1.]
3. Set $STACK[TOP] := ITEM$. [Inserts ITEM in new TOP position.]
4. Return.

Procedure 2:

POP(STACK, TOP, ITEM)

This procedure deletes the top element of STACK and assigns it to the variable ITEM.

1. [Stack has an item to be removed?]
If $TOP = 0$, then: Print: UNDERFLOW, and Return.
2. Set $ITEM = STACK[TOP]$. [Assigns TOP element to ITEM.]
3. Set $TOP := TOP - 1$. [Decreases TOP by 1.]
4. Return.

Frequently, TOP and MAXSTK are global variables; hence the procedures may be called using only

PUSH(STACK, ITEM) and POP(STACK, ITEM)

Respectively, we note that the value of TOP is changed before the insertion in PUSH but the value of TOP is changed after the deletion in POP.

DSA.6

- If the top of the stack is another operand continue looping (as stated above, in looping constraint)
 - Else if the top of the stack if an operator or the operand stack is empty, push the result and exit the loop.
4. Go back to step 1 (Loop)

PART-C

Prob.10(a) Convert following expression in its equivalent postfix expression.

- $A * B + C$ [R.T.U. 2015]
- $A * B^C + D$ [R.T.U. 2017, 2015]
- $A * (B + C * D) + E$ [R.T.U. 2017, 2015]

(b) Explain the concept of recursion using stack with a suitable example. What are the difficulties in dealing with infix expressions? [R.T.U. 2015]

OR

Define the concept of recursion using stack using suitable examples. What are the difficulties in dealing with infix expression? [R.T.U. 2017]

Sol.(a)(i) $A * B + C$

The order in which the operators appear is not reversed. When the '+' is read, it has lower precedence than the '*', so the '*' must be printed first.

We will show this in a table with three columns. The first will show the symbol currently being read. The second will show what is on the stack and the third will show the current contents of the postfix string. The stack will be written from left to right with the 'bottom' of the stack to the left.

	current symbol	operator stack	postfix string
1	A		A
2	*	*	A
3	B	*	A B
4	+	+	A B * {pop and print the '*' before pushing the '+'}
5	C	+	A B * C
6			A B * C +

The rule used in lines 1, 3 and 5 is to print an operand when it is read. The rule for line 2 is to push an operator onto the stack if it is empty. The rule for line 4 is if the operator on the top of the stack has higher precedence than the one being read, pop and print the one on top and then push the new operator on. The rule for line 6 is that when the end of the

B.Tech. (III Sem.) C.S. Solved Papers

expression has been reached, pop the operators on the stack one at a time and print them.

(ii) $A * B^C + D$

Here both the exponentiation and the multiplication must be done before the addition.

	current symbol	operator stack	postfix string
1	A		A
2	*	*	A
3	B	*	A B
4	^	* ^	A B
5	C	* ^	A B C
6	+	+	A B C ^ *
7	D	+	A B C ^ * D
8			A B C ^ * D +

When the '+' is encountered in line 6, it is first compared to the '^' on top of the stack. Since it has lower precedence, the '^' is popped and printed. But instead of pushing the '+' sign onto the stack now, we must compare it with the new top of the stack, the '*'. Since the operator also has higher precedence than the '+', it also must be popped and printed. Now the stack is empty, so the '+' can be pushed onto the stack.

(iii) $A * (B + C * D) + E$

	current symbol	operator stack	postfix string
1	A		A
2	*	*	A
3	(	*(	A
4	B	*(	A B
5	+	*(+	A B
6	C	*(+	A B C
7	*	*(+*	A B C
8	D	*(+*	A B C D
9	)	*	A B C D * +
10	+	+	A B C D * + *
11	E	+	A B C D * + * E
12			A B C D * + * E +

Sol.(b) A typical implementation of recursion involves the use of a stack. Each call to a subprogram generates an activation record to contain its return address, parameters, and local variables. The activation records are accessed in a Last-In-First-Out manner. Thus a stack is the choice of data structure. Recursion can be supported by systems and languages that use dynamic storage allocation. The procedure parameters and local variables are not bound to addresses until an activation record is created at run time. Thus multiple copies of the intermediate values of recursive calls to the program can be supported, as new activation records are created for them.

Data Structures and Algorithms

With static storage allocation, in contrast, a single location is reserved at compile time for each parameter and local variable of a procedure. There is no place to store intermediate values calculated by repeated nested calls to the same procedure. Therefore, systems and languages with only static storage allocation cannot support recursion.

Factorial by recursive method

```
public static long factorial (long n) {
    If (n <= 1)
        return 1;
    else
        return n * factorial (n-1);
}
```

The first time we call method factorial, its stack frame includes a local variable storing the value n. Method factorial () recursively calls itself to compute (n-1)!, which pushes a new frame on the stack. In turn, this recursive invocation calls itself to compute (n-2)!, etc. The chain of recursive invocations and thus the run-time stack, only grows up to size n, because calling factorial (1) returns 1 immediately without invoking itself recursively. The run-time stack allows for method factorial () to exist simultaneously in several active frames (as many as n at some point). Each frame stores the value of its parameter n as well as the value to be returned. Eventually, when the first recursive call terminates, it returns (n-1)!, which is then multiplied by n to compute n! for the original call of the factorial method.

Difficulties in dealing with infix expressions

- If the procedure is very long. The input is read far too many. Also once an expression is read once it makes it very difficult to repeat the process.
- This method can be extremely difficult to use when there are multiple parenthesis.
- It makes it difficult to check if an expression is correct [Every left parenthesis is closed].

Prob.11 What do you mean by Tower of Hanoi problem? Explain with suitable example. [R.T.U. Jan 2016]

OR

Explain tower of Hanoi problem. Explain using suitable diagram and example. [R.T.U. 2017]

OR

Can the Tower of Hanoi problem be solved using recursion? [R.T.U. 2013]

Sol. Tower of Hanoi : Suppose three pegs, labeled A, B and C are given and suppose on peg A there are placed a finite number n of disks with decreasing size. This is pictured in figure 1 for the case n = 6. The object of the game is to move the disks from peg A to peg C using peg B as an auxiliary. The rules of the game are as follows:

- Only one disk may be moved at a time. Specifically, only the top disk on any peg may be moved to any other peg.
- At no time can a larger disk be placed on a smaller disk.

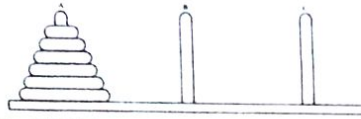


Fig. 1: Initial setup of Towers of Hanoi with $n = 6$

Sometimes we will write $X \rightarrow Y$ to denote the instruction "Move top disk from peg X to peg Y ," where X and Y may be any of the three pegs.

The solution to the Tower of Hanoi problem for $n = 3$ appears in figure (2). Observe that it consists of the following seven moves:

- $n = 3$:
- Move top disk from peg A to peg C.
 - Move top disk from peg A to peg B.
 - Move top disk from peg C to peg B.
 - Move top disk from peg A to peg C.
 - Move top disk from peg B to peg A.
 - Move top disk from peg B to peg C.
 - Move top disk from peg A to peg C.

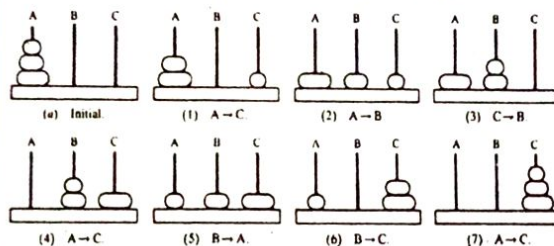


Fig. : 2

In other words,

For $n = 3$: $A \rightarrow C$, $A \rightarrow B$, $C \rightarrow B$, $A \rightarrow C$, $B \rightarrow A$, $B \rightarrow C$, $A \rightarrow C$

For completeness, we also give the solution to the Tower of Hanoi problem

for $n = 1$ and $n = 2$:

$n = 1$: $A \rightarrow C$

$n = 2$: $A \rightarrow B$, $A \rightarrow C$, $B \rightarrow C$

Note that $n = 1$ uses only one move and that $n = 2$ uses three moves.

Rather than finding a separate solution for each n , we use the technique of recursion to develop a general solution. First we observe that the solution to the Tower of Hanoi problem for $n > 1$ disks may be reduced to the following subproblems:

- Move the top $n - 1$ disks from peg A to peg B.
- Move the top disk from peg A to peg C: $A \rightarrow C$.
- Move the top $n - 1$ disks from peg B to peg C.

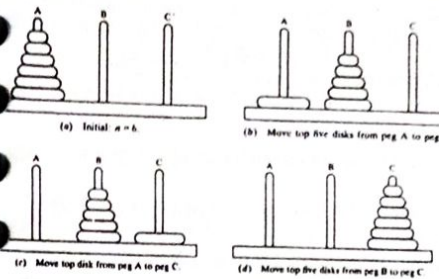


Fig. 3

This reduction is illustrated in figure 3 for $n = 6$. That is, first we move the top five disks from peg A to peg B, then we move the large disk from peg A to peg C, and then we move the top five disks from peg B to peg C.

Let us now introduce the general notation

$TOWER(N, BEG, AUX, END)$

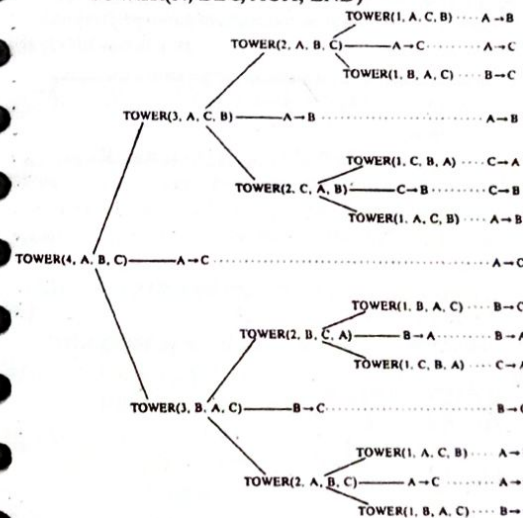


Fig. 4 : Recursive solution to Tower of Hanoi problem for $n = 4$

To denote a procedure which moves the top n disks from the initial peg BEG to the final peg END using the peg AUX as an auxiliary when $n = 1$, we have the following obvious solution:

$TOWER(1, BEG, AUX, END)$

It consists of the single instruction, $BEG \rightarrow END$

Furthermore, as discussed above, when $n > 1$, the solution may be reduced to the solution of the following three subproblems:

- $TOWER(N - 1, BEG, END, AUX)$
- $TOWER(1, BEG, AUX, END)$ or $BEG \rightarrow END$
- $TOWER(N - 1, AUX, BEG, END)$

Observe that each of these three subproblems may be solved directly or essentially the same as the original problem

using fewer disks. Accordingly, this reduction process does yield a recursive solution to the Towers of Hanoi problem.

Fig. 4 contains a schematic diagram of the above recursive solution for

$TOWER(4, A, B, C)$

Observe that the recursive solution for $n = 4$ disks consists of the following 15 moves:

$A \rightarrow B$, $A \rightarrow C$, $B \rightarrow C$, $A \rightarrow B$, $C \rightarrow A$, $C \rightarrow B$, $A \rightarrow B$, $A \rightarrow C$, $B \rightarrow C$, $B \rightarrow A$, $C \rightarrow A$, $B \rightarrow C$, $A \rightarrow B$, $A \rightarrow C$, $B \rightarrow C$

In general, this recursive solution requires $f(n) = 2^n - 1$ moves for n disks.

We summarize our investigation with the following formally written procedure.

Procedure : Refer to Prob.2.

Prob.12 (a) Derive an algorithm which converts infix expression to its postfix expression. Also write algorithm which evaluates the postfix expression.

(b) Write the insertion and deletion algorithms for Stack.

[R.T.U. 2015]

Sol.(a) Algorithm

Prefix (Q,P)

Suppose Q is an arithmetic expression written in infix notation. This algorithm finds the equivalent prefix expression.

- Push "(" onto STACK and add "(" to end of Q .
- Scan Q from right to left and repeat step 3 to 6 for each element of Q until the STACK is empty.
- If an operand is encountered add it to P .
- If a right parenthesis is encountered push it onto STACK.

5. If an operator $\otimes$ is encountered, then

- Repeatedly pop from STACK and add to P each operator (on the top of STACK) which has the same precedence as or higher precedence than $\otimes$.

(b) Add $\otimes$ to STACK.

[End of if structure]

6. If a left parenthesis is encountered, then.

- Repeatedly pop from STACK and add to P each operator (on the top of STACK) until a right parenthesis is encountered.

(b) Remove the right parenthesis [Do not add the right parenthesis to P]

[End of if structure]

[End of step 2 loop]

7. Exit

PART-A

Prob.1 What are the differences between normal queue and circular queue?
[R.T.U. 2018]

Sol. Queue : A queue is a linear list of elements in which deletions can take place only at one end, called the FRONT, and insertions can take place only at the other end, called the rear. The terms "FRONT" and "rear" are used in describing a linear list only when it is implemented as a queue.

Queues are also called first-in first-out (FIFO) lists since the first element in a queue will be the first element out of the queue. In other words, the order in which elements enter a queue is the order in which they leave. This contrasts with stacks, which are last-in first-out (LIFO) lists.

Circular Queue : In circular queue, when the rear pointer reaches the upper bound of array, it is again set to one or zero.

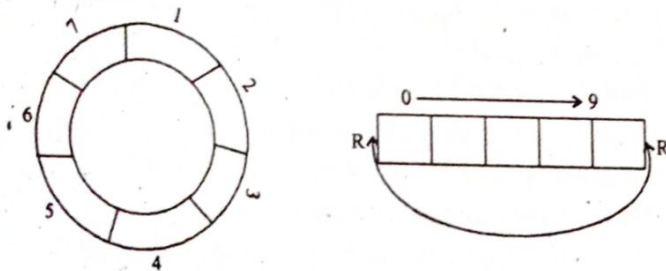


Fig.

Prob.2 Write down the advantages and disadvantages of singly linked list?
[R.T.U. 2018]

Sol. Singly Linked List : Advantages and disadvantages of singly linked list are followings-

Advantage :

1. Insertions and deletions can be done easily.
2. It does not need movement of elements for insertion and deletion.
3. It space is not wasted as we can get space according to our requirements.
4. Its size is not fixed.
5. It can be extended or reduced according to requirements.
6. Elements may or may not be stored in consecutive memory available, even then we can store the data in computer.
7. It is less expensive.

Disadvantage :

1. It requires more space as pointers are also stored with information.
2. Different amount of time is required to access each element.
3. If we have to go to a particular element then we have to go through all those elements that come before that element.
4. We can not traverse it from last & only from the beginning.
5. It is not easy to sort the elements stored in the linear linked list.

Prob.3 Write an algorithm to delete an element from a circular queue.
[R.T.U. Dec. 2016]

Sol. Circular Queue : Refer to Prob.1.

Deletion Operation :

DELETE CQ (QUEUE, N, FRONT, REAR)

Step 1: [check for underflow condition]

if FRONT = 0, then :

write : CQ is UNDERFLOW and exit.

[End of if structure]

Step 2: [Remove the FRONT element from CQ]

set VALUE := CQ[FRONT]

Step 3: [check whether circular queue becomes empty after deletion]

if FRONT = REAR, then

set REAR := 0

FRONT := 0

Step 4: Else if

FRONT = N, then :

set FRONT := 1

else

set FRONT := FRONT + 1

[End of if structure]

Step 5: [Process the deleted value]

write : Deleted Value = VALUE
and exit.

Prob.4 Explain polynomial representation using linked list with an example.
[R.T.U. Dec. 2016]

OR

How can a polynomial such as $5x^4 - 3x^2 + 9x - 11$ be represented by a linked list?
[R.T.U. Jan 2016]

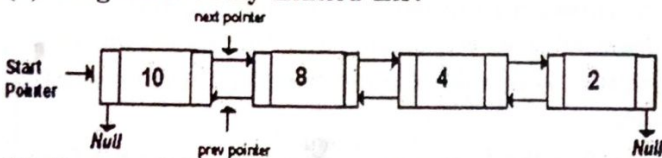
Sol. Representation of Polynomials using Linked List:
A polynomial can be thought of as an ordered list of non-zero

Explanation

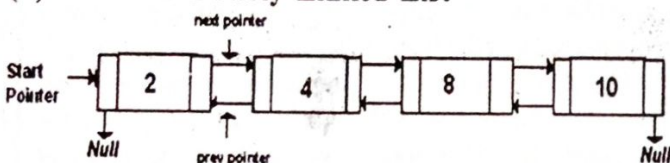
Every Node in a doubly linked list has next and prev pointer. Do the linear traversal of the linked list and keep swapping the next and prev pointers. At the end, make the last pointer as the head of the list. The above function is a simple method for reversing a Doubly Linked List. All we need to do is swap prev and next pointers for all nodes, change prev of the head (or start) and change the head pointer in the end.

Example

(a) Original Doubly Linked List



(b) Reversed Doubly Linked List



Prob.6 Explain comparison of array and linked list as data structure. [R.T.U. 2011]

Sol. Difference between Array and Linked List

S.No.	Array	Linked List
1.	Array is a linear data structure.	Linked list are linear or non-linear.
2.	Array data can be accessed using subscript.	In linked list data can be accessed using info part.
3.	Array have continuous memory location.	Linked list are not in continuous memory location.
4.	Array are fixed in size so resources must be consumed in advance.	Linked list are quite flexible can grow to any size up to resource limit and each node can be of different size.
5.	Inserting and deleting an item in the array is difficult.	Insertion and deletion of a node in linked list is quite easy.
6.	To access a particular element is fast because its location in memory can be determined directly.	In linked list its tough to find any node because previous node must be examined and have to follow pointers.
7.	In Array there is a problem of memory storage and wastage because memory is assigned into advance. Hence follow Static Memory Allocation.	In linked list we assign memory at run time so no problem as array.

8.	Array are classified as 1-D, 2-D & 3 D.	Linked list can be circular or doubly linked.
9.	Array don't store extra memory.	Linked list stores a bit extra memory due to pointer at the last element.
10.	It follows static memory allocation.	It follows dynamic memory allocation.

Prob.7 Write a short note on reversing a list or string.

Sol. Reversing the String

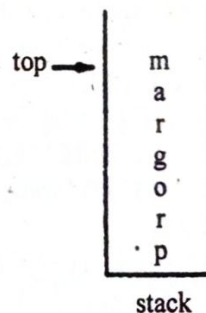
To reverse a string stack can be used. The simple mechanism is to push all the characters of a string onto the stack and then pop all the characters from the stack and print them. For example

If the input string is

p r o g r a m \0

Then push all the characters onto the stack till '\0' is not encountered.

Now if we pop each character from the stack and print it we get,



m a r g o r p

Fig.

which is a reversed string.

PART-B

Prob.8 Write down the algorithm for insertion of a node in the middle of doubly linked list. [R.T.U. 2018]

Prob.9 Write an algorithm to implement stack operation using linked list. [R.T.U. 2011]

OR

Write down the algorithm to implement stack using linked list. [R.T.U. 2018]

Sol. Stack Implementation : The linked representation of a stack, commonly termed linked stack is a stack that is implemented using a single linked list. The INFO fields of the nodes hold the elements of the stack and the LINK fields hold pointers to the neighbouring element in the stack. The START pointer of the linked list behaves as the TOP pointer variable of the stack and the null pointer of the last node in the list signals the bottom of stack. Fig. 1 illustrates the linked representation of the stack.

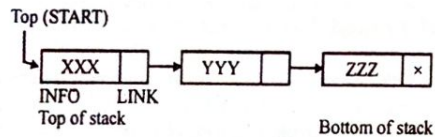
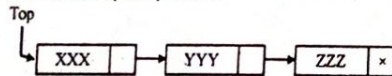


Fig. 1

A push operation into STACK is accomplished by inserting a node into the front or start of the list and a pop operation is undertaken by deleting the node pointed by the START pointer. Fig. 2 and Fig. 3 illustrate the push and pop operation on the linked stack STACK shown in Fig. 1.

Push 'WWW' into STACK
STACK before push operation :



STACK after push operation :

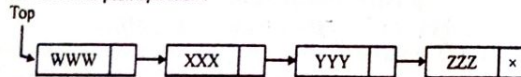
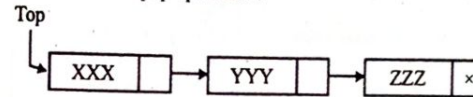


Fig. 2

The array representation of stack calls for the maintenance of a variable MAXSTK which gives the maximum number of elements that can be held by the stack. Also, it calls for the checking of OVERFLOW in the case of push operation ($TOP = MAXSTK$) and UNDERFLOW in the case of pop operation ($TOP = 0$). In contrast, the linked representation of stack is free of these requirements. There is no limitation on the capacity of the linked stack and hence it can support as many push operation (insertion of nodes) as the free-storage list (the AVAIL list) can support. This dispense with the need to maintain the MASSTK variable and consequently on checking of OVERFLOW of the linked stack during a push operation.

Pop from STACK

STACK before pop operation :



STACK after pop operation :

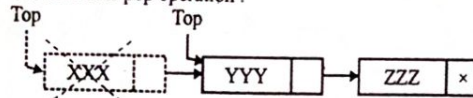


Fig. 3

Algorithms

Stack - Insertion

PUSH - LINKSTACK (INFO, LINK, TOP, AVAIL, ITEM)

This procedure pushes an ITEM into a linked stack.

1. [Available space?] if AVAIL = NULL, then write OVERFLOW and Exit.
2. [Remove first node from AVAIL list]
set NEW := AVAIL and AVAIL := LINK[AVAIL].
3. Set INFO[NEW] := ITEM [copies ITEM into new node].
4. Set LINK[NEW] := TOP [new node points to the original top node in the stack].
5. Set TOP = NEW [Reset TOP to point to the new node at the top of the stack].
6. Exit.

Stack-Deletion

POP_LINKSTACK (INFO, LINK, TOP, AVAIL, ITEM)

This procedure deletes the top element of a linked stack and assigns it to the variable ITEM.

1. [Stack has an item to be removed]
IF TOP = NULL then write : Underflow and exit.
2. Set ITEM := INFO[TOP] [copies the top element of stage into ITEM].
3. Set TEMP := TOP and TOP = LINK[TOP]
[Remember the old value of the Top pointer in TEMP and reset.
TOP to point to the next element in the stack]
4. [Return deleted node to the AVAIL list]
set LINK[TEMP] = AVAIL and AVAIL = TEMP.
5. Exit.

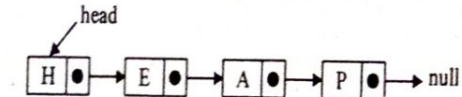
In the above example, INFO field of the nodes hold the elements of the stack and the LINK fields hold pointers to the neighbouring element in the stack.

Prob.10 Discuss concepts of Head Node in linked lists in brief with suitable example. [R.T.U. Jan 2016]

OR

Using suitable diagram explain the concept of Head Node in linked lists. [R.T.U. 2017]

Sol. Head Node : A linked list is a linear data structure where each element is a separate object.



Each element (we will call it a node) of a list is comprising of two items - the data and a reference to the next node. The last node has a reference to null. The entry point into a linked list is called the head of the list. It should be noted that head is not a separate node, but the reference to the first node. If the list is empty then the head is a null reference.

Therefore head node mark the reference to the first node and without the presence of the head node we cannot begin any operation in the linked list.

The head node is a node which resides at the beginning of the linked list. Sometime such extra node needs to be kept at the front of the list. This node basically does not represent any data of the linked list. But it may contain some useful information about linked list such as total number of nodes in the list, address of last node or some specific distinguishing information.

Head Node in Singly Links List :

In this type of linked list only one link is used to point to next element and this list is circular means that the last node's link field points to the first or head node.

Head node in a singly link list have been shown in following fig.

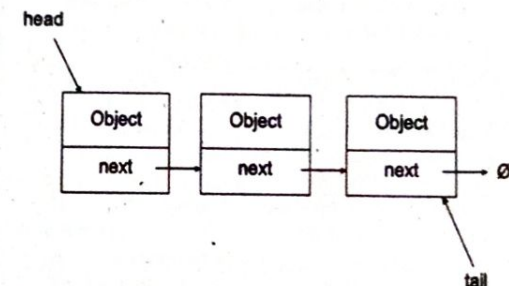


Fig.

Head Node in Doubly Link List :

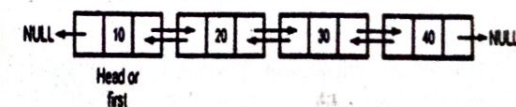


Fig.

This list called doubly because each node has two pointer previous and next pointers. The previous pointer points to previous node and next pointer points to next node. Only in

case of head node the previous pointer is obviously NULL and last node's next pointer points to NULL. This list is a linear one.

Head Node in Circular Doubly Linked List :

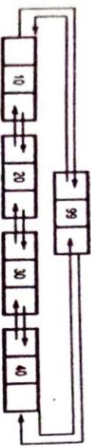


Fig.

In circular doubly linked list the previous pointer of first node and the next pointer of last node is pointed to head node. Head node is a special node which may have any dummy data or it may have some useful information such as total number of a nodes in the list which may be used to simplify the algorithms carrying various operation on the list.

Various algorithms for operating upon the head node are as follows:

Deletion :

Removal of an element at the head of a singly linked list is relatively easy.

Algorithm removeFirst()

if (head == null) then

Indicate an error: the list is empty

tmp = head

head = head.getNext()

tmp.setNext(null)

size = size - 1

Insertion:

Insert a new node at the head of the list is straightforward. The main idea is that we create a new node, set its next link to refer to the current head, and then set head to point to the new node.

Algorithm addFirst(String newData):

create a new node v containing newData

v.setNext(head)

head = v

size = size + 1

Prob.11 Write down an algorithm for insertion and deletion operation performed on the dequeue.

OR

Write an algorithm for dequeue and circular queue.

OR

What are the application of queue? Explain the algorithm for insertion and deletion in circular queue.

Sol. Dequeue : A dequeue is a linear list in which elements can be added or removed at either end but not in the middle. The term dequeue is a contraction of the name double ended queue. Two types of dequeue :

(1) **Input Restricted Dequeue :** Dequeue which allows insertion only at one end but allows deletion at both ends.

(2) **Output Restricted Dequeue :** Dequeue which allows insertion at both ends but allows deletion only at one end.

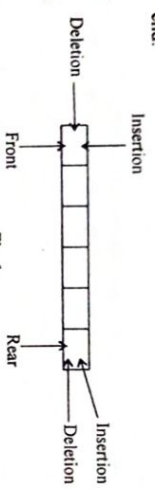


Fig. 1

Application of Queue : The major application of queue is in the operating system and in network.

(1) **Operating system :** Often maintains a queue of processes that are ready to execute or that are waiting for a particular event to occur. All the processes that are submitted to your processor or CPU are first taken by in a queue and then processed as various algorithm.

(2) **Network :** Similarly on a network if a number of users wants to access a resource their request is taken in a queue and then processed.

(3) **Print server :** Maintains a queue of print jobs.

(4) **Disk server :** Maintains a queue of disk input/output request.

(5) **Scheduler (e.g. in operating system) :** Maintains a queue of processes waiting a slice of machine time.

Algorithm for Dequeue :

(1) **Insertion operation :**

DOINSERT (QUEUE, N, FRONT, REAR, DATA).

Step 1: [check for overflow condition]

if REAR = N and FRONT = 1, then :

write : DQ is OVERFLOW and exit.

End of if structure.

Step 2: [check the value of FRONT]

if FRONT > 1, then

set DQ[FRONT] = DATA

else

write : Insertion is not possible from rear end and exit.

End of if structure]

Step 3: Exit

(2) **Deletion operation :**

DQ Delete (N, QUEUE, FRONT, REAR)

Step 1: [check for underflow condition]

if REAR = FRONT - 1, then :

write : DQ is UNDERFLOW and exit

[End of if structure]

Data Structures and Algorithms

Step 2: [check if rear is greater than FRONT]

if REAR > = FRONT, then :

set i = FRONT

Repeat while i < = REAR

write : CQ[i]

set i = i + 1

[End of loop]

Else

set i = FRONT

Repeat while i < = size

write : CQ[i]

set i = i + 1

[End of loop]

set i = 1

Repeat while i < = REAR

write : CQ[i]

set i = i + 1

[End of loop]

[End of if structure]

Step 4: Exit

[End of if structure]

Step 2: [check if rear is greater than FRONT]

if REAR > = FRONT, then :

set i = FRONT

Repeat while i < = REAR

write : CQ[i]

set i = i + 1

[End of loop]

Else

set i = FRONT

Repeat while i < = size

write : CQ[i]

set i = i + 1

[End of loop]

set i = 1

Repeat while i < = REAR

write : CQ[i]

set i = i + 1

[End of loop]

[End of if structure]

Step 4: Exit

[End of if structure]

Step 2: [check if rear is greater than FRONT]

if REAR > = FRONT, then :

set i = FRONT

Repeat while i < = REAR

write : CQ[i]

set i = i + 1

[End of loop]

Else

set i = FRONT

Repeat while i < = size

write : CQ[i]

set i = i + 1

[End of loop]

set i = 1

Repeat while i < = REAR

write : CQ[i]

Prob.12 What is dequeue? Write down the algorithms for the insertion and deletion operations performed on dequeue.

[R.T.U. Dec 2016]

Sol. Double-Ended Queue : A double-ended queue is an abstract data type similar to an simple queue, it allows you to insert and delete from both sides means items can be added or deleted from the front or rear end.

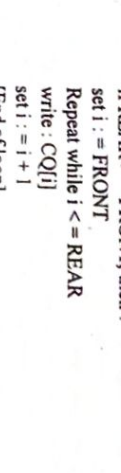


Fig.

Algorithm for insertion at rear end

Step -1: [check for overflow]

if (rear == MAX)

print ("Queue is Overflow");

return;

Step -2: [insert element]

else

rear = rear + 1;

q[rear] = no;

[set rear and front pointer]

If rear = 0

rear = 1;

if front = 0

front = 1;

Step 3: return

Prob.20 Explain Round Robin scheduling.

Sol. Round-robin (RR) is one of the algorithms employed by process and network schedulers in computing. As the term is generally used, time slices (also known as time quanta) are assigned to each process in equal portions and in circular order, handling all processes without priority (also known as cyclic executive). Round-robin scheduling is simple, easy to implement, and starvation-free. Round-robin scheduling can also be applied to other scheduling problems, such as data packet scheduling in computer networks. It is a operating system concept.

The name of the algorithm comes from the round-robin principle known from other fields, where each person takes an equal share of something in turn.

Process Scheduling

To schedule processes fairly, a round-robin scheduler generally employs time-sharing, giving each job a time slot or quantum (its allowance of CPU time), and interrupting the job if it is not completed by then. The job is resumed next time a time slot is assigned to that process. If the process terminates or changes its state to waiting during its attributed time quantum, the scheduler selects the first process in the ready queue to execute. In the absence of time-sharing, or if the quanta were large relative to the sizes of the jobs, a process that produced large jobs would be favoured over other processes.

Round-robin algorithm is a pre-emptive algorithm as the scheduler forces the process out of the CPU once the time quota expires.

For example, if the time slot is 100 milliseconds, and job1 takes a total time of 250 ms to complete, the round-robin scheduler will suspend the job after 100 ms and give other jobs their time on the CPU. Once the other jobs have had their equal share (100 ms each), job1 will get another allocation of CPU time and the cycle will repeat. This process continues until the job finishes and need no more time on the CPU.

Job = Total time to complete 250 ms (quantum 100 ms)

1. First allocation = 100 ms
2. Second allocation = 100 ms
3. Third allocation = 100 ms
4. Total PCU time of job1 = 250 ms

Consider the following table with the arrival time and execute time of the process with the quantum time of 100ms to understand the round-robin scheduling.

Process name	Arrival time	Execute time
P0	0	250
P1	50	170
P2	130	75
P3	190	100
P4	210	130
P5	350	50

Execute time		Round Robin Scheduling
0	P0	P0 arrives and the gets processed
50	P0 P1	P1 arrives and waits for quantum to expires
100	P1 P0	quantum time 100ms expires, so P0 is forced out of CPU and P1 gets processed
130	P1 P0 P2	P2 arrives
190	P1 P0 P2 P3	P3 arrives
200	P0 P2 P3 P1	next 100ms expires, so P1 is forced out of CPU and P0 gets processed
210	P0 P2 P3 P1 P4	P4 arrives
300	P2 P3 P1 P4 P0	Next 100ms expires, so P0 is forced out of CPU and P2 gets processed
350	P2 P3 P1 P4 P0 P5	P5 arrives
375	P3 P1 P4 P0 P5	P2 gets completed, sp P3 gets processed
475	P1 P4 P0 P5	P3 gets completed, so P1 gets processed
545	P4 P0 P5	P1 gets completed, so P4 gets processed
645	P0 P5 P4	Quantum time 100ms expired, so P4 is forced out of CPU and P0 gets processed

695	P5 P4	P0 gets completed, so P5 gets processed
745	P4	P5 gets completed, so P4 gets processed
775		P4 gets completed

Prob.21 Explain advantages and disadvantages of

- (i) Singly Linked List
- (ii) Doubly Linked List
- (iii) Circular Linked List

Sol. (i) Singly Linked List : Refer to Prob.2.

(ii) Doubly Linked List

Advantage :

1. We can traverse in both direction i.e. from starting to end & as well as from end to starting.
2. It is easy to reverse the linked list.
3. If we are at a node, we can go at any node. But in linked list, it is not possible to reach the previous node.

Disadvantage :

1. It requires more space per space per node because extra field is required for pointer to previous node.
2. Insertion and deletion take more time than linear linked list because more pointer operations are required than linear linked list.

(iii) Circular Linked List

Advantages :

1. If we are at a node, then we can go to any node. But in linear linked list it is not possible to go to previous node.
2. It saves time when we have to go to the first node from the last node. It can be done in single step because there is no need to traverse the in between nodes. But in double linked list, we will have to go through in between nodes.

Disadvantages :

1. It is not easy to reverse the linked list.
2. If proper care is not taken, then the problem of infinite loop can occur.
3. If we at a node and go back to the previous node, then we can not do it in single step. Instead we have to complete the entire circle by going through the in between nodes and then we will reach the required node.

DSAR

3. [Find new value of FRONT]

If FRONT = REAR, then: [Queue has only one element to start]

Set FRONT = NULL and REAR = NULL.

Else if FRONT = N, then:

Set FRONT = 1.

Else:

Set FRONT = FRONT + 1.

[End of If structure]

4. Return.

Prob 25 (a) Write an algorithm to insert an element into a single linked list. [R.T.U. 2013, R.U. 2006]

(b) A doubly linked list can be made circularly by setting the values of links in the first and last nodes appropriately. Discuss the advantage and disadvantages of circular doubly linked list in doing the various list operations. [R.T.U. 2013]

Sol. (a) Insertion into Single Linked List : A node may be inserted into a linked list in 3 ways.

First one is to insert a node at the beginning of the list. Second one is to insert a node after the node with a given location and third one is to insert a node into a sorted list.

We assume that the linked list is in memory in the form LIST[INFO, LINK, START, AVAIL] and that the variable ITEM contains the new information to be added to the list. So all of above 3 algorithms includes following steps:

- Checking to see if space is available in the AVAIL list. If not, that is if AVAIL = NULL, then the algorithm will print the message OVERFLOW.
- Removing the first node from the AVAIL list. Using the variable NEW to keep track of the location of the new node, means

NEW = AVAIL.

AVAIL = LINK [AVAIL]

- Copy the new information into the new node: INFO [NEW] = ITEM

Diagrammatically

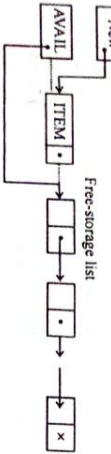


Fig. 1

Data Structures and Algorithms

Diagrammatically :

(1) Case I : Insertion at the beginning of a list:

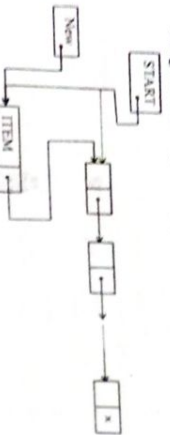


Fig. 2 : Insertion at the beginning of a list

Algorithm

INSFIRST (INFO, LINK, START, AVAIL, ITEM)

[This algorithm inserts a node at the beginning of a list].

1. [OVERFLOW?] If AVAIL = NULL, then,

Write: OVERFLOW and Exit.

2. [Remove first node from AVAIL list]

Set NEW = AVAIL and AVAIL = LINK [AVAIL]

Set INFO [NEW] = ITEM [Copies new data into new node]

Set LINK [NEW] = START [New node points to original first node]

Set START = NEW [Changes START so it points to the new node]

6. Exit.

(2) Case II : Insertion after a given node :

Suppose we are given the location, where ITEM is to be inserted

If LOC = NULL then new node N is inserted as the first node in the LIST else node N points to a node which was originally followed by node of location LOC. We have

LINK [NEW] = LINK [LOC]

LINK [LOC] = NEW

Algorithm :

INSLOC (INFO, LINK, START, AVAIL, LOC, ITEM)

[This Algorithm inserts ITEM so that ITEM follows the node with the location LOC or inserts ITEM as the first node when LOC = NULL.]

1. [OVERFLOW?] If AVAIL = NULL, then: Write: OVERFLOW and Exit.

2. [Remove first node from AVAIL list]

Set NEW = AVAIL and AVAIL = LINK [AVAIL]

Set INFO [NEW] = ITEM [Copies new data into new node]

4. If LOC = NULL, then: [Insert as first node]

Set LINK [NEW] = START and

START = NEW.

Else: [Insert after node with location LOC]

Set LINK [NEW] = LINK [LOC] and

LINK [LOC] = NEW.

[End of If structure]

5. Exit.

Data Structures and Algorithms

Diagrammatically :

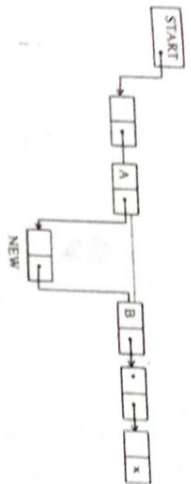


Fig. 3 : Insertion into a sorted linked list

(3) Case III : Insertion into a sorted linked list :

Suppose ITEM is to be inserted into a sorted linked list, then ITEM must be inserted between nodes A and B so that

INFO (A) < ITEM ≤ INFO (B).

First we find location of node A, which is the last node in LIST whose value is less than ITEM.

Traverse LIST by using a pointer variable PTR and comparing ITEM with INFO [PTR] at each node.

Keep track of the location of the preceding node by using a pointer variable SAVE as shown below in figure.

Thus SAVE and PTR are updated by using assignments

SAVE = PTR and

PTR = LINK [PTR]

The traversing continues as long as INFO [PTR]

> ITEM, or in other words traversing stops as soon as ITEM ≤ INFO [PTR]. Then PTR points to node B, so SAVE will contain the location of the node A.

In the case where ITEM < INFO [START], so LOC = NULL, are treated separately, since they do not involve the variable SAVE.

Fig. 4

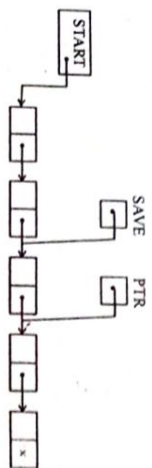


Fig. 4

Procedure :

FIND A (INFO, LINKS, START, ITEM, LOC)

This procedure finds the location LOC of the last node in a sorted list such that INFO [LOC] < ITEM, or sets LOC = NULL.

1. [List empty?] If START = NULL, then: Set LOC = NULL and Return.

2. [Special case?] If ITEM < INFO [START], then: Set LOC = NULL and Return.

3. Set SAVE = START and PTR = LINK [START].

[Initializes pointers]

4. Repeat steps 5 and 6 while PTR ≠ NULL.

5. If ITEM < INFO [PTR], then: Set LOC = SAVE and Return.

[End of If structure]

6. Set SAVE = PTR and PTR = LINK [PTR]

[Update pointers]

[End of Step 4 loop].

Set LOC = SAVE

7. Return

8. Return

Algorithm :

INSERT (INFO, LINK, AVAIL, START, ITEM)

[This algorithm inserts ITEM into a sorted linked list].

1. Call FIND A (INFO, LINK, START, ITEM, LOC).

2. Call INS LOC (INFO, LINK, START, AVAIL, LOC, ITEM).

3. Exit.

Sol. (b) In doubly linked list with null pointer are good enough to the extension of the singly linked list. And the circular doubly linked lists are the variation of doubly linked lists. The list is shown in the Fig. is a doubly linked circular list. In this list, the left link of the left most node contains the address of the right most node and the right link of the right most nodes contains the address of the left most node.

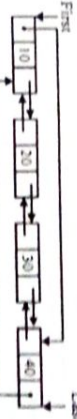


Fig. 1 : Circular doubly linked list

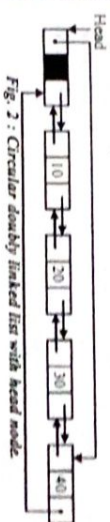


Fig. 2 : Circular doubly linked list with head node

Fig. 3 : Head node

A circular doubly linked list is one which has both the successor and predecessor pointer in circular manner. The aim of considering circular doubly linked list is to simplify the insertion and deletion operations performed on doubly linked list. Consider the situation of an empty list. In this situation the list will never be empty because of the presence of a special node called head node. That always remains in the list. A circular doubly linked list with a head node is shown in the Fig. 2. In this list, the left link of a header node contains address of the last node and right link of last node contains address of the header node. An empty list with a header node can be represented as shown in the Fig. 3. Where the left link and right link of a header node points to itself.

Whatever, the problems we solved in the singly linked list can also be solved using doubly linked list. The option is

DSA.34

left to the programmer to implement all the problems using doubly linked list and circular doubly linked list.

Advantages:

- (i) Can easily keep going "past" ends.
- (ii) Can visit all elements from any starting point.
- (iii) Can never fall off the end of a list.

Disadvantage : code must avoid infinite loop.

Prob.26 *Write detailed note on multiple stack representation along with its program.*

Sol. Multiple Stacks

In a single array any number of stacks can be adjusted. And push and pop operations on each individual stack can be performed. The following Fig. 1 shows how multiple stacks can be stored in a single dimensional array.

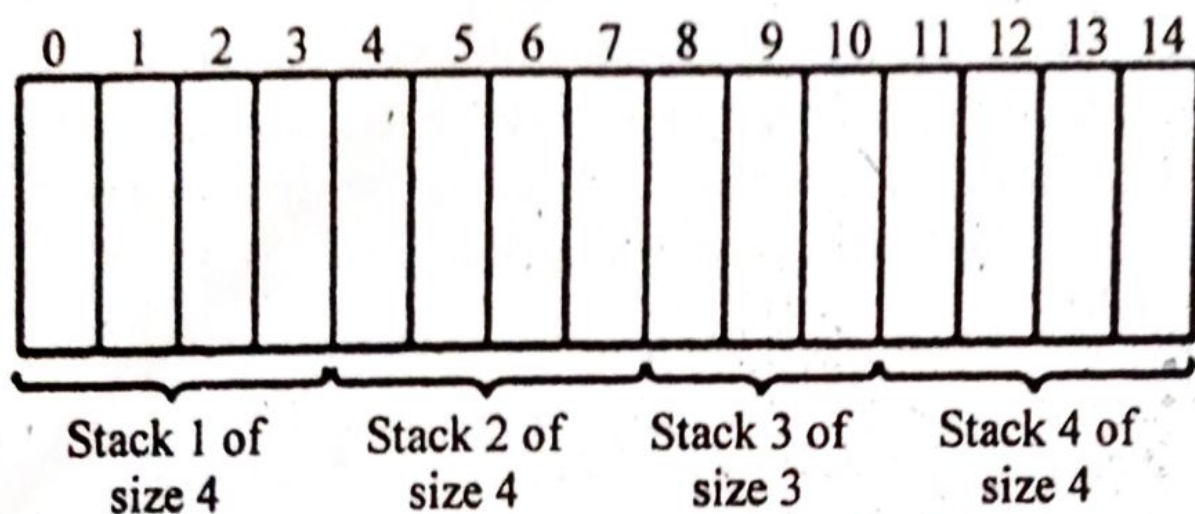


Fig. 1 : Multiple stacks in one dimensional array

In the Fig.1 stacks are adjusted in a single dimensional array of size 15. Each stack can be of any size. The only one

PREVIOUS YEARS QUESTIONS

PART-A

Prob.1 Write down the algorithm of binary search.

[R.T.U. 2018]

Sol. Algorithm for Binary Search :

LIST : Represents the list of element

N : Represents the no. of element in the list.

ELEMENT : Represents the value to be search in the list.

Step 1: [Initialization]

set LOW : = 1

HIGH : = N

FLAG : = 1

Step 2: Repeat through step 4 while

LOW $\leq$ HIGH

Step 3: Set MID : = $\frac{LOW + HIGH}{2}$

Step 4: If ELEMENT < LIST [MID], then :

set HIGH : = MID - 1

ELSE if ELEMENT > LIST [MID], then :

set LOW : = MID + 1

LIST : FLAG = 0

write : 'ELEMENT found at location' MID.

[End of if structure]

[End of step 2 loop]

Step 5: If FLAG = 1, then :

write : 'Element not found'

Step 6: Exit.

Prob.2 What is difference between an internal sorting and external sorting?

[R.T.U. Jan. 2016]

OR

Compare Internal sorting and External sorting.

[R.T.U. 2017]

Sol. Differences between Internal & External Sorting :

Internal sort	External sort
1. Internal sorting can reside in main memory.	External sorting uses secondary memory.
2. It takes input only which can be fit into its memory e.g. it takes small input.	External sorting can takes as much as large input.
3. For internal sort we need to code a program by specifying the sort criteria and that the program need to be compiled & linked.	But in case of external sort there is no need of coding the program. Moreover we can change the sort criteria just by changing sort parameters.
4. Less efficient.	More efficient.

Prob.3 Apply binary search to find 123 in a list.

49, 98, 101, 123, 149, 194, 199, 211, 240, 286, 840, 930 (12 data) [R.T.U. 2014]

Sol. 49, 98, 101, 123, 149, 194, 199, 211, 240, 286, 840, 930
Find 123

Suppose item = 123 which is need to be searched.

(1) BEG = 1 and END = 12

Mid value = 6

49, 98, 101, 123, 149, 194, 199, 211, 240, 286, 840, 930

(2) Since $123 < 194$, END has its value changed by
END = Mid - 1 = 5

Hence, Mid = $(1+5)/2=3$

49, 98, 101, 123, 149, 194, 199, 211, 240, 286, 840, 930

(3) Since $123 > 101$ and scan towards right of 101 wherefore next element between the mid [value] and END [value i.e., 101 and 149 is found to be 123]

Result : item = 123 search.

PART-B

Prob.4 Sort the following elements using quick sorting algorithm.

< 2, 10, 9, 6, 1, 15, 5, 11 >

[R.T.U. 2018]

Sol. Quick sort :

2	10	9	6	1	15	5	11
---	----	---	---	---	----	---	----

Take 2 as a pivot value:

Step 1 : search right to left for small value compare to pivot value if value is found then swipe the element:

1	10	9	6	2	15	5	11
---	----	---	---	---	----	---	----

1 is small than 2 exchange the position

Step2 : Now check left to right for bigger value then pivot value:

1	2	9	6	10	15	5	11
---	---	---	---	----	----	---	----

Step 3 : No small value found then pivot value now search stop and array divided in two part:

1	2						
9	6	10	15	5	11		

Step 4 : Now take 9 as a pivot value for array 2 and repeat searching. The final solution of quick sort is:

1	2	5	6	9	10	11	15
---	---	---	---	---	----	----	----

Prob.5 Prove that Heap sort, Merge sort, Quick sort. Takes $\Omega(n \log n)$ tie in the worst case.

[R.T.U. 2017, 2015]

Sol. Complexity of Heap Sort : To analyze the heap sort, note that a complete binary tree with n nodes (where n is one less than the power of two) has $\log(n+1)$ levels. Thus, if each element in the array were a leaf, requiring it to be filtered through the entire tree, both while creating and adjusting the heap, the sort would be $O(n \log n)$.

In the average case, the heap sort is not as efficient as the quick sort. However, heap sort is far superior to quick sort in the worst case, in fact, heap sort complexity remains $O(n \log n)$ in the worst case.

Merge Sort : One natural method of sorting is to sort by repeated merging. Our analysis of mergesort is as follows:

1. The time to merge a pair of lists to get an n element list in $O(n)$ is since each recursive call "retires" and element to the result list and each element of the result gets retired only once.
2. Thus, the time to merge pairs on a list of lists, the summed total length of which is n , is $O(n)$, since from (1) the time to merge a single pair is proportional to the number of elements in the result.
3. The time to use repeat on a list on n 1-element lists is the number of times repeat is called recursively times $O(n)$, from (2).
4. The number of times repeat is called is $O(\log n)$, since each call encounters a list which is about half as long as the previous call.
5. The time to mergesort an N element list is $O(n) +$ time to repeat an N element list, since mapping an n element list is $O(n)$. From (3) and (4), the time to repeat is $O(n \log n)$.

Therefore the overall time to mergesort an n element list is $O(n \log n)$. unfortunately, this is only

Prob.10 Write an algorithm for bubble sort. Sort the following list by four algorithm.

73, 49, 53, 12, 48, 10, 84, 70, 20, 16 [R.T.U. 2013]

Sol. Bubble Sort Algorithm: In this let n numbers are there. Sorting A refers to the operation of rearranging the element of A so that they are in increasing order. so

$$A[1] < A[2] < A[3] < \dots < A[N]$$

Algorithm:

Bubble sort : In bubble sort each element is compared with its adjacent element. If the first element is larger than the second one then the position of the elements is interchanged, otherwise it is not changed. Then next element is compared with its adjacent element and the same process is repeated for all the elements in the array. During the next pass the same process is repeated

Algorithm : (Bubble Sort) BUBBLE (DATA, N)

Here DATA is an array with N elements. This algorithm sorts the elements in DATA.

1. Repeat Steps 2 and 3 for $K = 1$ to $N - 1$.
2. Set $PTR := 1$. [initializes pass pointer PTR.]
3. Repeat while $PTR \leq N - K$: [Executes pass.]
 - (a) If $DATA[PTR] > DATA[PTR + 1]$, then:

Interchange $DATA[PTR]$ and $DATA[PTR + 1]$.

[End of If structure.]
 - (b) Set $PTR := PTR + 1$.

[End of inner loop.]

[End of Step 1 outer loop.]
4. Exit.

Given list:

73, 49, 53, 12, 48, 10, 84, 70, 20, 16,

Using Bubble sort Algorithm

Pass 1 73, 49, 53, 12, 48, 10, 84, 70, 20, 16,
 49, 73, 53, 12, 48, 10, 84, 70, 20, 16
 49, 53, 73, 12, 48, 10, 84, 70, 20, 16
 49, 53, 12, 73, 48, 10, 84, 70, 20, 16
 49, 53, 12, 48, 73, 10, 84, 70, 20, 16
 49, 53, 12, 48, 10, 73, 84, 70, 20, 16
 49, 53, 12, 48, 10, 73, 84, 70, 20, 16
 49, 53, 12, 48, 10, 73, 70, 84, 20, 16
 49, 53, 12, 48, 10, 73, 70, 20, 84, 16
 49, 53, 12, 48, 10, 73, 70, 20, 16, 84
 Pass 2 49, 12, 53, 48, 10, 73, 70, 20, 16, 84
 49, 12, 48, 53, 10, 73, 70, 20, 16, 84
 49, 12, 48, 10, 53, 73, 70, 20, 16, 84
 49, 12, 48, 10, 53, 73, 70, 20, 16, 84
 49, 12, 48, 10, 53, 70, 73, 20, 16, 84
 49, 12, 48, 10, 53, 70, 20, 73, 16, 84
 49, 12, 48, 10, 53, 70, 20, 16, 73, 84

Pass 3 12, 49, 48, 10, 53, 70, 20, 16, 73, 84
 12, 48, 49, 10, 53, 70, 20, 16, 73, 84
 12, 48, 10, 49, 53, 70, 20, 26, 73, 84
 12, 48, 10, 49, 53, 20, 70, 16, 73, 84
 12, 48, 10, 49, 53, 20, 16, 70, 73, 84
 Pass 4 12, 48, 10, 49, 53, 20, 16, 70, 73, 84
 12, 10, 48, 49, 50, 20, 16, 70, 73, 84
 12, 10, 48, 49, 20, 50, 16, 70, 73, 84
 12, 10, 48, 49, 20, 16, 50, 70, 73, 84
 Pass 5 10, 12, 48, 49, 20, 16, 50, 70, 73, 84
 10, 12, 48, 20, 49, 16, 50, 70, 73, 84
 10, 12, 20, 48, 49, 16, 50, 70, 73, 84
 10, 12, 20, 48, 16, 49, 50, 70, 73, 84
 Pass 6 10, 12, 20, 16, 48, 49, 50, 70, 73, 84
 Pass 7 10, 12, 16, 20, 48, 49, 50, 70, 73, 84
 Total Pass 7

Prob.11 What do you mean by counting sort? Also writes its algorithm.

Sol. Counting sort assumes that each of the n input elements is an integer in the range 0 to k , for some integer k . When $k = Q(n)$, the sort runs in $\Theta(n)$ time.

The basic idea of counting sort is to determine, for each input element x , the number of elements less than x . This information can be used to place element x directly into its position in the output array. For example, if there are 17 elements less than x , then x belongs in output position 18. This scheme must be modified slightly to handle the situation in which several elements have the same value, since we don't want to put them all in the same position.

In the code for counting sort, we assume that the input is an array $A[1 \dots n]$, and thus $\text{length}[A] = n$. We require two other arrays $B[1 \dots n]$ holds the sorted output, and the array $C[0 \dots k]$ provides temporary working storage.

Counting Sort (A, B, k)

1. for $i \leftarrow 0$ to k
2. do $C[i] \leftarrow 0$
3. for $j \leftarrow 1$ to $\text{length}[A]$
4. do $C[A[j]] \leftarrow C[A[j]] + 1$
5. $<C[i]$ now contains the number of elements equal to i .
6. for $i \leftarrow 1$ to k
7. do $C[i] \leftarrow C[i] + C[i - 1]$
8. $<C[i]$ now contains the number of elements less than or equal to i .
9. for $j \leftarrow \text{length}[A]$ downto 1
10. do $B[C[A[j]]] \leftarrow A[j]$
11. $C[A[j]] \leftarrow C[A[j]] - 1$

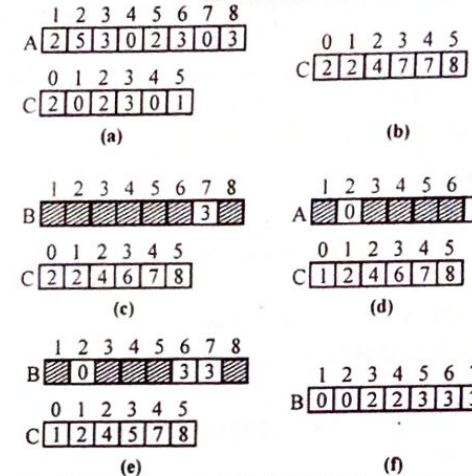


Fig. The operations of COUNTING-SORT on an input array $A[1..8]$, where each element of A is a nonnegative integer no larger than $k = 5$. (a) The array A and the auxiliary array C after line 4. (b) The array C after line 7. (c)-(e) The output array B and the auxiliary array C after one, two, and three iterations of the loop in line 9-11, respectively. Only the lightly shaded elements of array B have been filled in. (f) The final sorted output array B .

PART-C

Prob.12 Write down the algorithm of counting sorting and sort following elements using counting sorting.

$< 2, 1, 3, 9, 6, 1, 3, 9, 6, 5, 6, 8, 5, 3 >$

[R.T.U. 2018]

Sol. Counting Sort Algorithm : Counting sort is an algorithm for sorting a collection of objects according to keys that are small integers; that is, it is an integer sorting algorithm. It is a linear time sorting algorithm used to sort items when they belong to a fixed and finite set.

The algorithm proceeds by defining an ordering relation between the items from which the set to be sorted is derived (for a set of integers, this relation is trivial). Let the set to be sorted be called A . Then, an auxiliary array with size equal to the number of items in the superset is defined, say B . For each element in A , say e , the algorithm stores the number of items in A smaller than or equal to e in $B(e)$. If the sorted set is to be stored in an array C , then for each e in A , taken in reverse order, $C[B(e)] = e$. After each such step, the value of $B(e)$ is decremented.

Algorithm :

In the code for Counting sort, we are given array $A[1 \dots n]$ of length n . We required two more arrays, the

2. Binary Search :

This binary search technique works for sorted list of element and it is very efficient searching technique.

Algorithm for Binary Search : Refer to Prob.1.

Example :

Let DATA be the following sorted 13 elements array :

DATA : 11, 22, 30, 33, 40, 44, 55, 60, 66, 77, 80, 88, 99

We apply the binary search to DATA for different values of ITEM.

(a) Suppose ITEM = 40. The search for item in the array DATA is pictured in fig., where the values of DATA[LOW] and DATA[HIGH] in each stage of the algorithm are indicated by circles and the value of DATA[mid] by a square.

(1) Initially, Low = 1 and High = 13 and

MID = $\text{INT}[(1 + 13)/2] = 7$ and so DATA[MID] = 55

(2) Since $40 < 55$, High has its value changed by

High = MID - 1 = 6, Hence

MID = $\text{INT}[(1 + 6)/2] = 3$ and so DATA[MID] = 30

(3) Since $40 > 30$ Low has its value changed by

Low = MID + 1 = 4, Hence

MID = $\text{INT}[(4 + 6)/2] = 5$ and so DATA[MID] = 40

we have found ITEM in location LOC = MID = 5

(1) 11, 22, 30, 33, 40, 44, 55, 60, 66, 77, 80, 88, 99

(2) 11, 22, 30, 33, 40, 44, 55, 60, 66, 77, 80, 88, 99

(3) 11, 22, 30, 33, 40, 44, 55, 60, 66, 77, 80, 88, 99

[successful]

The item 40 is found.

Binary search requires a particular order of list whereas linear search is not.

Prob.15 Explain with a suitable example the principle of operation of heap sort. [R.T.U. Jan. 2016]

OR

By taking suitable example explain the principle of operation of heap sort. [R.T.U. 2017]

Sol. Heap sort : Refer to Prob.9(b).

Inserting into a Heap.

H is a heap with N elements, to insert ITEM into heap.

(1) First adjoin ITEM at the end of H so that H is still a complete tree, but not necessarily a heap.

(2) Then let ITEM rise to its "appropriate place" in H so that H is finally a heap.

(i) **Insert**

INSHEAP(TREE, N, ITEM)

A heap H with N elements is stored in the array TREE, and an ITEM of information is given. This procedure inserts

ITEM as a new element of H. PTR gives the location of ITEM as it rises in the tree, and PAR denotes the location of the parent of ITEM.

1. [Add new node to H and initialize PTR.]

Set $N := N + 1$ and $PTR := N$.

2. [Find location to insert ITEM.]

Repeat Steps 3 to 6 while $PTR < 1$.

3. Set $PAR := \lfloor PTR/2 \rfloor$. [Location of parent node.]

4. If $ITEM \leq TREE[PAR]$, then:

Set $TREE[PTR] := ITEM$, and Return.

[End of If structure.]

5. Set $TREE[PTR] := TREE[PAR]$. [Move node down.]

6. Set $PTR := PAR$. [Updates PTR.]

[End of Step 2 loop.]

7. [Assign ITEM as the root of H.]

Set $TREE[1] := ITEM$.

8. Return.

(ii) **Deletion in Heap :**

DELHEAP(TREE, N, ITEM)

A heap H with N elements is stored in the array TREE. This procedure assigns the root $TREE[1]$ of H to the variable ITEM and then reheaps the remaining elements. The variable LAST saves the value of the original last node of H. The pointers PTR, LEFT and RIGHT give the locations of LAST and its left and right children as LAST sinks in the tree.

1. Set $ITEM := TREE[1]$. [Removes root of H.]

2. Set $LAST := TREE[N]$ and $N := N - 1$. [Removes last node of H.]

3. Set $PTR := 1$, $LEFT := 2$ and $RIGHT := 3$. [Initializes pointers.]

4. Repeat Steps 5 to 7 while $RIGHT \leq N$:

5. If $LAST \geq TREE[LEFT]$ and $LAST \geq TREE[RIGHT]$, then:

Set $TREE[PTR] := LAST$ and Return.

[End of If structure.]

6. If $TREE[RIGHT] \leq TREE[LEFT]$, then:

Set $TREE[PTR] := TREE[LEFT]$ and

$PTR := LEFT$.

Else:

Set $TREE[PTR] := TREE[RIGHT]$ and

$PTR := RIGHT$

[End of If structure.]

7. Set $LEFT := 2 * PTR$ and $RIGHT := LEFT + 1$.

[End of Step 4 loop.]

8. If $LEFT = N$ and if $LAST < TREE[LEFT]$, then: Set $PTR := LEFT$.

9. Set $TREE[PTR] := LAST$.

10. Return.

(iii) **Heap Sort :**

Suppose an array A with N elements is given. The heapsort algorithm to sort A consists of the two following phases :

Phase A : Build a heap H out of the elements of A.

Phase B : Repeatedly delete the root element of H.

Since the root of H always contains the largest node in H, phase B deletes the element of A in decreasing order. A formal statement of the algorithm is given below.

Algorithm: HEAPSORT(A, N)

An array A with N elements is given. This algorithm sorts the elements of A.

1. [Build a heap H, using insertion procedure]

Repeat for J = 1 to N - 1:

Call INSHEAP(A, J, A[J + 1]). [End of loop.]

2. [Sort A by repeatedly deleting the root of H, using delete procedure]

Repeat while $N > 1$:

(a) Call DELHEAP(A, N, ITEM).

(b) Set $A[N + 1] := ITEM$. [End of Loop.]

3. Exit.

Prob.16 (a) Write an algorithm for merge sort and comment on its complexity.

(b) Sort the following data in ascending order using Quick sort :

9, 4, 12, 6, 5, 10, 7

[R.T.U. Dec. 2016]

Sol.(a) Merge Sort : The mergesort algorithm is based on the classical divide-and-conquer paradigm. It operates as follows:

Divide : Partition the n-element sequence to be sorted into two subsequences of $n/2$ elements each.

Conquer : Sort the two subsequences recursively using the mergesort.

Combine : Merge the two sorted subsequences of size $n/2$ each to produce the sorted sequence consisting of n elements.

Note that recursion "bottoms out" when the sequence to be sorted is of unit length. Since every sequence of length 1 is in sorted order, no further recursive call is necessary. The key operation of the mergesort algorithm is the merging of the two sorted sequence in the "combine step". To perform the merging, we use an auxiliary procedure Merge (A, p, q, r), where A is an array and p, q and r are indices numbering elements of the array such that we assume that the subarrays $A[p..q]$ and $A[q+1..r]$ are in sorted order. It merges them to form a single sorted subarray that replaces the current

An AVL search tree is a binary search tree that is also an AVL tree. The height of an empty (null) AVL tree is taken as -1.

Representation of an AVL Tree : The node in an AVL tree will have an additional field bf (for balance factor) in addition to the structure of a node in a binary search tree. The required memory declarations for representing a node of AVL tree will look like :

```
typedef struct nodeType
{
    struct nodeType *left;
    int info;
    int bf;
    struct nodeType *right;
} avlNode;
avlNode *root;
```

B.F. = height of left subtree - height of right subtree.
The value for field bf will be chosen as

$$bf = \begin{cases} -1 & \text{if } h_L < h_R \\ 0 & \text{if } h_L = h_R \\ +1 & \text{if } h_L > h_R \end{cases}$$

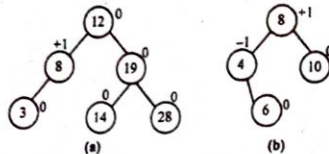


Fig. 4 : AVL trees

If the balance of a node becomes -2 or +2 then it will require rebalancing.

□ B-TREE

The B-tree structure was discovered by R. Bayer and E. McCreight of Boeing Scientific Research Labs and has become one of the most popular techniques for organizing an index structure.

A B-tree of order m is a m-way search tree with the following properties:

- Each node of the tree, except the root node and leaves, has at least m/2 subtrees and no more than m subtrees.
- Root of the tree has at least two subtrees unless it is a leaf node.

All leaves of the tree are on the same level.

The first condition ensures that each node of the tree is at least half full, the second one forces the tree to branch early while the third one keeps the tree balanced.

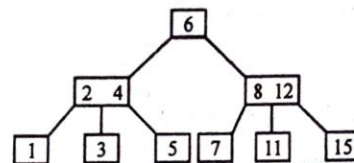


Fig. 5

Several variations of the basic B-tree have been developed, two of them, that have become popular are:

1. **B⁺-Tree:** It is a B-tree in which the leaf nodes are linked to each other from left to right to form a linked list of the keys in sequential order.

2. **B*-Tree:** It is a B-tree in which each interior node is at least two-third full rather than just half full.

Prob. 2 Write down the differences between B tree and B+ tree. [R.T.U. 2018]

Sol. B-Tree : The B-tree structure was discovered by R. Bayer and E. McCreight of Boeing Scientific Research Labs and has become one of the most popular techniques for organizing an index structure.

B⁺-Tree : It is a B-tree in which the leaf nodes are linked to each other from left to right to form a linked list of the keys in sequential order.

Prob. 3 Write short note on Binary Tree.

Sol. Binary Trees : A binary tree T is either empty or has a finite collections of elements. When the binary tree is not empty, one of its elements is called the *root* and the remaining elements, if any, are partitioned into two binary trees, which are called the *left* and *right subtrees* of T. A tree may be defined as a graph without cycles.

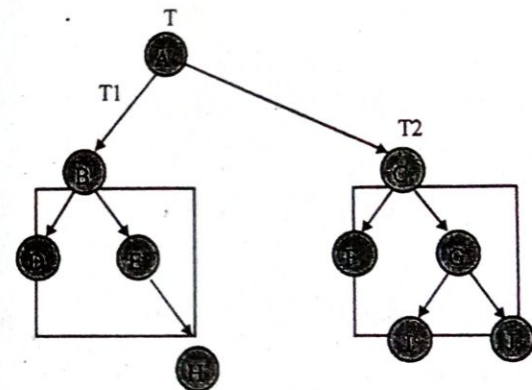


Fig.

Prob. 4 Define complete binary tree.

Sol. Complete binary tree : A complete binary tree of depth d is a strictly binary tree all of whose leaves are at level d.

A binary tree is made of nodes, where each node contains a "left" pointer, a "right" pointer, and a data element. The "root" pointer points to the topmost node in the tree. The left and right pointers recursively point to smaller "subtrees" on either side. A null pointer represents a binary tree with no elements, the empty tree. The formal recursive definition is: a **binary tree** is either empty (represented by a null pointer), or is made of a single node, where the left and right pointers (recursive definition ahead) each point to a binary tree.

PREVIOUS YEARS QUESTIONS

PART-A

Prob. 1 What do you mean by spanning tree? [R.T.U. 2018]

Sol. A spanning tree is a subset of Graph G, which has all the vertices covered with minimum possible number of edges. Hence, a spanning tree does not have cycles and it cannot be disconnected.

By this definition, we can draw a conclusion that every connected and undirected Graph G has at least one spanning tree. A disconnected graph does not have any spanning tree, as it cannot be spanned to all its vertices.

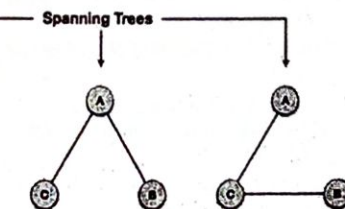
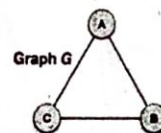


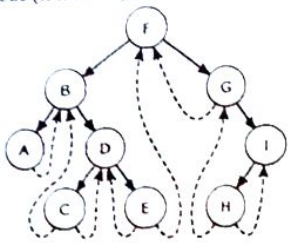
Fig.



Fig. Complete binary tree

Prob.5 Write short note on threaded binary.

Sol. Threaded Binary Tree : Inorder traversal of a Binary tree is either be done using recursion or with the use of an auxiliary stack. The idea of threaded binary trees is to make inorder traversal faster and do it without stack and without recursion. A binary tree is made threaded by making all right child pointers that would normally be NULL point to the inorder successor of the node (if it exists).



PART-B

Prob.6 The in-order and pre-order traversal sequence of nodes in a binary tree are given below :

In-order :	E	A	C	K	F	H	D	B	G
Pre-order :	F	A	E	K	C	D	H	G	B

Draw the binary tree. State briefly the logic used to construct the tree. [R.T.U. Jan. 2016]

OR

A Binary tree T has 9 nodes. The in order and pre order traversal for T yield the following sequences of nodes :

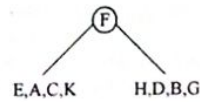
IN order : E A C K F H D B G

Pre order : F A E K C D H G B

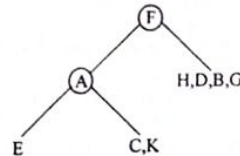
Draw the tree T [R.T.U. 2018]

Sol. In order : E, A, C, K, F, H, D, B, G
Pre-order : F, A, E, K, C, D, H, G, B

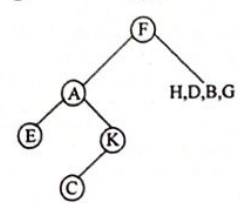
Step 1:



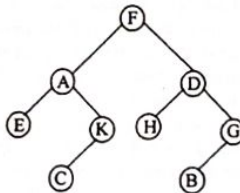
Step 2:



Step 3:



Step 4:



Logic used to construct the tree:

The concept/ logic used to construct this binary tree is that the leftmost element is the root of the tree in pre-order sequence. So, 'F' is the root of the given sequence. Now, by searching 'F' in the in-order sequence, all the elements on the left side of 'F' are in the left subtree, and elements on the right side are in right subtree. By recursively following the concept, get the binary tree.

Prob.7 What is an AVL tree? Explain the rotations of AVL tree. [R.T.U. Dec. 2016]

OR

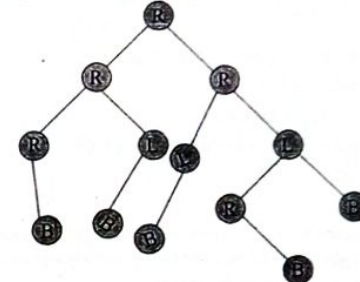
What are the different AVL tree rotations? Explain with suitable example. [R.T.U. 2018]

Sol. AVL tree: A tree is called AVL tree, if each node satisfies one of the properties :

1. A node is called left heavy, if the longest path in its left subtree is one longer than the longest path in its right subtree.
2. A node is called right heavy, if the longest path in

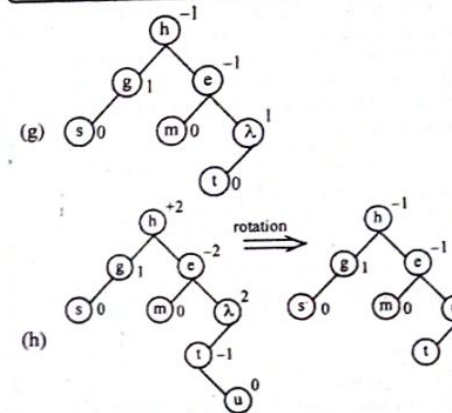
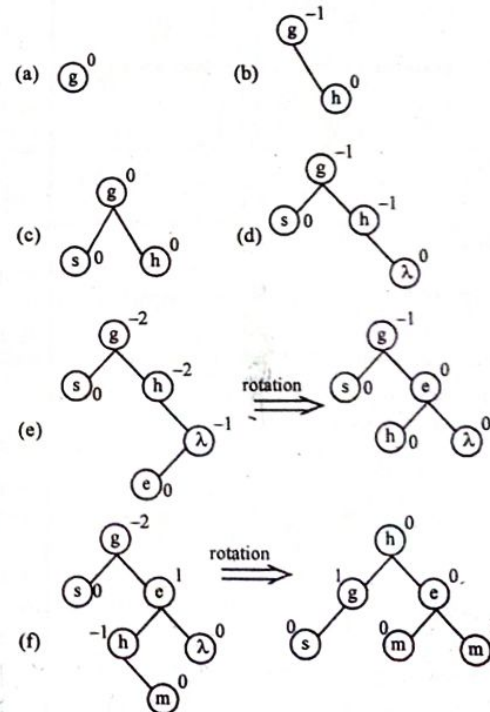
the right subtree is one longer than the longest path in its left subtree.

3. A node is called balanced, if the longest path in its right as well as left subtrees are equal.



Where B - Balanced
R - Right heavy
L - Left heavy

g, h, s, l, e, m, t, u (8 keys)



Prob.8 Suppose a binary tree T is in memory recursive procedure which finds the depth Dp of [R.T.U.]

Sol. A recursive method to find depth of a (not necessarily complete) Binary tree

//max() - returns the max of two numbers
int depth(root)

```

{
    if(root->left==NULL && root->right==NULL)
        return 0;
    else if(root->left!=NULL && root->right==NULL)
        right leaf
        return( max(depth(root->left),0)+1);
    else if(root->left==NULL && root->right!=NULL)
        left leaf
        return( max(0,depth(root->right)+1);
    else if(root->left->left==NULL && root->right->right==NULL && root->left->right==NULL && root->right->left==NULL)
        parent of two leaves
        return 1;
    else// for the condition that this is a parent of two
        return( max(depth(root->right),depth(root->left))+1);
}

```

Prob.9 What is binary tree? Mention the properties of a binary tree. Define the following :

- (i) Strictly binary tree
- (ii) Complete binary tree

[R.T.U.]

OR

Define the following binary tree

- (i) Strictly binary tree.
- (ii) Complete binary tree.

- The traversal operation is more faster than that of its unthreaded version, because with threaded binary tree non-recursive implementation is possible which can run faster and does not require the both operation of stack management.
- With a threaded binary tree, we can efficiently determine the predecessor and successor nodes starting from any node. In case of unthreaded binary tree, however, this task is more time consuming and difficult. For this case, a stack is required to provide upward pointing information in the tree whereas in a threaded binary tree, without having to incur the overhead of using a stack mechanism the same can be carried out with the threads.
- Any node is accessible from any other node. Threads are usually move upwards whereas links are downwards. Thus in a threaded tree, one can move in either directions and nodes are in fact circularly linked. This is not possible in unthreaded counter part because there we can move only in downward direction starting from root.
- Insertions into and deletions from a threaded tree are although time consuming operations (since we have to manipulate both links and threads) but these are very easy to implement.

Prob.11 Define Binary search tree. Write algorithm to implement insertion operation on Binary Search tree.
[R.T.U. Dec. 2016]

Sol. Binary Search Trees : Refer to Prob 9.

Prob.12 (a) Explain concept of balance factor in AVL tree with suitable example.

(b) Insert the following keys in the order given below to build them into an AVL tree.

g, h, s, l, e, m, t, u

Mention different rotation used and balance factor of each node.
[R.T.U. Jan. 2016]

Sol. (a) AVL tree: Refer to Prob 7.

Insertion in AVL Tree : We can insert in AVL tree by using binary tree insertion techniques, comparing key if new node with that in the root and inserting new node into the left or right subtrees accurately.

Algorithm of insertion in AVL tree :

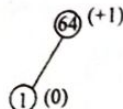
- If this is the first insertion then allocate a node, set its fields and return.

- If the insertion we want to insert is not already present in the tree then link the new node to the existing tree else discard the new information and return.
- Search for an unbalanced node.
- Adjust the balance factor if there is no critical node return.
- If the node was balanced and then becomes heavy or the node was heavy and new becomes balanced then adjust the balanced factors and return.
- Rebalance the tree and return.

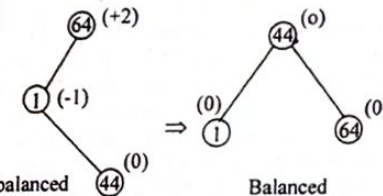
(1) insert 64



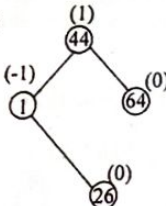
(2) insert 1



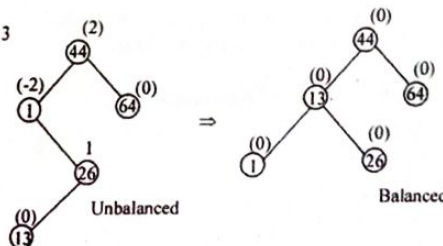
(3) insert 44



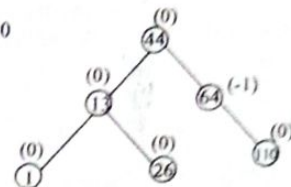
(4) insert 26



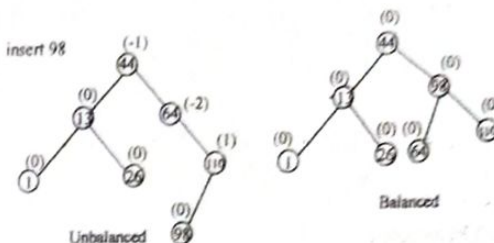
(5) insert 13



(6) insert 110



(7) insert 98



(8) insert 85

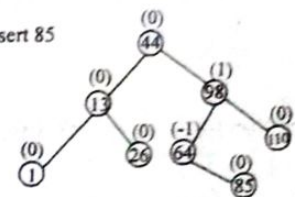


Fig. : B-tree of order 5

Sol. (b) Refer to Prob.7.

Prob.13 (a) Write an algorithm to delete a data key from a threaded binary tree.

(b) How will you use a tree for representing a set? Explain.

[R.T.U. 2014]

Sol.(a) Algorithm Delete_TBST

Input: PTR is the pointer of a node that has to be deleted.

Output: The threaded binary search tree without a node PTR or its data content.

Data structure: Linked structure of threaded binary tree.

Step

- parent = Parent(PTR) //Get the parent of the node PTR
- If (PTR → LTAG=TRUE) and (PTR → RTAG=TRUE) then //Identify the case of deletion
- case = 1
- Else
- If (PTR → LTAG=FALSE) and (PTR → RTAG=FALSE) then

PREVIOUS YEARS QUESTIONS

PART-A

Prob.1 Write short note on Comparison of sorting algorithms in terms of time complexity. [R.T.U. 2017]

OR
Write down the asymptotic upper bound of bubble sort, selection sort, quick sort and heap sort? [R.T.U. 2018]

Sol. Time Complexity comparison of Sorting Algorithms

Algorithm	Data Structure	Best	Average	Worst
Quick sort	Array	$O(n \log n)$	$O(n \log n)$	$O(n^2)$
Merge sort	Array	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$
Heap sort	Array	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$
Bubble sort	Array	$O(n)$	$O(n^2)$	$O(n^2)$
Insertion sort	Array	$O(n)$	$O(n^2)$	$O(n^2)$
Selection sort	Array	$O(n^2)$	$O(n^2)$	$O(n^2)$
Bucket sort	Array	$O(n+k)$	$O(n+k)$	$O(n^2)$
Radix Sort	Array	$O(nk)$	$O(nk)$	$O(nk)$

Prob.2 Write down the differences between BFS and DFS. [R.T.U. 2018]

Sol. Breadth-First Search: The general idea behind a breadth-first search is to start at a starting node A and then examine all the neighbours of A. Then we examine all the neighbours of A, and so on.

Depth-First Search: The general idea is a depth-first search is to start at a starting node A and then examine each node N along a path P which begins at A, that is, we process a neighbour of A, then a neighbour of a neighbour of A, and so on.

Prob.3 Write short notes on hash functions. [R.T.U. 2018]

Sol. The Hash function primarily is responsible to map between the original data item and the smaller table itself. Here the mapping takes place with the help of an output integer in a consistent range produced when a given data item (any data type) is provided for storage and thus output integer range determines the location in the smaller table for the data item.

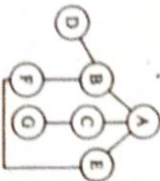
In terms of implementation, the hash table is constructed with the help of an array and the indices of this array are associated to the output integer range.

PART-B

Prob.4 Write an algorithm for DFS traversal. [R.T.U. Dec. 2016]

OR
Write short note on DFS traversal algorithms. [R.T.U. 2017]

Sol. DFS Traversal : It is an uninformed search that progresses by expanding the first child node of tree search that appears and thus going deeper and deeper until a goal node is found, or until a node without any further child is reached. Then the search backtracks, returning to the most recent node it hasn't finished expanding. The complexity of the algorithm is proportional to the number of vertices plus the number of edges in the graph, or $O(V+E)$.



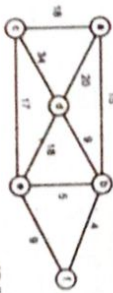
Suppose, for the adjacent graph, a DFS is initiated starting at node A, and assuming all left edges are expanded first before right edges and no nodes are repeated. Then, DFS traversal will yield the following order of nodes: A,B,D,F,E,C,G

Algorithm : This algorithm executes a depth - first search on a graph G beginning at a starting node A.

1. Initialize all nodes to the ready state (STATUS = 1).
2. Push the starting node A onto STACK and change its status to the waiting state (STATUS = 2).
3. Repeat steps 4 and 5 until STACK is empty.
4. Pop the top node N of STACK Process N and change its status to the processed state (STATUS = 3).
5. Push onto STACK all the neighbours of N that are still in the ready state (STATUS = 1), and change their status to the waiting state (STATUS = 2).
6. Exit.

The above algorithm will process only those nodes which are reachable from the starting node A. Suppose one wants to examine all the nodes in G. Then the algorithm must be modified so that it begins again with another node which we will call B - that is still in the ready state. This node B can be obtained by traversing the list of nodes.

Prob.5 Using Prim's and Kruskal's algorithm, find minimum spanning tree for the following graph.

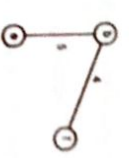


Sol. Prim's Algorithm

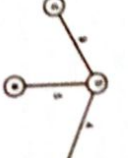
- (1)



- (2)



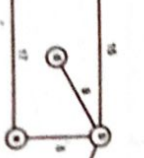
- (3)



- (4)



- (5)



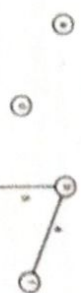
So, the minimum spanning tree with total Cost = 4 + 5 + 9 + 15 + 17 = 50 Ans.

Kruskal's Algorithm :

- (1)



- (2)



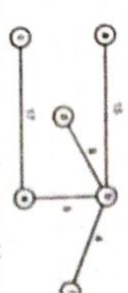
- (3)



- (4)



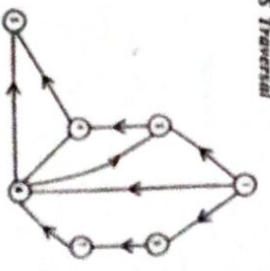
- (5)



So, the minimum spanning tree with total cost = 4 + 5 + 9 + 15 + 17 = 50

Prob.6 Find :

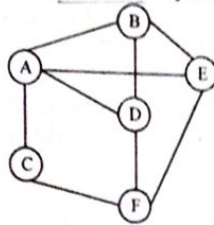
- (i) BFS Traversal
- (ii) DFS Traversal



For given graph, Explain in brief. [R.T.U. Jan. 2016]

Prob.7 For the graph shown below find the following:

- Adjacency list representation.
- Adjacency matrix representation.
- Adjacency multilist representation.



[R.T.U. 2015]

Sol.(i) Adjacency List Representation

Adjacency List: The type in which a graph is created with the linked list is called adjacency list. Now the above graph have the nodes as A, B, C, D, E and F. So we will maintain the linked list of these head nodes as well as the adjacent nodes.

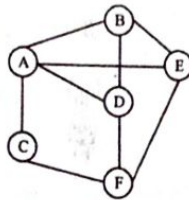
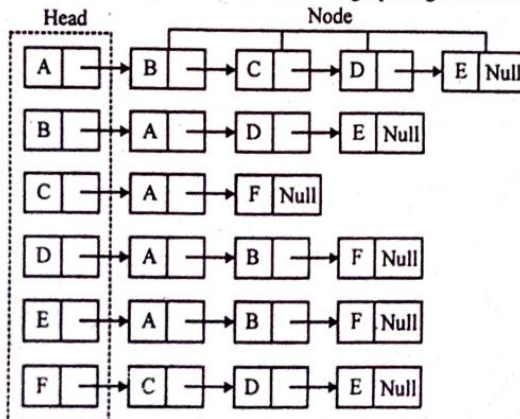


Fig.

Node	Adjacency list
A	B, C, D, E
B	A, D, E
C	A, F
D	A, B, F
E	A, B, F
F	C, D, E

The linked list representation of above graph is given below:



(ii) **Adjacency Matrix Representation :** In adjacency matrix (M), if there is an edge present between vertices V_i and V_j then

$$M[i][j] = 1 \quad \text{else } M[i][j] = 0$$

For an undirected graph if $M[i][j] = 1$ then for $M[j][i]$ is also 1

	A	B	C	D	E	F
A	0	1	1	1	1	0
B	1	0	0	1	1	0
C	1	0	0	0	0	1
D	1	1	0	0	0	1
E	1	1	0	0	0	1
F	0	0	1	1	1	0

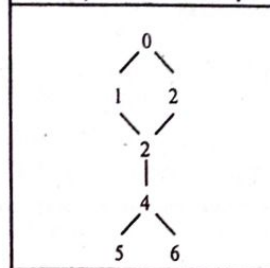
(iii) **Adjacency Multilists Representation :** Adjacency Multi-lists are an edge, rather than vertex based, graph representation. Typically, the following structure is used to represent an edge.

Visited	vertex at tail	vertex at head	Pointer to next edge containing vertex at tail	Pointer to next edge containing vertex at head
---------	----------------	----------------	--	--

The node structure is coupled with an array of head nodes that point to the first edge that contains the vertex as it tail, i.e., the first entry in the ordered pair.

The adjacency multi-list for our sample GRAPH would be:

A Sample Undirected Graph



[0]	Edge 1	Edge 1	0	1	Edge 2	Edge 3
[1]	Edge 1	Edge 2	0	2	NULL	Edge 4
[2]	Edge 2	Edge 3	1	3	NULL	Edge 4
[3]	Edge 3	Edge 4	2	3	NULL	Edge 5
[4]	Edge 5	Edge 5	3	4	NULL	Edge 6
[5]	Edge 6	Edge 6	4	5	Edge 7	NULL
[6]	Edge 7	Edge 6	4	6	NULL	NULL

Following the link fields for each edge let's you reconstruct the adjacency list for that vertex. The following table shows the edges that contain each of the vertices.

Vertex	Edge list
0	Edge 1, Edge 2
1	Edge 1, Edge 3
2	Edge 2, Edge 4
3	Edge 3, Edge 4, Edge 5
4	Edge 5, Edge 6, Edge 7
5	Edge 6
6	Edge 7

Now for given Graph : There are total 9 edges in Graph let each edge have a name as follows

- (A,B)-a
(A,C)-b
(A,D)-c
(A,E)-e
(B,D)-f
(B,E)-g
(C,F)-h
(D,F)-i
(E,F)-j

E _a	A	B	E _b	E _c	E _d	E _f	E _g
E _b	A	C	Null	Null	E _c	E _d	E _h
E _c	A	D	Null	Null	E _e	E _g	E _h
E _d	A	E	Null	Null	Null	Null	E _h
E _e	B	D	Null	Null	Null	E _g	E _i
E _f	B	E	Null	Null	Null	Null	E _j
E _g	C	F	Null	Null	Null	Null	E _i
E _h	D	F	Null	Null	Null	E _i	Null
E _i	E	F	Null	Null	Null	Null	Null

Like this we cover all the edges

Prob.8 (a) Is there any difference among forest, graph and tree? Explain.

(b) Write prim's algorithm to get minimum spanning tree out of a graph. [R.T.U. 2014]

Sol. (a) Yes, there is a difference between forest, graph and tree. Although they are associated to each other.

A tree is an undirected simple graph G that satisfies any of the following equivalent conditions:

- G is connected and has no cycles.
- G has no cycles and a simple cycle is formed if any edge is added to G .
- G is connected and it is not connected anymore if any edge is removed from G .
- G is connected and the 3-vertex complete graph K_3 is not a minor of G .
- Any two vertices in G can be connected by a unique simple path.

A forest is an undirected graph, all of whose connected components are tree. In other words, the graph consists of a disjoint union of trees. Equivalently, a forest is an undirected cycle-free graph. As special cases, an empty graph, a single tree and the discrete graph on a set of vertices (that is, the graph with these vertices that has no edges), all are examples of forests. The term hedge sometimes refers to an ordered sequence of trees.

A graph is an ordered pair $G = (V, E)$ comprising a set V of vertices or nodes together with a set E of edges or lines, which are 2-elements subsets of V (i.e., an edge is related with two vertices and the relation is represented as an unordered pair of the vertices with respect to the particular edge). To avoid ambiguity, this type of graph may be described precisely as undirected and simple.

Sol.(b) Prim's Algorithm

Suppose that $G = (V, E)$ be a graph with $|V| = n$ vertices.

The steps of the algorithm are as follows:

First Step: Display n isolated vertices and label them $V_1, V_2, \dots, V_n$.

Second step: Tabulate the given weights of the edges of G in $n \times n$ table. Set the weights of non-existent edges as very large (say ∞).

Third step: Now start moving from vertex V_1 and connect it to its nearest neighbour, say V_i .

Fourth step: To the subgraph V_1, V_i derived in step 3, connect a vertex which is closest to it. Let the new vertex is V_j , then V_1, V_i, V_j is a new sub graph, continue this process until all the n vertices have been connected by $(n-1)$ edges.

Example: Find the minimal spanning tree by the application of Prim's algorithm in the following graph.

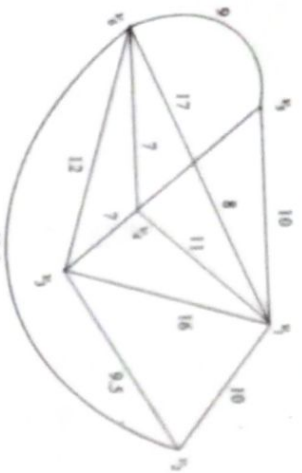


Fig.

Solution: There are 6 vertices and 12 edges. Mark the 6 vertices.

Tabulate the weights of edges in 6×6 table as detailed below

	v_1	v_2	v_3	v_4	v_5	v_6
v_1	-	10	16	11	10	17
v_2		-	9.5	∞	∞	19.5
v_3			-	7	∞	12
v_4				-	8	7
v_5					-	9
v_6						-

Now we construct the spanning tree as follows:



Therefore, the minimal weight = $8 + 7 + 7 + 9.5 + 10 = 41.5$

Prob.9 Write short note on Dijkstra algorithm.

(R.T.U. 2014)

Sol. Single Source Shortest Path: Dijkstra Algorithm

We are given a directed graph $G(V, E)$ in which each edge has a non-negative weight and one vertex is specified as the source. Our problem is to determine the cost of the shortest path from the source to every other vertex in V , where the length of a path is just the sum of the costs of the edges of the path. This problem is known as Single Source shortest path problem.

We might think problem is known as shortest distance from Jaipur to all other cities connected with Jaipur. The problem

can be solved by Dijkstra algorithm using greedy approach.

The algorithm works by maintaining a set S of vertices whose shortest distance from the source is already known. Initially, S contains only source vertex. At each step, we add to S a remaining vertex v whose distance from the source is as short as possible. The algorithm is as follows:

DIJKSTRA ((G(V, E)))

(This algorithm computes the cost of the shortest paths from vertex to every vertex of a directed graph.)

- $S = \{1\}$
- for $i = 2$ to n do
 $D[i] = C[1, i]$
- for $i = 1$ to $n - 1$ do
 choose a vertex w in $V - S$ such that $D[w]$ is a minimum add w to S .

for each vertex v in $V - S$ do
 $D[v] = \min(D[v], D[w] + C[w, v])$

Example

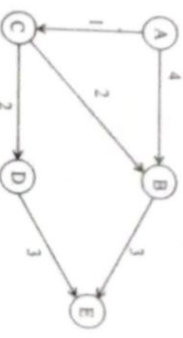


Fig.1: Weighted undirected graph

The above weighted graph has 5 vertices from A to E. The value between the two vertices is known as the weight between two vertices. For example the weight between A and C is 1. Using the above graph the Dijkstra's algorithm is used to determine the shortest path from the source A to the remaining vertices in the graph.

Step 1

$sDist[A] = 0$ [the initial value to the source itself]
 $sDist[B] = ?$, $sDist[C] = ?$, $sDist[D] = ?$, $sDist[E] = ?$ [the nodes not processed yet]

Step 2

$Adj[A] = \{B, C\}$ [computing the value of the adjacent vertices of the graph]
 $sDist[B] = 4$, $sDist[C] = 1$;

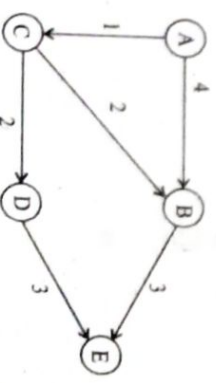


Fig.2: Shortest path to vertices B, C from A

Step 3

$Adj[B] = \{D, E\}$
 $sDist[B] > sDist[C] + EdgeCost[C, B]$ ($4 > 1 + 2$ (True))
Therefore, $sDist[B] = 3$
 $sDist[D] = sDist[C] + EdgeCost[C, D] = 1 + 2 = 3$

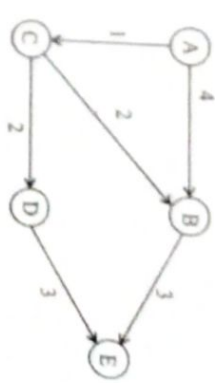


Fig.3: Shortest path from B, D using C as intermediate vertex

Step 4

$Adj[D] = \{E\}$
 $sDist[E] = sDist[B] + EdgeCost[B, E] = 3 + 3 = 6$
 $sDist[E] = sDist[D] + EdgeCost[D, E] = 3 + 3 = 6$
 $Adj[E] = \{E\}$
 $sDist[E] = sDist[D] + EdgeCost[D, E] = 3 + 3 = 6$
This is same as the initial value that was computed so $sDist[E]$ value is not changed.

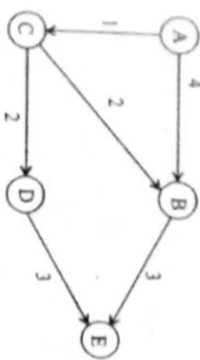


Fig.4: Shortest path to E using B as intermediate vertex

Step 5

$Adj[E] = \{ \}$ i.e. There is no outgoing edges from E, and no more vertices, algorithm terminated. Hence the path is presented in Fig.5.

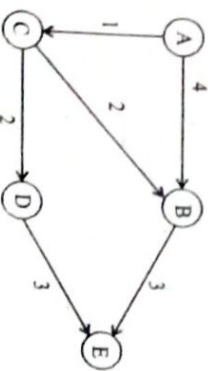
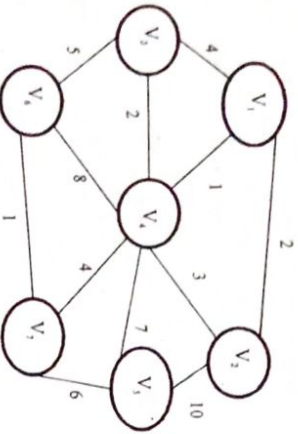


Fig.5: The path obtained using Dijkstra's Algorithm

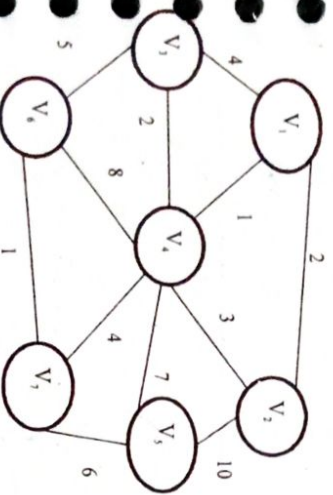
PART-C

Prob. 10: Obtain minimal spanning tree using prim's and Kruskal's algorithm on the following graph



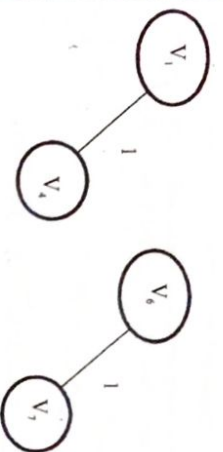
[R.T.U. 2018]

Sol. Krushal

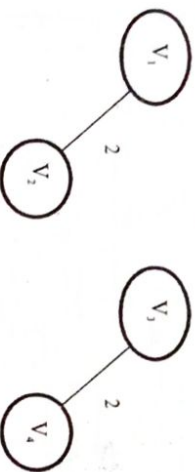


Arrange edges according to the ascending order weight

Step 1: Step 2:



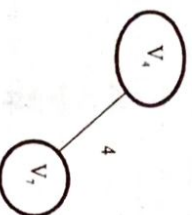
Step 3: Step 4:



Step 5:

Ignore $\{V_4, V_2\}$ and $\{V_1, V_3\}$ because it create cycle.

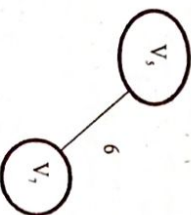
Step 6:



Step 7:

Ignore $\{V_3, V_6\}$ because it create cycle again

Step 8:

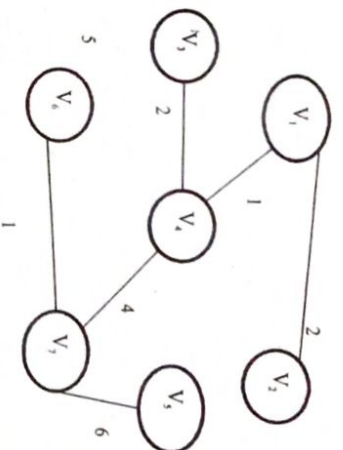


Step 9:

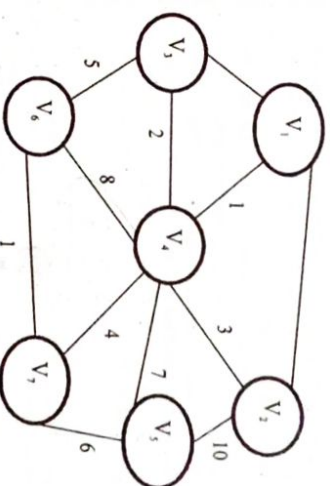
Ignore $\{V_4, V_5\}, \{V_4, V_6\}$ and $\{V_2, V_5\}$

Step 10:

The final graph is:



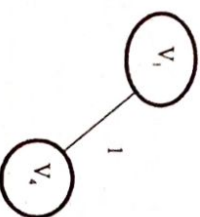
Prims:



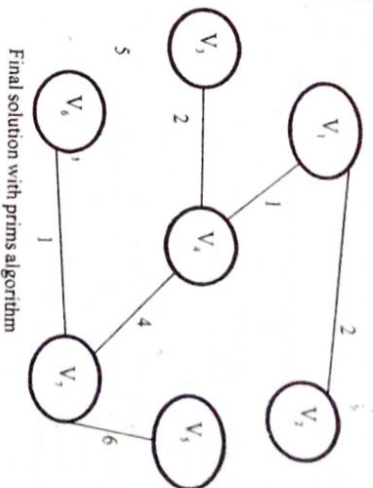
By using the prim's algorithm the solution is start from v_1 and the its adjacencies $\{V_3, V_2, V_4\}$

Select minimum one connection:

Step 1:



Step 2:



Final solution with prim's algorithm

Prob. 11) What do you mean by hash functions? Explain common hashing functions along with all address calculation techniques.

[R.T.U. 2018]

Sol. What is Hashing?

- Hashing is the process of mapping large amount of data item to smaller table with the help of hashing function.
- Hashing is also known as **Hashing Algorithm** or **Message Digest Function**.
- It is a technique to convert a range of key values into a range of indexes of an array.
- It is used to facilitate the next level searching method when compared with the linear or binary search.
- Hashing allows to update and retrieve any data entry in a constant time $O(1)$.
- Constant time $O(1)$ means the operation does not depend on the size of the data.
- Hashing is used with a database to enable items to be retrieved more quickly.
- It is used in the encryption and decryption of digital signatures.
- What is Hash Function?**
 - A fixed process converts a key to a hash key is known as a **Hash Function**.
 - This function takes a key and maps it to a value of a certain length which is called a **Hash value** or **Hash**.
 - Hash value represents the original string of characters, but it is normally smaller than the original.
 - It transfers the digital signature and then both hash value and signature are sent to the receiver. Receiver uses the same hash function to generate the hash value and then compares it to that received with the message.
 - If the hash values are same, the message is transmitted without errors.

What is Hash Table?

- Hash table or hash map is a data structure used to store key-value pairs.
- It is a collection of items stored to make it easy to find them later.
- It uses a hash function to compute an index into an array of buckets or slots from which the desired value can be found.
- It is an array of list where each list is known as bucket.
- It contains value based on the key.
- Hash table is used to implement the map interface and extends Dictionary class.
- Hash table is synchronized and contains only unique elements.

0	1	2	3	4	5	6	7	8	9

Fig. Hash Table

- The above figure shows the hash table with the size of $n = 10$. Each position of the hash table is called as Slot. In the above hash table, there are n slots in the table, names = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}. Slot 0, slot 1, slot 2 and so on. Hash table contains no items, so every slot is empty.
- As we know the mapping between an item and the slot where item belongs in the hash table is called the hash function. The hash function takes any item in the collection and returns an integer in the range of slot names between 0 to $n-1$.
- Suppose we have integer items {26, 70, 18, 31, 54, 93}. One common method of determining a hash key is the division method of hashing and the formula is:
Hash Key = Key Value % Number of Slots in the Table
- Division method or remainder method takes an item and divides it by the table size and returns the remainder as its hash value.

Data Item	Value % No. of Slots	Hash Value
26	$26 \% 10 = 6$	6
70	$70 \% 10 = 0$	0
18	$18 \% 10 = 8$	8
31	$31 \% 10 = 1$	1
54	$54 \% 10 = 4$	4
93	$93 \% 10 = 3$	3

0	1	2	3	4	5	6	7	8	9
70	31		93	54		26		18	

Fig. Hash Table

- After computing the hash values, we can insert each item into the hash table at the designated position as shown in the above figure. In the hash table, 6 of the

10 slots are occupied, it is referred to as the load factor and denoted by, $\lambda = \text{No. of items} / \text{table size}$. For example, $\lambda = 6/10$.

- It is easy to search for an item using hash function where it computes the slot name for the item and then checks the hash table to see if it is present.
- Constant amount of time $O(1)$ is required to compute the hash value and index of the hash table at that location.

Linear Probing

- Take the above example, if we insert next item 40 in our collection, it would have a hash value of 0 ($40 \% 10 = 0$). But 70 also had a hash value of 0, it becomes a problem. This problem is called as **Collision** or **Clash**. Collision creates a problem for hashing technique.
- **Linear probing is used for resolving the collisions in hash table**, data structures for maintaining a collection of key-value pairs.
- Linear probing was invented by Gene Amdahl, Elaine M. McGraw and Arthur Samuel in 1954 and analyzed by Donald Knuth in 1963.
- It is a component of open addressing scheme for using a hash table to solve the dictionary problem.
- The simplest method is called Linear Probing. Formula to compute linear probing is:

$$P = (1 + P) \% (\text{MOD}) \text{ Table_size}$$

For example,

0	1	2	3	4	5	6	7	8	9
70	31		93	54		26		18	

Fig. Hash Table

If we insert next item 40 in our collection, it would have a hash value of 0 ($40 \% 10 = 0$). But 70 also had a hash value of 0, it becomes a problem.

Linear probing solves this problem:

$$P = H(40)$$

$$44 \% 10 = 0$$

Position 0 is occupied by 70. so we look elsewhere for a position to store 40.

Using Linear Probing:

$$P = (P + 1) \% \text{table-size}$$

$$0 + 1 \% 10 = 1$$

But, position 1 is occupied by 31, so we look elsewhere for a position to store 40.

Using linear probing, we try next position : $1 + 1 \% 10 = 2$

Position 2 is empty, so 40 is inserted there.

0	1	2	3	4	5	6	7	8	9
70	31	40	93	54		26		18	

Fig. Hash Table

inside the boundary of the class.

2. **Protected:** The members declared as protected are accessible within the class as well as by the members of derived class. This access specifier useful in inheritance.
3. **Public:** The members declared under the access specifier public are accessible inside the class, in derived class and outside the class by objects of the class.

❑ PASSING OBJECT AS ARGUMENTS

Objects may be passed to functions in the same way as any other type of variables can. Objects can be passed as an argument by the following ways:

- (a) **Call by Value:** In this method entire copy of an object is passed to the function.

- (b) **Call by Reference:** In this method reference of an object is passed to the function.

Passing Objects by Value

In this method a copy of entire object is passed through the function call to its definition. If we make any modification in the called function definition then changes does not reflect in the object used in the calling function.

Passing Objects by Reference

In this method reference of object is passed to function's definition. Here we use '&' sign before objects in function definition so that they can receive references of objects passed in function call. If any modification is done in called function definition it will also be reflected in the calling function.

PREVIOUS YEARS QUESTIONS

PART-A

Prob.1 Name the operators which cannot overload. What is the significance of scope resolution operators (::)?

[R.T.U. 2018]

Sol. Operators that cannot be overloaded -

- (i) Pointer to member operator (.*)
- (ii) Scope resolution operator (::)
- (iii) Conditional operator (?:)

Significance of Scope Resolution Operators (::): The scope resolution operator (::) in C++ is used to define the declared member functions (in the header file with .cpp extension) of the program. In the .cpp file one can define the normal functions or the member functions of the class. To differentiate from the normal functions with the member functions of the class, one needs to use the scope resolution operator (::) in between the class name and the member function name i.e. ship::foo() where ship is the class and the foo() is the member function in the ship. The other use of the resolution operator is to resolve the scope of the variables if the same variable-name is used for the global, local, and the data member of the class. If the resolution operator is placed between the class name and the data member belonging to the class then the data name belonging to the particular class is affected. If the resolution operator is placed in front of the variable name then the global variable is affected. If no resolution operator is placed then the local variable is affected.

We can tell the compiler to use the global identifier rather than the local identifier by prefixing the identifier with ::, the scope resolution operator.

```
:: identifier
class-name :: identifier
namespace :: identifier
```

//The identifier can be a variable or a function.

If we have nested local scopes, the scope resolution operator does not provide access to identifiers in the next outermost scope. It provides access to only the global identifiers.

This example has two variables named amount. The first is global and contains the value 123. The second is local to the main function. The scope resolution operator tells the compiler to use the global amount instead of the local one.

```
#include <iostream.h>
using namespace std;
int amount = 123;    //A global variable
int main()
{
    int amount = 456; // A local variable
    cout << :: amount << endl
    // Print the global variable
    cout << amount << endl;
    // Print the local variable
}
```

Prob.2 What are the differences between class and structure in terms of memory allocation? [R.T.U. 2018]

Sol. The differences between Class and Structure in terms of Memory Allocation: Structures are value types and the classes are reference types. Imagine this is the memory within the machine:

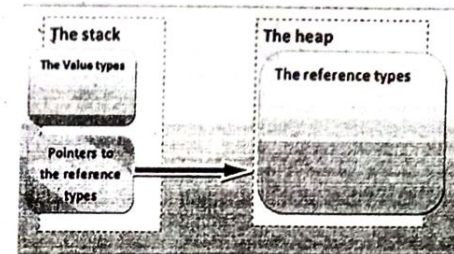


Fig.

The value types are stored in the stack but the reference types are not. In fact, what could be really stored in the stack for the reference types is a pointer that targets an address at the heap level.

Then the type of structure objects are stored in the stack exactly like any value type, say an integer, a double or a float. Meanwhile, memory locations could be reserved for reference types in the heap.

Classes are usually used for large amounts of data, whereas structures are usually used for smaller amounts of data.

Prob.3 What is Run-Time Error, Logical and Syntax Error? [R.T.U. 2018]

Sol. Syntax Error: Errors in coding which do not follow language syntax format. It can occur by human mistake in typing and due to lack of knowledge of language.

Example: Ram are a boy.

Error Correct: Ram is a boy.

Logical Error: Logical error will execute program properly but not gives correct answer.

Example: Table is a boy. Syntax correct but meaning is not.

Runtime Error: Overflow, underflow, out of memory capacity.

Prob.4 What is the role of public, private and protected access specifiers within the class? [R.T.U. 2018]

Sol. Private, Protected and Public Visibility Modes

We can inherit class by using private, protected or public visibility mode. In all three cases program behaves differently.

Private Visibility Mode

When the base class B is inherited by using the private access specifier, all public and protected members of the base class B become private members of the derived class D.

Protected Visibility Mode

When we use protected as visibility mode of base class B then all public and protected members of base class B become protected members in derived class D.

Public Visibility Mode

When the visibility mode for base class B is public, all public members of base class B become public members of the derived class D and all protected members of the base class B become protected members of the derived class D.

Prob.5 How do structures in C and C++ differ?

[R.T.U. Dec. 2016]

Sol. The difference between structures in C and C++ are as follows:

- (i) C structures don't contain functions means only data member are allowed but the structures in C++ contain both functions and data members.
- (ii) Struct keyword is necessary in C to create structure type variable, but it is redundant and not necessary in C++.
- (iii) Size of empty structure is undefined behaviours in C but it is always 1 in C++.
- (iv) Structures in C don't have static member. but C++ structures have static members. For e.g. following program fails in compilation in C, but compiles in C++.

```
#include <stdio.h>
struct demo
{
    static int a;
};
int main()
{
    return 0;}

```

- (v) Structure member can't be directly initialized inside the structure in C but it is allowed in C++.
- (vi) We can have both pointers and references to struct in C++ but only pointers to structs are allowed.
- (vii) C++ also have .* and -> operator to access individual member of structure, but C does not have such kind of operator.
- (viii) Structure declaration establishes a scope in C++ and not in C which make member enumerators and nested structures possible in C++.

Prob.6 What is the use of making a method private inside a class? [R.T.U. 2009]

Sol. A private method of class is not accessible outside the class. Although it is a normal practice to place all the data items in a private section and all the functions in public, some situations may require certain functions to be hidden (like private data) from the outside class. Ex. For deleting an account in a customer file or providing increment to an employee etc., we can place these function in the private section.

A private member function can only be called by another function that is a member of its class. Even an object cannot invoke a private function using dot operator.

```
class Sample
{
    int m;
    void read(void);
    // private member function
public:
    void update(void);
    void write(void);
};

if s1, is an object of sample then
s1.read();
is illegal.
So it will be
void sample :: update(void)
{
    read(); //simple call no object
           //used.
}
```

Prob.7 What is the significance of using a protected visibility specifier to a class member? [Raj. Univ. 2002]

Sol. A member declared as protected is accessible by the member functions within its class and any class immediately derived from it. It can not be accessed by the functions outside these two classes.

When a protected member is inherited in public mode, it becomes protected in the derived class too and therefore is accessible by the member function of the derived class. It is also ready for further inheritance. A protected member, inherited in the private mode derivation, becomes private in the derived class. Although it is available to the member functions of the derived class, it is not available for further inheritance.

PART-B

Prob.8 Use for loops to construct a program that displays a pyramid on the screen. the pyramid should look like this

```
1
121
12321
1234321
123454321
```

[R.T.U. 2018]

Sol.

```
#include<stdio.h>
#include<conio.h>
main()
{
    clrscr();
    int i,j,n,k;
    printf("Enter a value :");
    scanf("%d",&n);
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=n-i;j++)
            printf(" ");
        for(j=1;j<=i;j++)
            printf("%d",j);
        j=1;
        for(j=i;j>=j;j--)
            if(j==0)
                break;
            else
            {
                k=j-1;
                if(k==0)
                    break;
                else
                    printf("%d",k);
            }
        printf("\n");
    }
    getch();
}
```

Output:

Enter a value : 5

```
1
121
12321
1234321
123454321
```

Prob.9 What do you mean by Programming Paradigm? Describe the concept of message passing in object Oriented Programming. [R.T.U. 2018]

Sol. Programming Paradigms

A programming paradigm is a model of programming based on distinct concepts that shapes the way of programmers design, organize and write programs. A multi-

(ii) Inside the member function of the same class, accessing data members and member functions is simple. We can directly use it without any (.) or function.

Sol. (b) Objects : Objects are the basic run-time entities, in an object-oriented system. They may represent a person, a place, a bank account, a table of data or any item that the program has to handle. They may also represent user-defined data such as vectors, time and lists. Programming problem is analyzed in terms of objects and the nature of communication between them. Program objects should be chosen such that they match closely with the real-world objects. Objects take up space in the memory and have an associated address like a record in Pascal or a structure in C.

When a program is executed, the objects interact by sending messages to one another. Objects are identified by their unique name.

Process of Creating Objects

An object can be created by using **new**, as when required. A data object created inside a block with **new**, will remain in existence until it is explicitly destroyed by using **delete**. Thus, the lifetime of an object is directly under our control and is unrelated to the block structure of the program.

The **new** operator can be used to create objects of any type. It takes the following general form:

```
pointer - variable = new data - type;
```

Here, pointer-variable is a pointer of type data type. The **new** operator allocates sufficient memory to hold a data object of type data-type and returns the address of the object. The data type may be any valid data type. The pointer variable holds the address of the memory space allocated.

Examples:

```
p = new int;
q = new float;
```

where **p** is a pointer of type **int** and **q** is a pointer of type **float**. Here **p** and **q** have must have already been declared as pointers of appropriate types.

Prob.11 Explain the concept of classes and object in OOPs.

[R.T.U. 2016]

Sol. Objects : Refer to Prob.10(b).

Classes : The entire set of data and code of an object can be made a user-defined data type with the help of a class. In fact, objects are variables of the type class. Once a class has been defined, we can create any number of objects belonging to that class. Each object is associated with the data of type class with which they are created. A class is thus a collection of objects of similar type. For example, mango, apple and orange are members of the class fruit. Classes are user-defined data types and behave like the built-in types of a programming language. The syntax used to create

an object is no different than the syntax used to create an integer object in C. If fruit has been defined as a class, then the statement **fruit mango;** will create an object **mango** belonging to the class **fruit**.

Prob.12 What is object oriented concept of programming?
[R.T.U. 2014]

Sol. Object-Oriented Programming (OOP) is a programming language model organized around "objects" rather than "actions" and data rather than logic. Historically, a program has been viewed as a logical procedure that takes input data, processes it and produces output data.

The programming challenge was seen as how to write the logic not how to define the data. Object-oriented programming takes the view that what we really care about are the objects we want to manipulate rather than the logic required to manipulate them. Examples of objects range from human beings (described by name, address and so forth) to buildings and floors (whose properties can be described and managed) down to the little widgets on your computer desktop (such as buttons and scroll bars). In object oriented programming we bind data and code together in a single unit. **The Three Main Concepts of OOPs are:** Polymorphism, Inheritance and Encapsulation.

The computer is adapted to the problem. Object-oriented programming is designed around the data being operated upon as opposed to the operation themselves.

Prob.13 Write short note on Composition of classes.

[R.T.U. 2014]

Sol. Composition of Classes: A class is a way to bind the data and its associated functions together. It allows the data (and functions) to be hidden, if necessary, from external use. When defining a class, we are creating a new abstract data type that can be treated like any other built-in data type. Generally, a class specification has two parts:

1. Class declaration
2. Class function definitions

The class declaration describes the type and scope of its members. The class function definitions describe how the class functions are implemented.

The general form of a class declaration is:

```
class class_name
{
    private:
        variable declarations;
        function declarations;
    public:
        variable declarations;
        function declarations;
};
```


Data hiding is the highly valued characteristic, that an object can be used without the user knowing or caring how it works internally. Just as you can open a file on your computer without knowing how the hard disk, memory or word processor application works, so using data hiding concept one can use a well-designed object without knowing about its internal data members. C++ supports the properties of encapsulation and data hiding through the creation of user-defined types, called classes.

Prob.18(a) Distinguish between the following terms :

- (i) **Object and classes** [R.T.U. 2011, 2010]
 (ii) **Data abstraction and data encapsulation.**

[R.T.U. 2011, 2010]

(b) **Write a simple program to access and manipulate data members in C++.**

[R.T.U. 2011]

Sol. (a) (i) Object : Refer to Prob.10(b).

Classes : Refer to Prob.11.

(ii) **Data Encapsulation**

- The wrapping up of data and functions into a single unit (called class) is known as encapsulation.
- Data encapsulation is the most striking feature of a class.
- The data is not accessible to the outside world, and only those functions which are wrapped in the class can access it. These functions provide the interface between the object's data and the program.
- This insulation of the data from direct access by the program is called data hiding or information hiding.

Data Abstraction

- Abstraction refers to the act of representing essential features without including the background details or explanations.
- Classes use the concept of abstraction and are defined as a list of abstract attributes such as size, weight and cost, and functions to operate on these attributes, they encapsulate all the essential properties of the objects that are to be created.
- The attributes are sometimes called data members because they hold information.
- The functions that operate on these data are sometimes called methods or member functions. Since the classes use the concept of data abstraction, they are known as Abstract Data Types (ADT).

Sol.(b) A typical class declaration look like:

```
class item
{
    int number; // variables declaration
    float cost; // private by default
public:
    void getdata (int a, float b);
        // functions declaration
    void putdata (void);
        // using prototype
};
        // ends with semicolon
```

We usually give a class some meaningful name, such as **item**. This name now becomes a new type identifier that can be used to declare instance of that class type. The class **item** contains two data members and two function members. The data members are private by default while both the functions are public by declaration. The function **getdata()** can be used to assign values to the member variables **number** and **cost**, and **putdata()** for displaying their values. These functions provide the only access to the data members from outside the class. This means that the data cannot be accessed by any function that is not a member of the class **item**.

Accessing and Manipulating Class Members

As pointed out earlier, the private data of a class can be accessed only through the member functions of that class. The **main()** cannot contain statements that access **number** and **cost** directly. The following is the format for calling a member function.

object-name.function-name (actual-arguments);

For example, the function call statement

x.getdata(100, 75.5);

is valid and assigns the value 100 to **number** and 75.5 to **cost** of the object **x** by implementing the **get-data()** function. The assignment occur in the actual function.

Similarly, the statement

x.putdata();

would display the values of data members, Remember, a member function can be invoked only by using an object (of the same class). The statement like

getdata (100, 75.5);

has no meaning. Similarly, the statement

x.number=100;

is also illegal. Although **x** is an object of the type **item** to which belongs, the **number** (declared private) can be accessed only through a member function and not by the object directly.

PREVIOUS YEARS QUESTIONS

PART-A

Prob.1 What is copy constructor? Explain with suitable example. [R.T.U. 2018]

Sol. Copy Constructors : We use the copy constructor integer (integer and i); as one of the overloaded constructor. A copy constructor is used to declare and initialize an object from another object. For example, the statement

```
integer I2 (I1);
```

would define the objects I2 and at the same time initialize it to the values of I1. Another forms of this statement is integer I2 = I1;

The process of initializing through a copy constructor is known as copy initialization. Remember, the statement

```
I2 = I1;
```

will not invoke the copy constructor. However, if I1 and I2 are objects, this statement is legal and simply assigns the values of I1 to I2, member - by - member. This is the task of the overloaded assignment operator (=).

A copy constructor takes a reference to an object of the same class to itself as an argument.

Example

```
# include <iostream.h>
using namespace std;
class code
{
    int id;
    public :
    code () { }
        // constructor
    code (int a) {id = a;}
        // constructor again
    code (code and x)
        // copy constructor
    {
        id = x.id;
        // copy in the value
    }
    void display (void)
    {
        cout << id;
    }
};
int main ( )
```

```
{
    code A(100);
        // object A is created and
        // initialized
    code B(A);
        // copy constructor called
    code C = A;
        // copy constructor called again
    code D;
        // D is created, not initialized.
    D = A; // copy constructor not called
    cout << "\n id of A :"; A . display();
    cout << "\n id of B :"; B. display();
    cout << "\n id of C :"; C. display();
    cout << "\n id of D:" ; D. display();
    return 0;
}
```

The output of program is :

```
id of A : 100
id of B : 100
id of C : 100
id of D : 100
```

When no copy constructor is defined, the compiler supplies its own copy constructor.

Prob.2 What are the differences between member function of a class and friend function? [R.T.U. 2018]

Sol. Difference between Friend Function and Member Function:

A member function (or member method) is a function that is declared to be a method of a class, such as a class accessor (getter), mutator (setter), or operation.

A friend function is a highly privileged, external function or a member method of another class, that is permitted full access to all the private and protected members of the class in which it is declared.

Friend functions should always be treated with caution because of their highly privileged status. There are often alternatives to using friend functions, but when one class works closely with another and requires access to some or all of its hidden members it may be the only way to allow the two to work together.

Prob.3 Discuss difference between Pointers & References in C++. [R.T.U. 2015]

Sol.

References	Pointers
A reference is a variable which refers to another variable. Ex. : int i = 3; int &ref = i; It defines a reference to the variable i.	A pointer is a variable which stores the address of another variable. Ex. : int i = 3; int *ptr = &i; It defines a pointer to that variable's memory address.
With a reference no operator is required.	With pointers must use the * operator to dereference it.
Pointers can be reassigned while references cannot. In other words, a reference cannot assigned to a different address.	Pointers can be reassigned. In other words, a pointer can be assigned to a different address.
References cannot be NULL.	Pointers are often made NULL to indicate that they are not pointing to any valid thing.
A reference must be initialized when declared.	There is no such restriction with pointers that it must be initialized when declared.
Members of an object reference can be accessed with dot operator ('.').	In pointers arrow operator (->) is needed to access members.

Prob.4 What is friend function?

Sol. Friend Function : One of the convenient and controversial feature of C++ is allowing non-member function to access even the private member of a class using friend functions or friend classes.

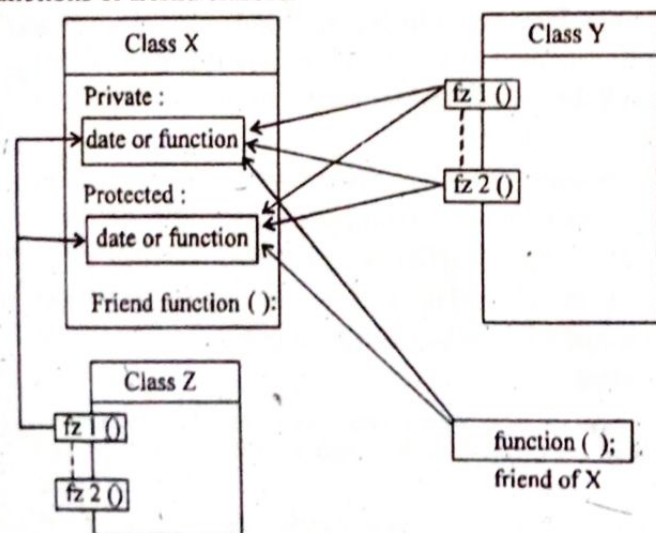


Fig.

Prob.5 Define 'this' pointer.

Sol. This Pointer : C++ uses a unique keyword called 'this' to represent an object that invokes a member function. 'this' is a pointer that points to the object for which it function was called. For example, the function call A.max() will set the pointer 'this' to the address of the object A. The starting address is the same as the address of the first variable in the class structure.

'this' unique pointer is automatically passed to a member function when it is called. The pointer 'this' acts as an implicit argument to all the member functions.

PART-B**Prob.6 Write short note on new and delete. [R.T.U. 2018]**

Sol. Program : Use of new and delete Operators

```
# include <iostream.h>
# include <conio.h>
using namespace std;
void main ()
{
    int *arr;
    int size;
    cout << "Enter the size of the
integer array: ";
    cin >> size;
    cout << "Creating an array of size"
<< size << "...";
    arr = new int[size];
    cout << "\nDynamic allocation of
memory for array arr is successful.
";
    delete arr;
    getch ();
}
```

Output

The output of this program would be:

Enter the size of the integer array: 5
Creating an array of size 5.

Prob.7 Design a class having the constructor and destructor functions that should display the number of objects being created or destroyed of class type.

[R.T.U. 2018]

Sol. #include <iostream>
using namespace std;
class Line {

malloc allocation. This is not required in C because the C standard allows implicit cast between void *, which is the type returned by malloc, and other pointer types. In fact, in C, casting malloc is considered undesirable. But if we want to use malloc in C++ code we must explicitly cast the pointer returned by malloc to the appropriate type. Actually the use of void* pointers in C++ is not recommended because it can break multiple inheritance, that is, in a multiple inheritance hierarchy some of the classes can view different values of the this pointer (the original value with some offset) and casting to void* can break the protection mechanisms of C++. If a void* cast is needed in this situation it is recommended to use a dynamic_cast or static_cast as soon as possible, to adjust the this pointer automatically.

The backward portability with malloc and free is useful in C++ for supporting legacy C code and to allow implementing (overloading) the new operator using calls to malloc. Because the operator new does more than just allocating memory (it also calls the object's constructor), it is not allowed to use free for data allocated with new or vice versa (delete with malloc).

The advantages of using 'New' over their older relatives malloc is the following:

- New point to the correct memory type (they are type safe), so casts are not necessary. But in malloc() casts are necessary.
- New invokes the constructor, allowing the initialization of objects, but If you use malloc, constructor do not get called.
- New figures out the size it needs to allocate, so the programmer doesn't have to specify it.
- New throws an exception of type std::bad_alloc when it fails, instead of only returning a NULL pointer like malloc, so the programmer doesn't have to write the deprecated check.

Prob.13 Discuss various use of :: operator in C++.

[R.T.U. 2015]

Sol. :: Operator: '::' is known as Scope Resolution Operator. C++ is a block structured language. Different program modules are written in various blocks. Same variable name can be used in different blocks. Scope of a variable extends from the point of declaration to the end of the block. A variable declared inside a block is 'local' variable. Blocks in C++ are often nested.

(i) The scope resolution operator :: is used to identify and disambiguate identifiers used in different scopes.

e.g.

```
....
{
    int x = 10;
    ....
    ....
}
```

Object Oriented Programming

```
{
    int x = 20;
}
```

The declaration of the inner block hides the declaration of same variable in outer block. This means, within the inner block, the variable x will refer to the data object declared therein. To access the global version of the variable, C++ provides scope resolution operator. When local variable and global variable are having same name, local variable gets the priority. C++ allows flexibility of accessing both the variables through a scope resolution operator.

In the above example, x has a value of 20 but ::x has value 10.

(ii) Similarly, this operator is used when a member function is defined outside the class

e.g.

Class MyClass

```
{
    int n1, n2;
public:
    void func1();           ...Function Declaration
};

public void MyClass::func1()
    ...Use of Scope Resolution Operator to write
    function definition outside class definition
{
    // Function Code
}
```

(iii) You must use the scope resolution operator to call static members of classes.

class ClassG

```
{
public:
    static int get_x()
    {
        return x;
    }
    static int x;
};
```

int ClassG::x = 6;

int main()

```
{
    int gx1 = ClassG::x;
    int gx2 = ClassG::get_x();
}
```

(iv) The scope resolution operator is also used with the values of a scoped enumeration. Enumeration Declarations, as in the following example:

enum class EnumA

```
{
    First,
    Second,
    Third
};
```

int main()

```
{
    EnumA enum_value = EnumA::First;
}
```

Prob.14 What is the use of 'this' pointer. Explain with example. [R.T.U. 2015]

OR

What is 'this' pointer? Explain how is 'this' pointer different from a pointer to an object?

[Raj. Univ. 2007, 2006, 2004]

Sol. This Pointer : Refer to Prob.4.

Consider the following simple example:

```
class ABC
{
    int a;
    ....
    ....
};
```

The private variable 'a' can be used directly inside a member function, like:

a = 123;

We can also use the following statement to do the same job:

this->a = 123;

Since C++ permits the use of shorthand form a = 123, we have not been using the pointer 'this' explicitly so far. However, we have been implicitly using the pointer this when overloading the operators using member function.

Recall that, when a binary operator is overloaded using a member function, we pass only argument to the function. The other argument is implicitly passed using the pointer this. One important application of the pointer this is to return the object it points to. For example, the statement

return *this;

inside a member function will return the object that invoked the function. This statement assumes importance when we want to compare two or more objects inside a member function and return the invoking object as a result. Example:

```
person & person :: greater(person & x)
{
    if x.age > age
        return x;           // argument object
    else
        return *this;       // invoking object
}
```


Suppose we invoke this function by the call
`max = A.greater(B);`

The function will return the object B (argument object) if the age of the person B is greater than A, otherwise, it will return the object A (invoking object) using the pointer `this`. T, the dereference operator `*` produces the contents at the address contained in the pointer.

Prob.15 What is function overloading? Explain with help of suitable example. [R.T.U. 2014, 2010]

Sol. Overloading : Overloading refers to the use of the same thing for different purposes. C++ also permits overloading of functions. This means that we can use the same function name to create other functions that perform a variety of different tasks. This is known as function polymorphism in OOP.

Function Overloading : Using the concept of function overloading, we can create a family of functions with one function name but with different argument lists. The function would perform different operations depending on the argument list in the function call. The correct function to be invoked is determined by checking the number and type of the arguments but not on the function type. For example, on overloading `add()` function handles different types of data as shown below :

```
// Declarations
int add (int a, int b);    // Prototype 1
int add (int a, int b, int c); // Prototype 2
double add (double X, double Y); // Prototype 3
double add (int p, double q); // Prototype 4
double add (double p, int q); // Prototype 5
// function calls
cout << add (5, 10);        // uses prototype 1
cout << add (15, 10.0);     // uses prototype 4
cout << add (12.5, 7.5);    // uses prototype 3
cout << add (5, 10, 15);    // uses prototype 2
cout << add (0.75, 5);      // uses prototype 5
```

A function call first matches the prototype having the same number and type of arguments and then calls the appropriate function for execution. A best match must be unique. The selection of function involves the following steps :

1. The compiler first tries to find an exact match in which the types of actual arguments are the same, and uses that function.
2. If an exact match is not found, the compiler uses the integral promotions to the actual arguments, such as, char to int, float to double, to find a match.
3. When either of them fails, the compiler tries to use the built - in conversions (the implicit assignment conversion) to the actual arguments and then uses the function whose match is unique. If the

conversion is possible to have multiple matches, then the compiler will generate an error message. Suppose we use the following two functions :

```
long square (long x);
double square (double x);
```

A function call such as

```
square (10);
```

will cause an error because `int` argument can be converted to either `long` or `double`, thereby creating an ambiguous situation as to which version of `square()` should be used.

4. If all the steps fail, then the compiler will try the user - defined conversions in combination with integral promotions and built - in conversions to find a unique match. User-defined conversion are often used in handling class objects.

Program : Function Overloading

```
// function volume () is overloaded three times
#include <iostream.h>
using namespace std;
// Declarations (prototype)
int volume (int);
double volume (double, int);
long volume (long, int, int);
int main()
{
    cout << volume (10) << "\n";
    cout << volume (2.5, 8) << "\n";
    cout << volume (1001, 75, 15) << "\n";
    return 0;
}
// function definitions
int volume (int s) // cube
{
    return (s * s * s);
}
double volume (double r, int h) // cylinder
{
    return (3.14519 * r * r * h);
}
long volume (long l, int b, int h)
// rectangular box
{
    return (l * b * h);
}
```

Output :

```
1000
157.25
112500
```

Overloading of the function should be done with caution. We should not overload unrelated functions and should reserve function overloading for functions that perform closely related tasks. Sometimes, the default arguments may be used instead of overloading. This may reduce the number

of functions to be defined, thereby making the code easier to read.

Overloaded functions are extensively used for handling class objects.

Prob.16 Explain mechanism of accessing a private data member of the class from a non member function and justify it with a program in C++ to swap private data members of two classes. [R.T.U. 2013]

Sol. The central idea of encapsulation and data hiding concept is that any non-member function has no access permission to the private data of the class. The private members of the class are accessed only from member functions of that class. Any non-member function cannot access the private data of the class.

C++ allows a mechanism, in which a non-member function has access permission to the private members of the class. This can be done by declaring a non-member function friend to the class whose private data is to be accessed. Here `friend` is a keyword. Consider the following example:

```
class ac
{
private:
    char name [15];
    int acno;
    float bal;
public:
    void read();
    friend void showbal ();
};
```

The keyword `friend` must precede the function declaration whereas function declarator must not. The function can be defined at any place in the program like normal function. The function can be declared as friend function in one or more classes. The keyword `friend` or scope access operator must not precede the definition of friend function. The declaration of friend function is done inside the class in private or public part and a function can be declared as friend function in any number of classes. These functions use objects as arguments.

Program to swap private data members of two classes

```
# include <iostream.h>
# include <conio.h>
Class b;
```


PART-C

Prob.17 Write a C++ program which will implement '==' binary operators. The overload class should contain one parameterized constructor to initialize default value of the member variable, and a display function to display value of member variables. [R.T.U. 2018]

Sol. Implementation on Comparison Operator (==) Using Parameterized Constructor:
class Point

```
{
private:
    double m_dX, m_dY, m_dZ;
public:
    Point(double dX=0.0, double dY=0.0, double dZ=0.0)
    {
        m_dX = dX;
        m_dY = dY;
        m_dZ = dZ;
    }
    friend bool operator==(Point &cP1, Point &cP2);
    friend bool operator!=(Point &cP1, Point &cP2);
    double GetX() { return m_dX; }
    double GetY() { return m_dY; }
    double GetZ() { return m_dZ; }
};
bool operator==(Point &cP1, Point &cP2)
{
    return (cP1.m_dX == cP2.m_dX &&
            cP1.m_dY == cP2.m_dY &&
            cP1.m_dZ == cP2.m_dZ);
}
bool operator!=(Point &cP1, Point &cP2)
{
    return !(cP1 == cP2);
}
```

Prob.18 Write short note on constructors and destructors in C++. [R.T.U. 2017]

OR

(a) What is constructor? How to invoke a constructor function? With an example distinguish between parameterized constructor and copy constructor.
(b) Describe the importance of destructor function. Write a program of dynamic memory management using new & delete operators. [R.T.U. 2016]

Sol. (a) Constructors : A constructor is special member function whose task is to initialize the objects of its class. It is special because its name is same as the class name. The constructor is invoked whenever an object of its associated class is created. It is called constructor because it constructs the value of data members of the class or we can say that it constructs the objects of a class. A constructor is declared and defined as follows:

```
#include<iostream.h>
class demo
{
    public
    demo()
    {
        cout <<"Hello Constructure \n";
    }
};
void main()
{
    demo d;
```

Output :

Hello constructor

The name of the class is demo and the following declaration

```
demo ()
{
    count<< "Hello constructor \n";
}
```

is a constructor of the class

A constructor that accepts no parameters is called a default constructor. The default constructor for class A is A::A(). If no such constructor is defined, then the compiler supplies a default constructor.

Therefore a statement such as

A a;

invokes the default constructor of the compiler to create the object a.

Parameterized Constructors : The constructors that can take arguments are called parameterized constructors. The constructor integer () initializes the data members of all the objects to zero. However, in practice it may be necessary to initialize the various data elements of different objects with different values when they are created. C++ permits us to achieve this objective by passing arguments to the constructor function when the objects are created. The constructor integer () may be modified to take arguments as shown below :

```
class integer
{
    int m, n;
    public :
    integer (int x, int y);
    // parameterized constructor
    : : :
```

```
integer : : integer (int x, int y)
{
    m = x; n = y;
}
```

when a constructor has been parameterized, the object declaration statement such as

```
integer int1;
```

may not work. We must pass the initial values as arguments to the constructor function when an object is declared. This can be done in two ways:

- By calling the constructor explicitly.
Integer int 1 = integer (0, 100); // explicit call
- By calling the constructor implicitly.
Integer int 1 (0, 100); // implicit call.

Copy Constructors : Refer to Prob.1.

Sol. (b) Destructor Function : A destructor, as the name implies, is used to destroy the objects that have been created by a constructor. Like a constructor, the destructor is a member function whose name is the same as the class name but is preceded by a tilde (~). This sign distinguishes constructor and destructor because both have same name as of their respective class.

The destructors are used to destroy the resources allocated to an object by constructor.

A destructor neither takes any argument nor does it return any value. It is invoked implicitly by the compiler upon exit from the program to clean up storage that is no longer accessible. It is a good practice to declare destructors in a program since it release memory space for further use.

Program : Use of new and delete Operators : Refer to Prob.6.

Prob.19 What are Friend functions in C++? Explain characteristics of Friend functions. [R.T.U. 2014, R.U. 2006]

OR

What is friend function and friend class. Discuss with suitable C++ code. [R.T.U. 2015]

Sol. Friend function : Refer to Prob.4.

Characteristics of friend function:

1. The scope of a friend function is not limited to the class in which it has been declared as a friend.
2. A friend function can not be called using the object of that class which is not in the scope of that class. It can be invoked like a normal function without the help of any object.
3. Unlike class member function it can not access the class member directly. However it can use the object and the dot operator with each member name to access both private and public members.
4. It can be either declared in the private part or the public part of a class without affecting its meaning.

The function can be allowed explicitly to access A's data and class B members can be allowed to access the class A's data. To accomplish this, the offending classless function `increment()` and the class B are declared to be friends of the class A as illustrated in the following code:

```
class A
{
public:
    friend class B;
    // B is my friend, I trust him
    friend void decrement (A &what);
    // decrement ( ) is also a good pal
};
```

Prob.20 What are classes and objects? Explain the process of creating and destroying objects dynamically using "new" and "delete" operators. [R.T.U. 2014]

Sol. Classes Refer to Prob. 11, Unit-1.

Objects : Refer to Prob.10(b), Unit-1.

Process of Creating and Destroying Objects : Refer to Prob.9.

Prob.21 (a) What is constructor? How many types of constructors are in C++? Describe with the help of a suitable example. [R.T.U. 2013, 2010]

(b) Explain the concept of inline function in C++. How it is beneficial in programming? [R.T.U. 2013, R.U. 2007, 2004]

Sol.(a) Constructors : Refer to Prob.18(a).

There are four types of constructors in C++:

- Parameterized Constructors
- Constructors with Default Arguments.
- Copy Constructors.
- Dynamic Constructors.

(i) Parameterized Constructors : Refer to Prob.18(a).
(ii) Constructor with Default Arguments : It is possible to define constructors with default arguments. For example, the constructor `complex()` can be declared as follows:

```
complex (float real, float imag = 0);

The default value of the argument imag is zero. Then, the statement complex c(5.0); assigns the value 5.0 to the real variable and 0.0 to imag (by default).
```

However, the statement

```
complex c(2.0, 3.0);
```

assigns 2.0 to real and 3.0 to imag. The actual parameter, when specified, overrides the default value. As pointed out earlier, the missing arguments must be the trailing ones.

It is important to distinguish between the default constructor `A::A()` and the default argument constructor `A::A(int = 0)`. The default argument constructor can be

called with either one argument or no arguments. When called with no arguments, it becomes a default constructor. When both these forms are used in a class, it causes ambiguity for a statement such as

A. a;

The ambiguity is whether to call `A::A()` or `A::A(int=0)`.

(iii) Copy Constructors : Refer to Prob.1.

Benefits :

- It is used as an overloaded constructor.
- It is used to declare and initialize an object from another object.
- It takes a reference to an object of the same class to itself as an argument.

(iv) Dynamic Constructors : The constructors can also be used to allocate memory while creating objects. This will enable the system to allocate the right amount of memory for each object when the objects are not of the same size, thus resulting in the saving of memory. Allocation of memory to objects at the time of their construction is known as dynamic construction of objects. The memory is allocated with the help of the new operator.

Following program shows the use of new, in constructors that are used to construct strings in objects.

```
#include<iostream.h>
using namespace std;
class String
{
    char *name;
    int length;
public:
    String() // constructor 1
    {
        length = 0;
        name = new char [length + 1];
    }
    String(char *s) // constructor 2
    {
        length = strlen(s);
        name = new char[length + 1];
        // one additional
        // character for \0
        strcpy(name, s);
    }
    void display(void)
    {
        cout<<name<<"\n";
    }
    void join(String &a, String &b);
};

void String::join(String &a, String &b)
{
    length = a.length + b.length;
    delete name;
    name = new char[length+1]; // dynamic
```

allocation

```
strcpy(name, a.name);
strcpy(name, b.name);
};
```

```
int main()
```

```
{
    char *first = "Joseph";
    String name1(first), name2("Louis"),
```

```
name3("Lagrange"), s1, s2;
```

```
s1.join(name1, name2);
```

```
s2.join(s1, name3);
```

```
name1.display();
```

```
name2.display();
```

```
name3.display();
```

```
s1.display();
```

```
s2.display();
```

```
return 0;
```

```
}
```

The output of this program would be

Joseph

Louis

Lagrange

Joseph Louis

Joseph Louis

Joseph Louis Lagrange

Joseph Louis Lagrange

Joseph Louis Lagrange

Joseph Louis Lagrange

Joseph Louis Lagrange

Joseph Louis Lagrange

Joseph Louis Lagrange

Joseph Louis Lagrange

Joseph Louis Lagrange

Joseph Louis Lagrange

Joseph Louis Lagrange

Joseph Louis Lagrange

Joseph Louis Lagrange

Joseph Louis Lagrange

Joseph Louis Lagrange

Joseph Louis Lagrange

Joseph Louis Lagrange

Joseph Louis Lagrange

Joseph Louis Lagrange

Joseph Louis Lagrange

Joseph Louis Lagrange

Joseph Louis Lagrange

Joseph Louis Lagrange

Joseph Louis Lagrange

Joseph Louis Lagrange

Joseph Louis Lagrange

Joseph Louis Lagrange

Joseph Louis Lagrange

Joseph Louis Lagrange

Joseph Louis Lagrange

Joseph Louis Lagrange

Joseph Louis Lagrange

Joseph Louis Lagrange

Joseph Louis Lagrange

Joseph Louis Lagrange

Joseph Louis Lagrange

Example:

```
inline double cube(double a)
```

```
{
    return (a*a*a);
}
```

```
The above inline function can be invoked by statements like
```

```
c = cube(3.0);
```

```
d = cube(2.5 + 1.5);
```

```
On the execution of these statements, the values of c
```

```
and d will be 27 and 64 respectively. If the arguments are
```

```
expressions such as 2.5 + 1.5, the function passes the value
```

```
of the expression, 4 in this case. This makes the inline feature
```

```
far superior to macros.
```

```
It is easy to make a function inline. We need to do is to
```

```
prefix the keyword inline to the function definition. All inline
```

```
functions must be defined before they are called.
```

```
The speed benefits of inline functions diminish as the
```

```
function grows in size. At some point the overhead of the
```

```
function call becomes small compared to the execution of
```

```
the function and the benefits of inline functions may be lost.
```

```
In such cases, the use of normal functions will be more
```

```
meaningful. Usually, the functions are made inline when they
```

```
are small enough to be defined in one or two lines.
```

```
Example:
```

```
inline double cube(double a)
```

```
{
    return (a*a*a);
}
```

```
Remember that the inline keyword merely sends a request,
```

```
not a command, to the compiler. The compiler may ignore
```

```
this request if the function definition is too long or too
```

```
complicated and compile the function as a normal function.
```

```
Some of the situations where inline expansion may not
```

```
work are:
```

- For functions returning values, if a loop, a switch, or a goto exists.
- For functions not returning values, if a return statement exists.
- If functions contain static variables.
- If inline functions are recursive.

Benefits of Inline Function

- It is beneficial where small code segment size functions are used and code does not involve recursion, looping etc.
- It is easy to make a function inline.
- This is used to eliminate the cost of calls to small functions in C++.
- This functions works faster than other function.

PREVIOUS YEARS QUESTIONS

PART-A

Prob.1 Write a C++ program to implement pure virtual function.

[R.T.U. 2018]

Sol. Virtual functions : Virtual functions defined inside the base class normally serve as a framework for future design of the class hierarchy; these functions can be overridden by the methods in the derived classes. In most of the cases, these virtual functions are defined with a null-body; it has no definition. Such functions in the base class are similar to doing nothing or dummy functions and in C++, they are called pure virtual functions. The syntax of defining pure virtual functions is shown in Fig. Pure virtual function is declared as a virtual function with its declaration followed by = 0.

```
Class MyClass
{
    public:
    ...
    ...
    //keyword
    Virtual Return Type FunctionName
    (Arguments) = 0;
    ...
    ...
};
```

Example : Syntax of pure virtual function

Prob.2 Write short note on abstract data type.

Sol. Abstract Data Types : Abstract Data type (ADT) is a type (or class) for objects whose behavior is defined by a set of value and a set of operations.

The definition of ADT only mentions what operations are to be performed but not how these operations will be implemented. It does not specify how data will be organized in memory and what algorithms will be used for implementing the operations. It is called "abstract" because it gives an implementation independent view. The process of providing only the essentials and hiding the details is known as abstraction.

Prob.3 What is inheritance?

Sol. Inheritance : Inheritance is an important feature of object oriented programming, it is a way to arrange information in

hierarchical manner. Inheritance allows us to reuse the code by creating collection of attributes and behaviors called objects which can be based on previously created objects. In other words we can say that in inheritance a new developed class uses the attributes and behaviors of previously developed and debugged class. The advantage of inheritance is that we do not have to write similar code again and again. We can simply inherit any class into another class. It saves a lot of time and human efforts of developing and testing the same code again.

PART-B

Prob.4 Write short note on function overriding.

[R.T.U. 2018]

Sol. Function Overriding : In method overriding a base class method is overridden in the derived class. That is the same method is written with the same signature as of the base class method but different implementation. Method overriding is often used in inheritance to override the base class implementation of the method and providing its own implementation. In method overloading, arguments and type of arguments are used to distinguish between two functions, but in method overriding no such distinction can be made. In method overriding the signature of the two methods must match.

Example of method overriding

```
#include<iostream.h>
class A
{
    public:
    void show()
    {
        cout<<"hello from A \n";
    }
};
class B : public A
{
    public:
    void show()
    {
        cout<<"Hello from B";
    }
};
void main()
{
    B obj;
    obj.show();
}
```


Protected Visibility Mode

When we use protected as visibility mode of base class then all public and protected members of base class become protected members in derived class.

Explanation

Protected visibility mode is used to inherit base class members in derived class. As we know when we use protected visibility mode all protected and public members of base class becomes protected members in derived class. So we cannot access those functions which become protected in base class and compiler generates compile time error.

Public Visibility Mode

When the visibility mode for base class is public, all public members of base class become public members of the derived class and all protected members of the base class become protected members of the derived class.

Explanation

In this, public members of base class become public member of derived class; similarly if we have used protected members in base class then they will become protected members of derived class. Here, in main() we have called four function two are defined in public part of base class and two are defined in public part of derived class and have used public visibility mode. So all four functions work properly and produce the proper results.

Note : In all the above cases, the base class's private members remain private to the base class and are not accessible by the members of the derived class.

Prob.7 Write short notes on Multiple inheritance.

[R.T.U. Dec. 2016]

OR

Design two classes to show the multiple inheritance. How does a object of derived class access the member of base class?

[R.T.U. 2017]

Sol. Multiple Inheritance : A class can inherit the attributes of two or more classes as shown in Fig. This is known as multiple inheritance. Multiple inheritance allows us to combine the features of several existing classes as a starting point for defining new classes. It is like a child inheriting the physical features of one parent and the intelligence of another.

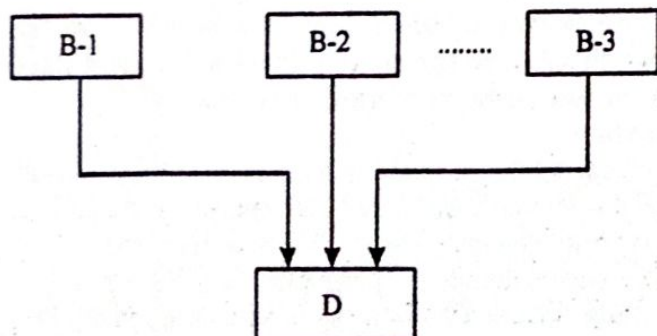


Fig. : Multiple inheritance

The syntax of a derived class with multiple base classes is as follows:

```

class D: visibility B-1, visibility B-2 ...
{
    .....
    ..... (Body of D)
};
  
```

where, visibility may be either *public* or *private*. The base classes are separated by commas.

Example:

```

class P : public M, public N
{
    public :
    void display(void);
};
  
```

// Classes M and N have been specified as follows:

```

class M
{
    protected:
    int m;
    public :
    void get_m(int);
    void M :: get_m(int x)
};

void M :: get_m(int x)
{
    m = x;
}

class N
{
    protected:
    int n;
    public:
    void get_n(int);
    void N :: get_n(int y)
    {
        n = y;
    }
};
  
```

The derived class P, as declared above, would, in effect, contain all the members of M and N in addition to its own members as shown below:

```

class P .
{
    protected:
    int m;           // from M
    int n;           // from N
    public :
    void get_m(int);  // from M
    void get_n(int);  // from N
    void display(void); // own member
};
  
```


//The member function display() can be defined as follows:

```
void P::display(void)
{
    cout << "m = " << m << "\n";
    cout << "n = " << n << "\n";
    cout << "m*n = " << m*n << "\n";
}
```

The main() function which provides the user-interface may be written as follows:

```
main()
{
    P p;
    p.get_m(10);
    p.get_n(20);
    p.display();
}
```

Prob.8 Explain the concept of base class and derived class. [R.T.U. Dec. 2016]

Sol. Base Class and Derived Class

A base class is a class from which other classes are inherited. A class which has common features that can be used by another class can become base class. A base class can be inherited into more than one class or we can say more than one class can have one base class.

The class that reuses the features of previously developed class is called derived class. A derived class has features of base class and it also has features of its own. A derived class can serve as base class for another derived class.

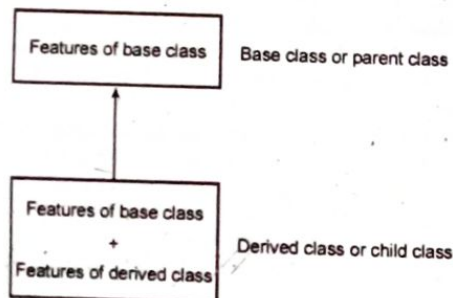


Fig.: Relationship of base and derived class in inheritance

Following notation is required to define a derived class:

```
class Derived_Class : [VisibilityMode]
Base_Class
{
    //body of derived class
}
```

The derivation of derived class from the base class is shown by colon (:) sign. The visibility mode can be private, protected or public. The default visibility mode is private.

Visibility mode specifies the access control on features of base class that are to be inherited in derived class.

Prob.9 (a) When do we make an abstract class? Explain in detail.

(b) Describe how an object of a class that contains object of other classes created? [R.T.U. Dec. 2016]

Sol.(a) Abstract Class : Refer to Prob.5.

Sol.(b) With the help of inheritance an object of a class that contains object of other classes created. Inheritance is most important concept in OOPs. Inheritance allows to define a class in terms of another class, which makes it easier to create and maintain an application. When creating a class, instead of writing completely new data members, member functions, objects, the programmer can designate the new class should inherit the members of existing class.

```
#include <iostream>
using namespace std;
class person
{
public:
    string profession;
    int age;
    Person(): profession("unemployed"), age(16){}
    void display()
    {
        cout<<"My profession is:"<<profession<<endl;
        cout<<"My age is:"<<age<<endl;
        walk();
        talk();
    }
    void walk(){cout<<"I can walk."<<endl;}
    void talk(){cout<<"I can talk."<<endl;}
};
```

//Maths Teacher class is derived from base class Person.

```
class MathsTeacher : public Person
{
public:
    void teachMaths(){cout<<"I can teach Maths."<<endl;}
};
```

//Footballer class is derived from base class Person.

```
class Footballer : public person
{
public:
    void playfootball(){cout<<"I can play Football."<<endl;}
};
```

```
int main()
```

```
{
    MathsTeacher teacher;
```

```
teacher.profession = "Teacher";
teacher.age = 23;
teacher.display();
teacher.teachMaths();
Footballer footballer;
footballer.profession = "Footballer";
footballer.age = 19;
footballer.display();
footballer.playFootball();
return 0;
```

Output

```
My profession is : Teacher
My age is : 23
I can walk.
I can talk.
I can teach Maths.
My profession is: Footballer
My age is: 19
I can walk.
I can talk.
I can play Football.
```

Prob.10 What does inheritance mean in C++? How can we make private member inheritable without modifying.

[R.T.U. 2016]

Sol. Inheritance : Refer to Prob.3.

Making a private member inheritable : If we want to inherit private data of class by its derived class, then we modify the visibility limit of the private member by making it public. This would make it accessible to all other functions of the program, thus taking away the advantage of data hiding.

C++ provides third visibility modifier, *protected*, which serves a limited purpose in inheritance. A member declared as *protected* is accessible by the member functions within its class and any class immediately derived from it. It cannot be accessed by the function outside these two classes. A class can now use all the three visibility modes as shown below :

```
class alpha
{
private:
    -----
    -----
protected :
    -----
    -----
public:
    -----
    -----
};
```



```

~b()
{
    printf("\nDerived Destructor\n");
}
};
int main()
{
    a* obj=new b;
    delete obj;
    return 0;
}

```

Output: Base Constructor
Derived Constructor
Base Destructor
By Changing-
~a()

```

{
    printf("\nBase Destructor\n");
}
to
virtual ~a()
{
    printf("\nBase Destructor\n");
}

```

Output: Base Constructor
Derived Constructor
Derived Destructor
Base Destructor

Prob.12 What is a virtual base class? Explain.

[R.T.U. Dec. 2016]

Sol. Virtual Base Classes: Consider a situation where all the three kinds of inheritance, namely multilevel, multiple and hierarchical inheritance, are involved. This is illustrated in Fig.1. The 'child' has two direct base classes 'parent1' and 'parent2' which themselves have a common base class 'grandparent'. The 'child' inherits the traits of 'grandparent' via two separate paths. It can also inherit directly as shown by the broken line. The 'grandparent' is sometimes referred to as indirect base class.

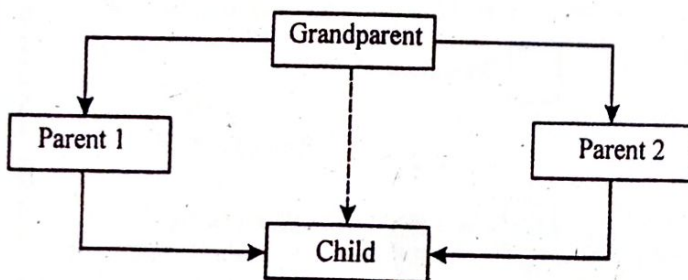


Fig.1 : Multipath inheritance

Inheritance by the 'child' as shown in Fig.1 might pose some problems. All the public and protected members of

'grandparent' are inherited into 'child' twice, first via 'parent1' and again via 'parent2'. This means, 'child' would have duplicate sets of the members inherited from 'grandparent'. This introduces ambiguity and should be avoided.

The duplication of inherited members due to these multiple paths can be avoided by making the common base class (ancestor class) as virtual base class while declaring the direct or intermediate base classes as shown below :

```

class A // grandparent
{
    .....
    .....
};
class B1 : virtual public A // parent1
{
    .....
    .....
};
class B2 : public virtual A // parent2
{
    .....
    .....
};
class C: public B1, public B2 // child
{
    ..... // only one copy of A
    ..... // will be inherited
};

```

When a class is made a virtual base class, C++ takes necessary care to see that only one copy of that class is inherited, regardless of how many inheritance paths exist between the virtual base class and a derived class.

Suppose we have two derived classes B and C that have a common base class A and you also have another class D that inherits from B and C. We can declare the base class A as virtual to ensure that B and C share the same sub object of A. In the following example, an object of class D has two distinct sub objects of class L, one through class B1 and another through class B2. We can use the keyword virtual in front of the base class specified in the base lists of classes B1 and B2 to indicate that only one sub object of type L, shared by class B1 and class B2, exists. For example:

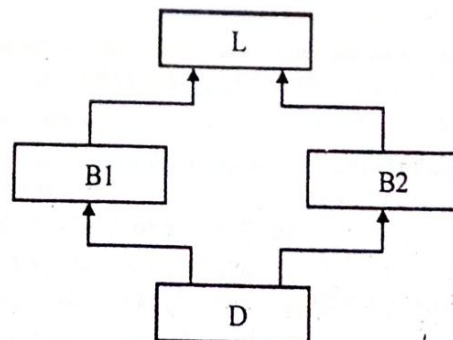


Fig.2

class L { /* */ }; // indirect base class
 class B1: virtual public L { /* */ };
 class B2: virtual public L { /* */ };
 class D: public B1, public B2 { /* */ };
 Using the keyword virtual in this example ensures that an object of class D inherits only one sub object of class L. A derived class have both virtual and non virtual base classes.
 For example:

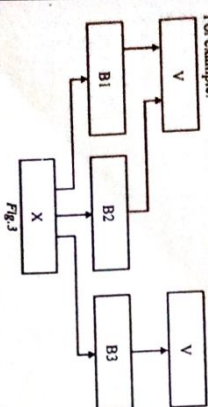


Fig.3

```

class V
{
    /* */
};

class B1: virtual public V
{
    /* */
};

class B2: virtual public V
{
    /* */
};

class X: public B1, public B2, public B3
{
    /* */
};
  
```

In the above example, class X has two sub objects of class V, one that is shared by classes B1 and B2 and one through class B3.

Prob.13 Write short note on Type of Inheritance.

(R.T.U. 2015)

Sol. Forms of Inheritance : Inheritance is a technique through which we can reuse the properties of already developed class in a new class. To fulfill this reusability C++ support different form of inheritance through which we can share properties among classes. These forms define the levels of inheritance. It may be possible that a base class is derived from another base class in the inheritance hierarchy.

Different Forms of Inheritance :

The following are different forms of inheritance:

1. **Single Inheritance :** In this type of inheritance only two classes take part. One is base class and another is derived class. Properties of base class are inherited in derived class.

```

class A
{
    //body of base class
};

class B: public A
{
    //body of derived class
};
  
```

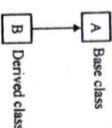


Fig. 1 : Single Inheritance

2. **Multilevel Inheritance :** In multilevel inheritance derived class becomes base class for another derived class. More than two classes are involved in this type of inheritance.

```

class A
{
    //body of base class
};

class B: public A
{
    //body of derived class
};

class C: public B
{
    //body of derived class
};
  
```

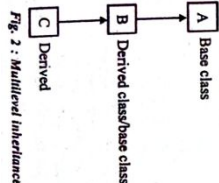


Fig. 2 : Multilevel Inheritance

Object Oriented Programming

3. **Multiple Inheritance :** When derived class is form through more than one base classes then this type of inheritance is called multiple inheritance.

```

class A
{
    //body of first base class
};

class B
{
    //body of second base class
};

class C: public A, public B
{
    //body of derived class
};
  
```

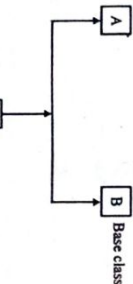


Fig. 3 : Multiple Inheritance

4. **Hierarchical Inheritance :** When one class becomes base class for more than one derived classes or we can say one class is inherited by several classes it is called hierarchical inheritance.

```

class A
{
    //body of base class
};

class B: public A
{
    //body of first derived class
};

class C: public A
{
    //body of second derived class
};
  
```

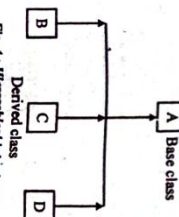


Fig. 4 : Hierarchical Inheritance

5. **Multipath Inheritance :** Derivation of a class from other derived classes which are derived from the same base class is called multipath inheritance. In this type of inheritance we use the concept of virtual base class to prevent ambiguity.

```

class A
{
    //body of base class
};

class B: virtual public A
{
    //body of first derived class
};

class C: virtual public A
{
    //body of second derived class
};

class D: public B, public C
{
    //body of class D derived from B & C
};
  
```

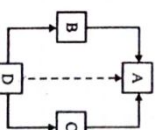


Fig. 5 : Multipath Inheritance

6. **Hybrid Inheritance :** When a class is involved in more than one form of inheritance then it is known as hybrid inheritance.

```

class A
{
    //body of class A
};

class B: public A
{
    //body of class B
};

class C:
{
    //body of class C
};

class D: public B, public C
{
    //body of class D
};
  
```

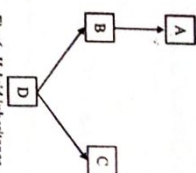


Fig. 6: Hybrid Inheritance.

Prob.14 Write short notes

(a) Abstract class

(b) Function overloading and function over riding.

(R.T.U. 2014)

(a) **Abstract Class: Refer to Prob.5**

(b) **Function Overloading :** Using the concept of function overloading, we can create a family of functions with one function name but with different argument lists. The function would perform different operations depending on the argument list in the function call. The correct function to be called is determined by checking the number and type of the arguments but not on the function type. For example, on overloading add () function handles different types of data shown below :

```

// Declarations
int add (int a, int b); // Prototype 1
double add (double X, double Y); // Prototype 2
double add (int p, double q); // Prototype 3
double add (double p, int q); // Prototype 4

// Function calls
cout << add (5, 10); // uses prototype 1
cout << add (15, 10.0); // uses prototype 2
cout << add (12.5, 7.5); // uses prototype 3
cout << add (5, 10, 15); // uses prototype 4
cout << add (0.75, 5); // uses prototype 5
  
```

A function call first matches the prototype having the same number and type of arguments and then calls the appropriate function for execution. A best match must be unique. The selection of function involves the following steps :

1. The compiler first tries to find an exact match in which the types of actual arguments are the same, and uses that function.
2. If an exact match is not found, the compiler uses the integral promotions to the actual arguments, such as, char to int.
3. float to double.

to find a match.

When either of them fails, the compiler tries to use the built-in conversions (the implicit assignment conversion) to the actual arguments and then uses the function whose match is unique. If the conversion is possible to have multiple matches, the

Table

Base class visibility	Derived class visibility		
	Public	Private	Protected
Public	derivation	derivation	derivation
Private	Non inherited	Non inherited	Non inherited
Protected	Protected	Private	Protected
Public	Public	Private	Protected

Prob.20 (a) What is inheritance? What are the different terms of inheritance supported by C++? Explain each with suitable example.

(b) What is visibility mode? What are different inheritance visibility modes supported by C++? [R.T.U. 2013]

Sol. (a) Refer to Prob.3 and Prob. 13.

Sol. (b) Refer to Prob.6.

Prob.21 (a) What is pure virtual function? What are the implications of making function pure virtual function?

(b) What are abstract classes? Explain the role of abstract class while building the class hierarchy. [R.T.U. 2013]

Sol. (a) Virtual functions : Refer to Prob.1.

A pure virtual function declared in a base class has no implementation as far as the base class is concerned. The classes derived from a base class having a pure virtual function have to define such a function or redeclare it as a pure virtual function. It must be noted that, a class containing pure virtual functions cannot be used to define any objects of its own and hence such classes are called pure abstract classes or simply abstract classes. Whereas all other classes without pure virtual functions and which are instantiated are called as concrete classes.

A pure virtual function is an unfinished placeholder that the derived class is expected to complete.

The following are the properties of pure virtual functions:

- A pure virtual function has no implementation in the base class hence, a class with pure virtual function cannot be instantiated.

- In acts as an empty bucket (virtual function is a partially filled bucket) that the derived class is supposed to fill.

- A pure virtual member function can be invoked by its derived class.

The concept of abstract class (A class with pure virtual function) is necessary in order to understand pure virtual

B.Tech. (III Sem.) C.S. Solved Papers

functions and it is illustrated in the program pure.cpp. Note that a class with one or more pure virtual functions cannot be instantiated.

```
//pure.cpp: pure virtual function with
abstract class
# include <iostream.h>
class Absperson
{
public:
virtual void Service1 (int n); //normal
virtual void Service2 (int n) = 0; //
pure virtual member function
};
void Absperson::Service1(int n);
Service2(n);
Class Person : Public Absperson
{
public:
void Service2(int n);
void Person::Service2(int n);
cout<<"The number of years of service:"
<<(58 - n) <<endl;
void main()
{
Person Father, Son;
Father.Service1(50)
Son.Service2(20);
}
```

```
Run
The number of years of service1: 8
The number of years of service2: 38
In main(), the statement
Father.Service1(50);
invokes the virtual function Service1() defined in the
class Absperson and this in turn invokes Service2(). The
service2() of the class Person is invoked instead of Absperson;
it is declared as a pure virtual function.
Sol.(b) In order to exploit the potential benefits of inheri-
tance, the base classes are improved or enhanced without
modifications, which results in a derived class or inherited
class. The objects created often are the instances of a
```

Object Oriented Programming

derived class but not of the base class. The base class becomes just the foundation for building new classes and hence such classes are called abstract base classes or abstract classes. An abstract class is one that has no instances and is not designed to create objects. An abstract class is only designed to be inherited. It specifies an interface at a certain level of inheritance and provides a framework, upon which other classes can be built.

```
// stack.cpp: Overloading of functions
in base and derived classes
# include <iostream.h>
const int MAX_ELEMENTS = 5; // maximum
size of stack, you can change this
class Stack // base class
{
protected: // Note: cannot be private
int stack (MAX_ELEMENTS + 1); // for
stack[1]... stack(MAX_ELEMENTS)
int StackTop; // It points to current
stack top element
public:
Stack()
StackTop = 0; // Initially no elements
in stack, stack empty
void push (int element)
++StackTop; //Update StackTop for new
entry
stack[StackTop] = element; //put element
into the stack
void pop (int element)
element = Stack (StackTop);
--StackTop //update StackTop to point to next
element
// derivation of a new class from the class
Stack
Class MyStack: public Stack
{
public:
int push(int element) //return 1, if suc-
cess, 0 otherwise
```

```
Stack: :push(element); //calls base class
push
return 1; //push successful
cout <<"stack overflow"<< endl;
return 0; //stack overflow
int pop (int element) // return 1, if
success, 0 otherwise
{
if (StackTop > 0) // if stack is not full
{
Stack::pop (element); // calls base class
pop
return 1; //pop successful
}
cout <<, "Stack Underflow" << endl;
return 0; //stack underflow
}
void main()
{
MyStack stack;
int element;
// push elements into stack until it over-
flows
cout << "Enter Integer data to put into
the stack..." << endl;
do
{
cout << "Element to push?";
cin >> element;
while(stack.push(element)); //push and
check for overflow
//pop all elements from stack
cout << "The Stack Contains..." << endl;
while(stack.pop(element))
cout << "pop: " <<element << endl;
}
Run
Enter Integer data to put into the
stack...
```


Element to push ? 1
Element to Push ? 2
Element to Push ? 3
Element to Push ? 4
Element to Push ? 5
Element to Push ? 6

Stack overflow

The Stack contains...

Pop: 5

Pop: 4

Pop: 3

Pop: 2

Pop: 1

Stack Underflow

In the example (stack.cpp), the class Stack serves as a framework for building the derived classes and it is treated as a member of the derived class MyStack. The abstract class is the most important class and normally exists at the root of the hierarchy; it is a pathway to extending the sys-

tem. Hence, the class Stack is sometimes loosely called as abstract class or abstract base class, meaning that no actual instances (objects) of these classes are created. However, abstract classes, in addition to inheritance, have more significance in connection with virtual functions.

Abstract classes have other benefits. It provides a framework upon which other classes can be built and need not follow the trick of C (language, C++'s base class) programming. Most of the C programmers follow tricks of creating skeleton code and then copying and modifying the skeleton to create new functionality. One problem with skeleton code is if any modification is done to skeleton code, the changes must be propagated manually throughout the system - an error prone process at best. In addition, it is difficult to find out whether bugs are in original skeleton or in modified system versions. By using abstract classes, interface can be changed which immediately propagate changes throughout the system with no errors. All changes made by the programmer in the derived classes are shown explicitly in the code, any bugs that show up are almost isolated in the new code.

PREVIOUS YEARS QUESTIONS

PART-A

Prob.1 What is the purpose of using overloading operators? [R.T.U. 2018]

Sol. Purpose of using Operator Overloading : Operator overloading is one of the important features of C++ language that enhances the exhaustibility. C++ incorporates the option to use standard operators to perform operations with classes in addition to fundamental types. For example:

```
int a,b,c;
a=b+c;
```

This is obviously a valid code in C++, since the different variables of the addition are all fundamental types. Nevertheless, it is not so obvious that we could perform an operation similar to the following one:

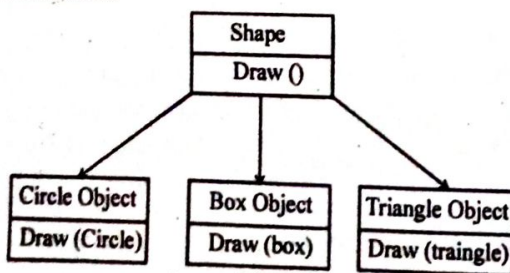
```
struct
{
    string product;
    float price;
}
a,b,c;
a=b+c;
```

Prob.2 Write short note on Dynamic Binding. [R.T.U. 2018, 2016]

OR

Explain the term dynamic binding. How it is useful in OOPs? [R.T.U. 2017]

Sol. Dynamic Binding : Binding refers to the linking of a procedure call to the code to be executed in response to the call. Dynamic binding (also known as late binding) means that the code associated with a given procedure call is not known until the time of the call at run-time. It is associated with polymorphism and inheritance. A function call associated with a polymorphic reference depends on the dynamic type of that reference.



Consider the procedure "draw" in Fig. By inheritance, every object will have this procedure. Its algorithm is, however, unique to each object and so the draw procedure will be redefined in each class that defines the object. At run-time, the code matching the object under current reference will be called.

Prob.3 Differentiate unary and binary operators. [R.T.U. Dec. 2016]

Sol. Difference between Unary and Binary Operator

	Unary operator	Binary operator
1.	A type of operator that works on one operand	A type of operator that works on two operands
2.	Following operators are unary operators : -, ++, --	Following operators are binary operators : +, -, *, %, /
3.	Example. ++i, i++, I--	Example. i/2, a + b
4.	It only requires operator and variable.	Binary operators typically require more than one variable

Prob.4 What is an operator function? Describe the syntax of an operator function. [R.T.U. Dec. 2016]

Sol. Operator Function

To define an additional task to an operator, we must specify what it means in relation to the class to which operator is applied. This is done with the function called operator function.

Syntax of operator function ;

```
return type classname::operator op(arg
list)
{
    function body    // task define
}
```

Prob.5 How to call virtual function with same name but different parameters. [R.T.U. 2016]

Sol.

```
#include <iostream>
class IBase
{
public:
    virtual void f(int) = 0;
    virtual void f(float) = 0;
};
class HelpBase : public IBase
{
public:
    using IBase :: f;
```



```

virtual void f(float arg) override
{
    std::cout << 1 << std::endl;
    f(static_cast<int>(arg));
    std::cout << 3 << std::endl;
}
};
class X : public HelpBase
{
public:
    using HelpBase::f;
    virtual void f(int) override
    {
        std::cout << 2 << std::endl;
    }
};
int main()
{
    X x;
    x.f(0.1f);
}

```

Prob.6 What is the basic difference in inheritance and polymorphism. [R.T.U. 2014]

Sol. Difference between Inheritance and Polymorphism

Inheritance	Polymorphism
Inheritance is one of the primary concepts of object-oriented programming. It allows us to reuse existing code. Through effective employment of reuse, we can save time in our programming.	There are two types of polymorphism: 1. Compile time polymorphism 2. Run time polymorphism.
Inheritance is transitive in nature.	Polymorphism is not transitive in nature
When one object inherits functionality implemented by another.	When functionality defined in some base object can vary in implementation depending on the actual object instance and what derived class it actually is.
This allows us to create new object that share base functionality with other object, while implementing their own new, additional functionality.	This allows us to create base object that define and even implement a basic contract (or "interface") but allow derived classes to change the way that contract is implemented.
Inheritance is the feature where functionality of one class is available for another class.	Polymorphism refers to the feature where the same entity exists as multiple items. Ex: method overriding, method overloading etc.

PART-B

Prob.7 Write short note on Abstract Class. [R.T.U. 2018]

Sol. Abstract Classes : An abstract class is, conceptually, a class that cannot be instantiated and is usually implemented as a class that has one or more pure virtual (abstract) functions.

A pure virtual function is one which must be overridden by any concrete (i.e., non-abstract) derived class. This is indicated in the declaration with the syntax "= 0" in the member function's declaration.

Characteristics of Abstract Class

1. Abstract class cannot be instantiated, but pointers and references of abstract class type can be created.
2. Abstract class can have normal functions and variables along with a pure virtual function.
3. Abstract classes are mainly used for upcasting, so that its derived classes can use its interface.
4. Classes inheriting an abstract class must implement all pure virtual functions, or else they will become abstract too.

Prob.8 Create a time class that includes integer member values for hours, minutes and seconds. Make a member function `get_time()` that gets a time value from the user, and a function `put_time()` that displays a time in 12:59:59 format. Add error checking to the `get_time()` function to minimize user mistakes. This function should request hours, minutes and seconds separately, and should check that the range should be in between 0 and 23, and minutes and seconds between 0 and 59. [R.T.U. 2018]

Sol.

```
#include <iostream>
```

```
#include <conio.h>
```

```
using namespace std;
```

```
class time{
```

```
private:
```

```
    int hours, minutes, seconds;
```

```
public:
```

```
    time(){
```

```
        hours = minutes = seconds = 0;
```

```
    }
```

```
    time(int h, int m, int s){
```

```
        hours = h;
```

```
        minutes = m;
```

```
        seconds = s;
```

```
    }
```

```
    void showTime() const {
```

```
        cout << hours << ":" << minutes << ":" << seconds;
```

```
    }
```

```
    void addTime(time x, time y){
```

```
        seconds = x.seconds + y.seconds;
```

```
        if(seconds > 59){
```


of the base class can be overridden by functions of the derived class. The keyword 'virtual' provides a mechanism for defining the virtual functions. When declaring the base class member functions, the keyword 'virtual' is used with those functions, which are to be bound dynamically. The syntax of defining a virtual function in a class is shown in example :

```
class MyClass
{
    public :
    -----
    -----
    virtual Return Type
    FunctionName (arguments)
        //virtual is keyword
    -----
    -----
};
```

Virtual functions should be defined in the public section of a class to realize its full potential benefits. When such a declaration is made, it allows to decide which function to be used at runtime, based on the type of object, pointed to by the base pointer, rather than the type of the pointer.

It is important to note that, virtual functions have to be accessed through a pointer to the base class. However, they can be accessed through objects instead of pointers. It is to be remembered that runtime polymorphism is achieved only when a virtual function is accessed through a pointer to the base class. Note that, when a function is defined as virtual in the base class and the same function is redefined in the derived class, then that function is virtual by default. Only class member functions can be declared as virtual functions. Regular functions and friend functions do not qualify as virtual functions.

Make a Virtual Function Pure : When the class in which it is declared is intended to be abstract (meaning it must be derived from, it cannot be instantiated by itself), and where the derived class is expected to provide an implementation for the method even if the base class has its own implementation. Derived classes that do not fully implement all the pure-virtual methods of their base class become abstract themselves. However derivatives inherit the sum of all pure-virtual implementations from their base classes, other than from those that initially declared the pure-virtual methods.

Prob.10 Overload + operator so that it can be used to concatenate two character strings. Write workable code.
[R.T.U. 2015]

OR

How to concatenate two strings by (+) operator give suitable example.
[R.T.U. 2017]

ol. Program to concatenate two strings by + operator which uses operator overloading:

```
#include<iostream.h>
#include<string.h>
const int BUFF_SIZE=50;
class string
{
    private:
    char str [BUFF_SIZE];
    public:
    string()
    {
        strcpy(str, " ");
    }
    string(char * Mystr)
    {
        strcpy(str, Mystr);
    }
    void echo()
    {
        cout<<str;
    }
    string operator + (string)
    {
        string temp = str;
        strcat ( temp.str, s.str );
        return temp;
    }
};

void main()
{
    string str1 = "welcome to";
    string str2 = "operator overloading";
    string str3 ;
    cout<< "\n str1 = ";
    str1.echo();
    cout<< "\n str2 = ";
    str2.echo();
    cout<< "\n str3 = ";
    str3.echo();
    str3= str2 + str1;
    cout<< "\n After str3 = str1 + str2";
}
```

Prob.11 (a) What are the restrictions and limitations for operator overloading?

(b) List the operators that cannot be overloaded. Define a complete class by name distance with feet and inches as data member and overload += operator and two objects.
[R.T.U. 2016]

Sol. Restrictions Applied to Operator Overloading Process : There are certain restrictions and limitations in overloading process. They are listed below:

1. Only existing operators can be overloaded. New operators cannot be created.
2. The overloading operator must have at least one operand that is of user-defined type.

Expression	Operator	Member function	Global function
@a	+ - * & ! ~ + +	A::operator@()	operator@(A)
a@	+ + - -	A::operator@ (int)	operator@ (A, int)
a@b	+ - * / % ^ & < > = = ! <= >= >= << >> & &	A::operator@ (B)	operator@ (A, B)
a@b	+ + - - * * / / % = ~ & & = << >> =	A::operator@ (B)	-
a(b, c, ...)	()	A::operator()(B, C, ...)	-
a → x	→	A::operator→ ()	-

Although the prototype of a function `operator+` can seem obvious since it takes what is at the right side of the operator as the parameter for the operator member function of the object at its left side, other operators may not be so obvious. Here we have a table with a summary on how the different operator functions have to be declared (replace @ by the operator in each case):

Where a is an object of class A, b is an object of class B, and c is an object of class C. You can see in this panel that there are two ways to overload some class operators: as a member function and as a global function. Its use is indistinct; nevertheless remind you that functions are not members of a class cannot access the private or protected members of that class unless the global function is its friend.

Prob. 20 Explain Run time polymorphism, its advantages and how it is implemented in C++. Also discuss abstract class and its purpose. [R.T.U. 2015]

Sol. Run-Time Polymorphism : Run-time polymorphism is a way for a programmer to take advantage of the benefits offered by polymorphism and late binding. Run-time polymorphism uses virtual functions to create a standard interface and to call the underlying functions. Those function definitions are bound to function calls during run time.

The term virtual function is one of those computer terms that is baffling the first few times you hear it used. Let's pick apart the term and review an example to clear up any confusion you might have. Virtual means that something appears to be real, but isn't real. For example, a flight simulator lets you fly a virtual airplane. The airplane isn't really there, but you have the feeling you are flying a real airplane.

In the case of a virtual function, the computer is tricked into thinking a function is defined, but the function doesn't have to be defined at that moment. Instead, the virtual function can be a placeholder for the real function. The real function is defined when the program is running.

Implementation of Run-time polymorphism in C++

Virtual functions are the C++ language's feature to achieve the run-time polymorphism. When we use the same function name in both the base and the derived classes, the

function in base class is declared as virtual using the keyword "virtual" prefixed with the function declaration. When a function is made virtual, the decision as to which function to be invoked is done at the run-time based on the type of the object pointed to by the pointer, rather than the type of the pointer. The keyword virtual is required only for the declaration but not for the definition. Once a function is declared as virtual in the base class, it is virtual in all the derived classes.

The syntax of a virtual function is as follows :

<return_type> virtual <function_name> (<parameter_list>)

Or

virtual <return_type> <function_name> (<parameter_list>)

where,

virtual → keyword

function_name → name of the function declared as virtual

return_type → data type returned by the function

parameter_list → formal arguments

Let us have a look at an example program.

Example of Run Time Polymorphism

```
#include <iostream.h>
using namespace std;
class A
{
public:
    void virtual display () // virtual function
    {
        cout << "Subhash loves kids" << endl;
    }
};
class B : public A
{
public:
    void display ()
    {
        cout << "Subhash loves Apple" << endl;
    }
};
int main ()
{
    A a;
    B b;
    A *ap;
    ap = & b;
    ap → display (); // calls derived class 'display ()'
}
function
return 0;
```

Output

Subhash loves Apple

In the above program, as can be observed from the

base class declaration, the function 'display ()' is declared as virtual. Now, in 'main ()' function the pointer of type 'base class A' points to the derived class object 'b'. Calling the 'display ()' function with reference to the pointer in the 'main ()' function, the 'display ()' function definition in the derived class gets invoked. Now, the problem, we faced earlier is solved. The 'display ()' function in the base class is said to be virtually hidden from the derived class. The decision to invoke the right function happens at run-time, and therefore, virtual functions help in achieving run-time polymorphism.

Advantages of Run Time Polymorphism

1. The advantage of run-time polymorphism is that a program can respond to events that occur during execution.
2. It allows you to create programs than can respond to events occurring while the program executes without having to create a large amount of "contingency code."
3. Helps in reusability of code.
4. Provides easier maintenance of applications.
5. Helps in achieving robustness in applications.
6. The main advantage of run time polymorphism is flexibility.

Abstract Classes : Refer to Prob. 7.

Example

```
class AbstractClass {
public:
    virtual void AbstractMemberFunction() = 0;
    // Pure virtual function makes
    // this class Abstract class.
    virtual void NonAbstractMemberFunction1();
    // Virtual function

    void NonAbstractMemberFunction2();
};
```

Purpose of Abstract Class

The purpose of an abstract class (often referred to as an ABC) is to provide an appropriate base class from which other classes can inherit. Abstract classes cannot be used to instantiate objects and serves only as an interface. Attempting to instantiate an object of an abstract class causes a compilation error.

An abstract class is a class that is designed to be specifically used as a base class. In general an abstract class is used to define an implementation and is intended to be inherited from by concrete classes. It's a way of forcing a contract between the class designer and the users of that class. If we wish to create a concrete class (a class that can be instantiated) from an abstract class we must declare and define a matching member function for each abstract member

function of the base class. Otherwise, if any member function of the base class is left undefined, we will create a new abstract class (this could be useful sometimes). Sometimes we use the phrase "pure abstract class," meaning a class that exclusively has pure virtual functions (and no data). The concept of interface is mapped to pure abstract classes in C++, as there is no construction "interface" in C++.

Prob.21 What is operator overloading? How will you overload binary unary operations? Discuss both processes with the help of programming implementation.

[R.T.U. 2014, 2012]

Sol. Operator Overloading : Refer to Prob.1.
Process of Overloading Stream Function

The best way to overload the stream operators is not to make them members of any class, but to keep them as friends. i.e., wherever there is a need to use the stream operators, use them as friend functions with the suitable parameters.

The following example shows the use of these operators for a class.

```
#include<iostream.h>
#include <string.h>
class Base
{
    char strVal[100];
public:
    Base () {strcpy(strVal, "");}
    Base (char *val){strcpy(strVal, val);}
    ~Base(){*strVal= '\0';}
    friend istream&operator>>(istream & is,
    Base & obj);
    friend
    ostream&operator<<(ostream&os, constBase&obj);
};
istream&operator>>(istream&is, Base &obj)
{
    is>>strVal;
    return is;
}
ostream&operator<<(ostream&os, const
Base&obj)
{
    os<<obj.strVal;
    reutrn os;
}
void main ()
{
    Base b;
    cin>>b;
    cout<<"Printing the value\n";
    cout<<b<<endl;
}
```

If there are derived classess which need to use the stream operators, one way is to define some more versions of the stream operators with the derived class parameters.

Restrictions Applied to Operator Overloading Process :
Refer to Prob.11(a).

Operator Function : Refer to Prob.4.
Overloading Unary Operators

The unary minus operator. A minus operator when used as a unary, takes just one operand. We know that this operator changes the sign of an operand when applied to a basic data item. We will see here how to overload this operator so that it can be applied to an object in much the same way as is applied to an int or float variable. The unary minus when applied to an object should change the sign of each of its data items.

Program: Shows how the unary minus operator is overloaded.

```
# include <iostream.h>
using namespace std;
class space
{
    int x;
    int y;
    int z;
public :
    voidgetdata (int a, int b, int c);
    void display (void);
    void operator - ( );
    // overload unary minus
};
void space::getdata(int a, int b, int c)
{
    x = a;
    y = b;
    z = c;
}
void space : : display(void)
{
    cout << x << " ";
    cout << y << " ";
    cout << z << "\n";
}
void space : : operator - ( )
{
    x = - x;
    y = - y;
    z = - z;
}
int main ( )
{
    space S;
    S.getdata (10, -20, 30);
    cout << "S : " ;
    S. dispaly();
    -S; // activates operator - ( ) function
    cout << "S : " ;
    S.display();
    return 0;
}
```

Output :

```
S : 10      -20   30
S : -10     20   -30
```


OOP-72

"The function () takes no arguments." Then, what does this operator function do? It changes the sign of the data member of the object S. Since this function is a member function of the same class, it can directly access the members of the objects which activated it.

S2 = - S1;

will not work because, the function operator () does not return any value. It can work if the function is modified to return an object.

Overloading Binary Operators Using Member Function : (Arithmetic Operations)

Like unary operators, some mechanism can be used to overload a binary operator. How to add two complex numbers using a friend function. A statement like

C = Sum (A, B); // functional notation.
is used. The functional notation can be replaced by a natural looking expression:

C = A+B; // arithmetic notation
/*by overloading the + operator using an operator +() function.

Overloading + Operator: Program*/

```
#include <iostream.h>
using namespace std;
class complex
{
    float x; // real part
    float y; // imaginary part
public:
    complex ()
    {
        // constructor 1
        complex (float real, float imag)
        // constructor 2
        {
            x = real; y = imag;
        }
        complex operator + (complex);
        void display (void);
    };
    complex operator + (complex C)
    {
        complex temp; // temporary
        temp.x = x + C.x; // these are
        temp.y = y + C.y; // float additions.
        return (temp);
    }
    void complex::display (void)
    {
        cout << x << "+j" << y << "\n";
    }
    {
        complex c1, c2, c3;
        //invokes constructor 1
        c1 = complex (2.5, 3.5);
        //invokes constructor 2
        c2 = complex (1.6, 2.7);
        c3 = c1 + c2;
```

B.Tech. (III Sem.) C.S. Solved Papers

```
cout << "c1 = "; c1.display ();
cout << "c2 = "; c2.display ();
cout << "c3 = "; c3.display ();
return 0;
```

Output:

```
c1 = 2.5 + j3.5
c2 = 1.6 + j2.7
c3 = 4.1 + j6.2
```

How the operator overloading is implemented.

```
complex complex::operator + (complex c)
{
    complex temp;
    temp.x = X + C.x;
    temp.y = Y + C.y;
    return (temp);
}
```

There are following features of this function :

1. It receives only one complex type argument explicitly.
2. It returns a complex type value.
3. It is a member function of complex.

The function is expected to add two complex values and return a complex value as the result but receive only one value as argument.

```
c3 = c1 + c2; // invokes operator + ( )
function
```

Prob.22 What is type conversion in C++? How it is achieved in user defined data types?

[R.T.U. 2014, 2013, 2010]

Sol. Type Conversion

We know that when constants and variables of different types are mixed in an expression, C language applies automatic type conversion to the operands as per certain rules. Similarly, an assignment operation also causes the automatic type conversion. The type of data to the right of an assignment operator is automatically converted to the type of the variable on the left. For Example:

```
int m;
float x = 3.14159;
m = x;
```

Converts x to an integer before its value is assigned to m. Thus, the fractional part is translated. The type conversions are automatic as long as the data types involved are built-in types. Consider the following statement that adds two objects and then assigns the result to a third object.

V3 = V1 + V2; // V1, V2 and V3 are class type objects.

When the objects are of the same class type, the operations of addition and assignment are carried out smoothly and the compiler does not make any complaints. We have seen, in the case of class objects, that the values of all the data members of the right-hand object are simply copied into the corresponding members of the object on the left-hand.

Object Oriented Programming

Since the user-defined data types are designed by us to suit our requirements, the compiler does not support automatic type conversions for such data types. We must therefore, design the conversion routines by ourselves, if such operations are required.

Three types of situations might arise in the data conversion between incompatible types :

1. Conversion from basic type to class type.
2. Conversion from class type to basic type.
3. Conversion from one class type to another class type.

1. Basic to class type: The conversion from basic type to class type is easy to accomplish. It may be recalled that the use of constructors was illustrated in a number of examples to initialize objects. For example, a constructor is used to build a vector object from an int type array. Similarly, we use another constructor to build a string type object from a char * type variable. These are all examples where constructors perform a defacto type conversion from the arguments type to the constructor's class type.

Consider the following constructor:

```
String::String (char * a)
{
    length = strlen(a);
    P = new char [length + 1];
    strcpy (P, a);
}
```

This constructor builds a string type object from a char* type variable a. The variable length and p are data members of the class string. Once this constructor has been defined in the string class, it can be used for conversion from char * type to string type. For example :

```
String S1, S2;
char * name 1 = "IBM PC";
char * name 2 = "Apple Computers";
S1 = string (name1);
S2 = name 2;
```

The statement S1 = string (name 1);

first converts name 1 from char * type to string type and then assigns the string type values to the object S1. The statement

S2 = name2;

also does the same job by invoking the constructor implicitly. Let us consider another example of converting an int type to a class type.

```
class time
{
    int hrs;
    int mins;
public:
    time (int) // constructor
    {
```


The following conversion statement can be used in a function

```
time T1; // object T1 created
int duration = 85;
T1 = duration; // int to class type
```

After this conversion, the *hours* member of T1 will contain value of 1 and *mins* member a value of 25, denoting 1 hour and 25 minutes.

2. Class to Basic Type : The constructor functions do not support this operation. C++ allows us to define an overloaded casting operator that could be used to convert a class type data to a basic type. The general form of an overloaded casting operator function, usually referred to as a conversion function is :

```
Operator typename ()
{
    ..... (function statement)
}
```

This function converts a class type to typename. For example, the **operator double ()** converts a class objects to type **double**, the **operator int()** converts a class type objects to type **int**, and so on. Consider the following conversion function :

```
Vector : : operator double ()
{
    double sum = 0;
    for (int i = 0; i < size; i++)
        sum = sum + V[i] * u[i];
    return sqrt(sum);
}
```

This function converts a vector to the corresponding scalar magnitude. Recall that the magnitude of a vector is given by the square root of the sum of the squares of its components. The operator **double()** can be used as follows:

```
double length = double (V1);
or
double length = V1;
```

where V1 is an object of type vector. Both the statements have exactly the same effect. When the compiler encounters a statement that requires the conversion of a class type to a basic type, it quietly calls the casting operator function to do the job. The casting operator function should satisfy the following conditions :

- It must be a class member.
- It must not specify a return type.
- It must not have any argument.

Since it is a member function, it is invoked by the object and, therefore, the values used for conversion inside the function belong to the object that invoked the function. This means that the function does not need an argument.

In the string example we can do the conversion from **string** to **char *** as follows :

```
String : : operator char * ()
{
    return(p);
}
```

3. One class to Another Class Type : We have just seen data conversion techniques from a basic to class type and a class to basic type. But there are situations where we would like to convert one class type data to another class type.

Example

```
obj X = obj Y; // objects of different types
obj X is an object of class X and obj Y is an object of class Y. The class Y type data is converted to the class X type data and the converted value is assigned to the obj X. Since the conversion takes place from class Y to class X, Y is known as the source class and X is known as the destination class.
```

Such conversions between objects of different classes can be carried out by either a constructor or a conversion function. The compiler treats them the same way. Then, how do we decide which form to use? It depends upon where we want the type - conversion function to be located in the source class or in the destination class.

We know that the casting operator function :

```
operator . typename ()
converts the class objects of which it is a member to typename. The typename may be a built - in type or a user defined (another class type).
```

It shows that the conversion from a class of any other type (or any other class) should make use of a casting operator in the source class. On the other hand, to perform the conversion from any other type/class to a class type, a constructor should be used in the destination class.

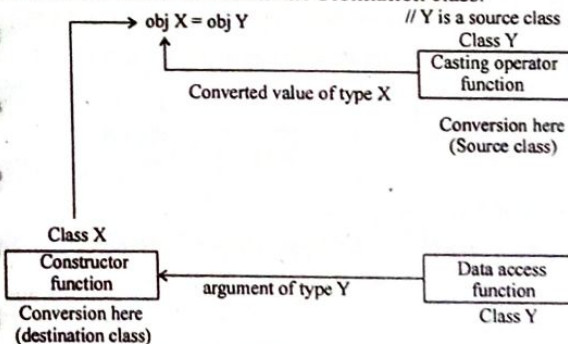


Fig. 1: Conversion between objects

Conversion Required	Conversion take place in	
	Source Class	Destination Class
Basic → Class	Not applicable	Constructor
Class → Basic	Casting Operator	Not applicable
Class Class	Casting Operator	Constructor

Fig. 2: Type conversions

When a conversion using a constructor is performed in the distribution class, we must be able to access the data member of the object sent (by the source class) as an argument. Since data members of the source class are private, we must use special access functions in the source class to facilitate its data flow to the destination class.

Prob.23 (a) Explain the process of overloading stream function with the help of an example. What are the restrictions applied to operator overloading process?

[R.T.U. 2012]

OR

Explain overloading stream function with suitable example.

[R.T.U. 2011]

(b) Explain the process of initializing member values with and without use of constructors.

[R.T.U. 2012]

Sol.(a) Process of Overloading Stream Function : Refer to Prob.21.

Sol.(b) A constructor is a special member function whose task is to initialize the objects of its class. It is special because its name is the same as the class name. The constructor is invoked whenever an object of its associated class is created.

We illustrate the use of constructors by using the following example program.

```
//class with constructors
#include <iostream.h>
class integer
{
    int m,n;
public :
    integer (int, int)
    //constructor declared
    void display (void)
{
    cout <<"m=" <<m<<"\n";
    cout <<"n=" <<n<<"\n";
};
integer : : integer (int x, int y)
//constructor defined
{
    m=x; n=y;
}
int main ()
{
    integer int1(0, 100)
    //constructor called implicitly
    integer int2= integer (22,75);
    cout <<"\nOBJECT 1" <<"\n";
    int1.display();
    cout <<"\nOBJECT 2" <<"\n";
    int2. display();
    getch();
}
```


PREVIOUS YEARS QUESTIONS

PART-A

Prob.1 Write a C++ program using function template to swap two values. [R.T.U. 2018]

Sol. Program for Swapping using Templates :

```
#include <iostream.h>
template<class FUNC>
void swap(FUN &a, FUNC &b)
{
    FUN t;
    t = a;
    a = b;
    b = t;
}
void main ()
{
    int a=10, b=20;
```

```
swap(a,b);
cout<<"a="<<a<<"\tb="<<b<<endl;
}
```

Prob.2 What is the output of the program

```
#include<iostream.h>
void main( )
{
    int n = 1;
    cout<<endl<<"The numbers are;"<
        do
        {
            cout<n<<"\t";
            n++;
        }
        while (n<=100);
    cout<<endl;
}
```


1. The numbers are :

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

Prob.3 What are the difference between error and exception?

Sol. Difference between Error and Exception :

Error is a compile time entity while exception is a runtime entity.

Error are those that do not allow program to run. Some error cannot be fixed at compilation time, they are runtime error. But the exceptions are those that occur because of the user during runtime which we are unable to avoid. For that we use exception handling to handle such kind of error. These allow the programs to execute and exit gracefully.

Prob.4 Write a short note on exception handling process.

Sol. Exception Handling Process

The exception handling process of C++ is handled through three keywords: *try*, *catch* and *throw*. The keyword *try* is used to contain the block of code which can generate an exception.

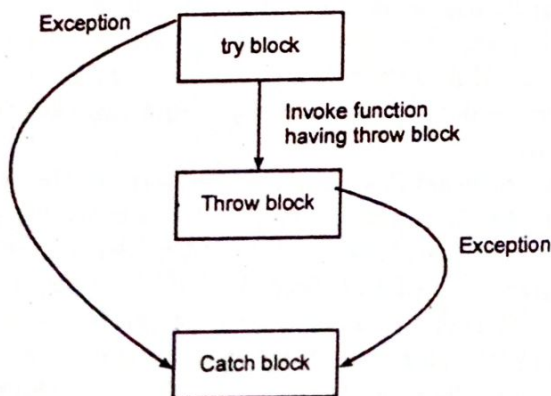


Fig. : Exception handling process

This block of code is also known as try block. We can have more than one try block in our program. When any exception is generated by try block then it is thrown using a

throw statement in the try block. The try block must be followed by a catch block. When any exception is thrown by try block using *throw* statement then the program control is automatically transferred to the appropriate catch block. The catch block is also known as exception handler. If program does not contain any catch block then program terminates abnormally.

PART-B

Prob.5 Write a template function that returns the average of all the elements of an array. The arguments to the function should be the array name and the size of the array (type int). In main (), exercise the function with arrays of type int, long, double, and char.

[R.T.U. 2018]

Sol. #include <iostream>

using namespace std;

```
template<typename T, size_t Size>
struct array {
    T val[Size];
};
```

```
template<typename T, size_t Size>
void print(array<T,Size>& a) {
    for (size_t i = 0; i < Size; i++) {
        cout<<a.val[i] << " ";
    }
    cout<<endl;
}
```

```
template<typename T, size_t Size>
void setElement(array<T,Size>& a, size_t index, T val) {
    if (index < Size) {
        a.val[index] = val;
    } else {
        // an exception may be good here ...
        cout<< "setElement : index out of range" <<endl;
    }
}
```

```
template<typename T, size_t Size>
float avg(array<T,Size>& a) {
    float total = 0.0;
    for (size_t i = 0; i < Size; i++) {
        total += a.val[i];
    }
}
```



```

}
return total / Size;
}

int main(int argc, char *argv[]) {
    constint ARRAY_SIZE = 10;
    array<int, ARRAY_SIZE>myIntArray;
    array<float, ARRAY_SIZE>myFltArray;

    // Initialize
    for (int i = 0; i < ARRAY_SIZE; i++) {
        setElement(myIntArray, i, i+1);
    }
    cout<< "Array contents: " << endl;
    print(myIntArray);
    cout<< "Average of elements = ";
    cout<< avg(myIntArray) << endl<< endl;

    // Initialize
    for (int i = 0; i < ARRAY_SIZE; i++) {
        setElement(myFltArray, i, (float)(i*1.1));
    }
    cout<< "Array contents: " << endl;
    print(myFltArray);
    cout<< "Average of elements = ";
    cout<< avg(myFltArray) << endl;

    return 0;
}

```

if 0

Program output:

Array contents:

1 2 3 4 5 6 7 8 9 10

Average of elements = 5.5

Array contents:

0 1.1 2.2 3.3 4.4 5.5 6.6 7.7 8.8 9.9

Average of elements = 4.95

#endif

Prob.6 Write short note on Templates. [R.T.U. 2017, Dec. 2016]

Sol. Templates : It is a new concept which enable us to define generic classes and functions and thus provides support for generic programming. Generic programming is an approach where generic types are used as parameters in algorithms so that they work for a variety of suitable data types and data structures.

A template can be used to create a family of classes or functions. For example, a class template for an array class would enable us to create arrays of various data types such as *int* array and *float* array. Similarly, we can define a template for a function, say *mul()*, that would help us to create various versions of *mul()* for multiplying *int*, *float* and *double* data type values.

A template can be considered as a kind of macro. When an object of a specific type is defined for actual use, the template definition for that class is substituted with the required data type. Since a template is defined with a parameter that would be replaced by a specified data type at the time of actual use of the class or function, the templates are sometimes called parameterized classes or functions.

Class templates:

Consider a vector class defined to template :

```

class vector
{
    int *V;
    int size ;
public:
    vector (int m) // create a null vector
    {
        V= new int [size = m];
        for (int i=0; i <size; i++)
            V[i]= 0 ;
    }
    vector (int * a)
        // create a vector from an array
    {
        for (int i=0; i <size; i++)
            V [i]= a [i];
    }
    int operator * (vector &y)
        // scalar product
    {
        int sum=0;
        for (int i=0; i <size; i++)
            sum += this -> V [i] * y. V [i];
        return sum;
    }
};

```

Now suppose we want to define a vector that can store an array of float values. We can do this by simply replacing the appropriate *int* declarations with *float* in the vector class. This means that we have to redefine the entire class all over again.

Assume that we want to define a vector class with the data type as a parameter and then use this classes to create a vector of any data type instead of defining a new class every time. The template mechanism enables us to achieve this goal.

This **general format** of a class template is:

```

template <class T>
class classname
{
    body of template;
};

```


The class template definition is very similar to an ordinary class definition except the prefix template <class T> and the use of type T. This prefix tells the compiler that we are going to declare a template and use T as a type name in the declaration. Thus, vector has become a parameterized class with the type T as its parameter. T may be substituted by any data type including the user-defined types.

Example:

```
Vector<int> V1(10); // 10 element int vector
```

```
Vector<float> V2(25); // 25 element float vector
```

A class created from a class template is called a template class. The syntax for predefining an object of template class is:

```
classname <type> objectname <arglist>;
```

This process of creating a specific class from a class template is called instantiation. The compiler will perform the error analysis only when an instantiation take place. It is, therefore, advisable to create and debug an ordinary class before converting it into a template.

Example of class template :

```
#include <iostream.h>
using namespace std;
const size=3;
template <class T>
class Vector
{
    T*V; // type T vector
public:
    vector ()
    {
        V= new T [size];
        for (int i=0; i<size; i++)
            V[i]=0;
    }
    vector (T*a)
    {
        for (int i=0; i <size ; i++)
            V[i]= a[i];
    }
    T operator * (vectort &y)
    {
        T sum=0;
        for (int i=0; i <size; i++)
            sum += this->V[i] * y.V[i];
        return sum;
    }
};

int main ()
{
    int x [3] = {1,2,3};
    int y [3] = {4,5,6};
    vector <int> V1;
    vector <int> V2;
    V1 = x;
```

```
V2 = y;
int R= V1 * V2;
cout << " R= "<<R<<"\n";
return 0 ;
```

Prob.7 Write short note on exception handling with example. [R.T.U. 2016, 2015]

OR

Write short note on exception handling. [R.T.U. 2017]

OR

What is the exception handling? Discuss the exception handling mechanism. [R.T.U. 2014]

Sol. Exceptions: Exceptions are peculiar problems that a program may encounter at runtime.

Concept of Exception Handling Process:

Exceptions are of two kinds, namely, *synchronous exceptions* and *asynchronous exceptions*. Errors such as "out-of-range index" and "over-flow" belong to the synchronous type exceptions. The errors that are caused by events beyond the control of the program (such as keyboard interrupts) are called asynchronous exceptions. The proposed exception handling mechanism in C++ is designed to handle only synchronous exceptions.

The purpose of the exception handling mechanism is to provide means to detect and report an "exceptional circumstance" so that appropriate action can be taken. The mechanism suggests a separate error handling code that performs the following tasks:

- Find the problem (Hit the exception)
- Inform that an error has occurred (Throw the exception).
- Receive the error information (Catch the exception).
- Take corrective actions (Handle the exception).

The error handling code basically consists of two segments, often to detect errors and to throw exceptions, and the other to catch the exceptions and to take appropriate actions.

Exception Handling Mechanism

C++ exception handling mechanism is basically built upon three keywords, namely try, throw, and catch. The keyword try is used to preface a block of statements (surrounded by braces) which may generate exceptions. This block of statements is known as *try block*. When an exception is detected it is thrown using a *throw* statement in the try block. A *catch block* defined by the keyword *catch* 'catches' the exception 'throw' by the throw statement in the try block, and handles it appropriately. The relationship is shown in fig.1.

The catch block that catches an exception must immediately follow the try-block that throws the exception. The general form of these two block are as follows:

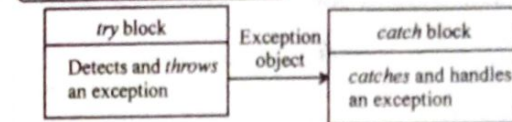
Object Oriented Programming

Fig.1: The block throwing exception

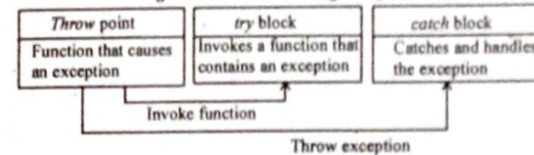


Fig.2: Function Invoked by try block throwing exception

```
try:
{
    .....
    throw exception; // Block of statements
    .....           // which detects and
                    // throws an exception
}
catch (type org) // Catches exception
{
    .....
    .....           // Block of statements
                    // that
    .....           // handles the exception
}
.....
.....
```

When the *try* block throws an exception, the program control leaves the *try* block and enters the *catch* statement of the catch block. Note that exceptions are objects used to transmit information about a problem. If the type of object thrown, matches the *arg* type in the catch statement, then catch block is executed for handling the exception. If they do not match, the program is aborted with the help of the *abort* () function which is invoked, by default. When no exception is detected and thrown, the control goes to the state immediately after the catch block. That is the catch block is skipped.

Most often, exceptions, are thrown by functions that are invoked from within the *try* blocks. The point at which the *throw* is executed is called the throw point. Once an exception is thrown to the catch block, control cannot return to the throw point. This kind of relationship is shown in Fig.2.

Prob.8 What are the difference between error and exception? Explain the keywords used in exception handling. [R.T.U. Dec. 2016]

Sol. Difference between Error and Exception : Refer to Prob.3.

Exception Handling Mechanism : Refer to Prob.7.

Prob.9 What is template class and template function? Use suitable example to explain them. [R.T.U. 2016]

Sol. Templates : Refer to Prob.6.

Prob.10 Write short note on exception handling with example. [R.T.U. 2016, 2015]

OR

What is the exception handling? Discuss the exception handling mechanism. [R.T.U. 2014]

Sol. Refer to Prob.7.

Prob.11 Explain the concept of stream classes.

Sol. C++ supports two complete I/O systems the one inherited from C and the object oriented I/O system defined by C++ (hereafter called simply the C++ I/O system). Like C-based I/O, C++'s I/O system is fully integrated. The different aspects of C++'s I/O system, such as console I/O and disk I/O, are actually just different perspectives on the same mechanism. Every program takes some data as input and generates processed data as output following the input-process output cycle.

C++ supports all of C's rich set of I/O functions that can be used in the C++ programs. But these are restrained from using due to two reasons, first I/O methods in C++ supports the concept of OOP and secondly I/O methods in c can not handle the user defined data types such as class objects C++ uses the concepts of streams and stream classes to implement its I/O operation with the console and disk files is linked to a physical device by the I/O system.

All streams behave in the same way even though the actual physical devices they are connected to may differ substantially. Because all streams behave the same, the same I/O functions can operate on virtually any type of physical device. For example, one can use the same function that writes to a file to write to the printer or to the screen. The advantages to this approach is that you need learn only one I/O system.

A stream act like a source or destination. The source stream that provide data to the program is called the input stream and the destination stream that receive output from the program is called the output stream C++ contains cin and cout predefined stream that opens automatically when a program begins its execution cin represents the input stream connected to the standard input device and cout represents the output stream.

The C++ I/O system contains a hierarchy of classes that are used to define various streams to deal with both the console and disk files. These classes are called stream classes.

OOP.82

Figure below shows the hierarchy of the stream classes for input and output operations with the console unit.

These class are declared in the header file iostream. The file should be included in all programs that communicate with the console unit.

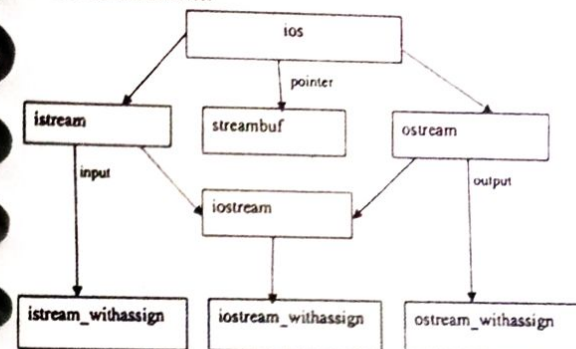


Fig. Stream classes for console I/O operations.

As in figure above ios is the base class for istream (input system) and ostream (output stream) which are base classes for iostream (input/output stream). The class ios is declared as the virtual base class so that only one copy of its members are inherited by the iostream.

Stream classes for console operations

Class name	Contents
ios (General input/output stream class)	Contains basic facilities that are used by all other input and output classes Also contains a pointer to buffer object (stream object) Declares constants and functions that are necessary for handling formatted input and output operations
istream (input stream)	Inherits the properties of ios Declares input functions such as get(), getline () and read () Contains overloaded extraction operator>>
ostream (output stream)	Inherits the property of ios Declares output functions put () and write () Contains overloaded insertion operator<<
iostream (input/output stream)	Inherits the properties of ios stream and ostream through multiple inheritance and thus contains all the input and output functions
streambuf	Provides an interface to physical devices through buffer Acts as a base for filebuf class used ios files

Prob.12 Explain file handling in object oriented programming**Sol. Data File Handling In C++**

File : The information/ data stored under a specific name on a storage device, is called a file.

Stream : It refers to a sequence of bytes.

Text file : It is a file that stores information in ASCII characters. In text files, each line of text is terminated with a special character known as EOL (End of Line) character or delimiter character. When this EOL character is read or written, certain internal translations take place.

Binary file : It is a file that contains information in the same format as it is held in memory. In binary files, no delimiters are used for a line and no translations occur here.

Classes for file stream operation

ofstream: Stream class to write on files

ifstream: Stream class to read from files

fstream: Stream class to both read and write from/to files.

Opening a file**Opening File Using Constructor**

```
ofstream outFile("sample.txt"); // output only
```

```
ifstream inFile("sample.txt"); //input only
```

Opening File Using Open ()

```
Stream-object.open("filename", mode)
```

```
ofstream outFile;
```

```
outFile.open("sample.txt");
```

```
ifstream inFile;
```

```
inFile.open("sample.txt");
```

File mode parameter	Meaning
ios::app	Append to end of file
ios::ate	go to end of file on opening
ios::binary	file open in binary mode
ios::in	Open file in reading only
ios::out	Open file for writing only
ios::nocreate	Open fails if the file does not exist
ios::noreplace	Open fails if the file already exist
ios::trunc	Delete the contents of the file if it exist

All of these flags can be combined using the bitwise operator OR (|). For example, if we want to open the file example.bin in binary mode to add data we could do it by the following call to member function open() :

```
fstream file;
```

```
file.open("example.bin", ios::out | ios::app | ios::binary);
```

Closing File

```
outFile.close();
```

```
inFile.close();
```

Object Oriented Programming**Input and Output Operation**

Put() and get () function

the function put () writes a single character to the associated stream. Similarly, the function get () reads a single character from the associated stream.

example :

```
file.get (ch);
```

```
file.put (ch);
```

write () and read () function

write () and read () functions write and read blocks of binary data.

example:

```
file.read((char*)&obj, size of(obj));
```

```
file.write((char*)&obj, size of(obj));
```

Error Handling Function

Function	Return Value and Meaning
eof ()	Returns true (non zero) if end of file is encountered while reading; other wise return false (zero)
fail()	return true when an input or output operation has failed
bad()	returns true if any invalid operation is attempted or any unrecoverable error has occurred.
Good ()	returns true if no error has occurred.

PART-C**Prob.13 Handle the following exceptions to input two integers from keyboard to perform division operation.**

(a) A try block to throw an exception when a wrong type of data is entered.

(b) When division by zero occurs.

Write appropriate catch block to handle above exceptions which are thrown from "thrown block". [R.T.U. 2018]

Sol. The code should throw an exception when the second integer is zero, ask the user to re-enter the second integer, and then complete the divide operation.

```
#include <iostream>
```

```
#include <stdexcept>
```

```
using std::cin;
```

```
using std::cout;
```

```
using std::endl;
```

```
int main() {
```

```
int i1, i2;
cout<< "Give two integers" << endl;
```

```
cin>> i1;
```

```
try {
```

```
first:
```

```
cin>> i2;
```

```
if (i2 == 0)
```

```
throw 0;
```

```
else
```

```
cout<< i1 / i2 << endl;
```

```
}
```

```
catch (...) {
```

```
cout<< "Second integer cannot be zero, please enter a valid integer" << endl;
```

```
goto first;
```

```
}
```

```
system("pause");
```

```
return 0;
```

```
}
```

Prob.14 Write a C++ program file handling to perform the following operations.

(a) A file named 'data.txt' contains number from 0 to 100. Open the 'data.txt' file in read mode.

(b) Read data from 'data.txt' file.

(c) If the number is odd, open a new file named 'odd.txt' and write the odd number into 'odd.txt' file.

(d) If the number is even; open a new file named 'even.txt' and write the even number into 'even.txt' file.

(e) Display the data of all the files. [R.T.U. 2018]

Sol.(a) Open a file for reading :

```
#include <iostream>
```

```
#include <fstream>
```

```
using namespace std;
```

```
int main() {
```

```
char data[100]; // buffer to store a line read from file
```

```
ifstream ifile; // declaring an object of class ifstream
```

```
ifile.open("data.txt"); // open "data.txt" for reading
```

```
cout<< "Reading data from a file :-" << endl << endl;
```

```
while (!ifile.eof()) { // while the end of file [ eof() ] is not reached
```

```
ifile.getline(data, 100); // read a line from file
```

```
cout<< data << endl; // print the file to console
```

```
}
```

```
ifile.close(); // close the file
```

```
return 0;
```

```
}
```


(b) Open the file for reading :

```
#include<iostream>
#include<fstream>
using namespace std;
```

```
int main() {
    char line[100];
    fstream file; // declare an object of fstream class
    file.open("data.txt", ios::out | ios::app); // open file in
    append mode
    if (file.fail()) { // check if file is opened successfully
        // file opening failed
        cout<<"Error Opening file ... "<<endl;
    }
    else {
        // proceed with further operations
        cout<<"Enter a line : ";
        cin.getline(line, 100);
        file<< line <<endl; // Append the line to the file
        cout<<"Line written into the file" <<endl;
    }
    return 0;
}
```

(c) and (d) Read a text file called data.txt and segregate the even and odd words into two different files names even.txt and odd.txt

```
1. #include <stdio.h>
2. #include <string.h>
3.
4. int main()
5. {
6.     FILE *fe;
7.     FILE *fo;
8.     FILE *fg;
9.     int i;
10.    int j, l;
11.    char ch;
12.    char tmp[100];
13.
14.    fe = fopen("oddfile.txt", "a");
15.    if (fe == NULL)
16.    {
17.        perror("ODDFILE");
18.    }
19.    fclose(fe);
20.
21.    fe = fopen("evenfile.txt", "a");
22.    if (fe == NULL)
23.    {
24.        perror("EVENFILE");
25.
```

```
26.    }
27.    fclose(fe);
28.
29.    fg = fopen("generalfile", "r");
30.    if (fg == NULL)
31.    {
32.        perror("GENERALFILE");
33.    }
34.
35.    while((ch = fgetc(fg)) != EOF)
36.    {
37.        if (ch != '\n' && ch != ' ' && ch != '\t')
38.        {
39.            tmp[i] = ch;
40.            i++;
41.        }
42.
43.        else
44.        {
45.            printf("%s", tmp);
46.            l = strlen(tmp);
47.            j = l % 2;
48.            if (j == 0)
49.            {
50.                fe = fopen("evenfile.txt", "ab");
51.                if (fe == NULL)
52.                {
53.                    perror("EVENFILE");
54.                }
55.
56.                fwrite(&tmp, sizeof(tmp), 1, fe);
57.                fclose(fe);
58.            }
59.
60.            else
61.            {
62.                fo = fopen("oddfile.txt", "ab");
63.                if (fo == NULL)
64.                {
65.                    perror("ODDFILE");
66.                }
67.
68.                fwrite(&tmp, sizeof(tmp), 1, fo);
69.                fclose(fo);
70.            }
71.
72.        }
73.    }
74.
75.
76.
77. }
```

(e) Display the data of the files

```
#include<iostream>
#include<string.h>
#include<fstream>
#include<stdlib.h>
using namespace std;

int main()
{
    ifstream ifile;
    char s[100], fname[100];

    cout<<"Enter file name to read :: ";
    cin>>fname;

    ifile.open(fname);

    if(!ifile)
    {
        cout<<"nError in opening file...!\n";
    }

    exit(0);
}

cout<<"n";

while(ifile.eof() == 0)
{
    ifile>>s;
    cout<<s<<" ";
}

cout<<"n";
ifile.close();

return 0;
}
```

Prob.15 Explain container class and proxy classes in detail.
[R.T.U. Dec. 2016]

Sol. Proxy Classes

A proxy class allows us to hide even the private data of a class from clients of the class. Providing clients of our class with a proxy class that knows only the public interface to our class enables the clients to use our class's services without giving the clients access to our class's implementation details.

Implementing a proxy class requires several steps. First, we create the class definition for the class that contains the proprietary implementation we would like to hide. Our example class, called Implementation, is shown in Ex. 1. The proxy class Interface is shown in Ex. 2.

Containers set and multiset can store a number of items and provide operations for manipulating them using the values as the keys.

Containers map and multimap are used to store pairs of items, one called the key and the other called the value. We can manipulate the values using the key associated with them. The values are sometimes called mapped values. The main difference between a map and a multimap is that a map allows only one key for a given value to be stored while multimap permits multimap keys.

Derived Containers

The STL provides three derived containers namely, stack, queue and priority_queue. These are also known as container adaptors.

Stack, queues and priority queues can be created from different sequence containers. The derived containers do not support iterators and therefore we cannot use them for data manipulation.

Prob.16 (a) What is generic programming? How is it implemented in C++? Write a program to implement swapping using templates.

(b) Explain try, throw and catch constructs used in C++ with their syntax. [R.T.U. 2013]

Sol. (a) Generic Programming: Generic programming is a style of computer programming in which algorithms are written in various terms. Types that are then instantiated when needed for specific types are provided as parameters.

Generic programming refers to features of certain statically typed programming languages that allow some code to effectively refine the static type system.

C++ support Generic Programming very well through its template system, but to full express the ideas of generic programming in C++ one must use a variety of template techniques.

1. Defining and Instantiating Function Templates

A function template allows us to design a function that can be used to process different types of objects. For instance, we can specify that an argument can be an object of different types. The types of the return value can also vary depending on template parameters. The following code snippet shows the definition of a function template:

```
template <Class Class X, Class Class Y>
void square2(Class X and X, Class Y and Y)
{
    X * = X;
    Y * = Y;
}
```

The square2() function can be used to process different types of objects, as long as those objects implement the overloading operator *. For example, we can call the following function to calculate the square of a int and a float, two objects of different types:

```
square2 (int a, int, float aFloat);
```

We instantiate a function template by generating a function from the template function and parameters. For example, for the function template square2(), consider this statement:

```
square2(int aInt, float aFloat);
```

The preceding statement instantiates the following function:

```
square2(int, float);
```

from template <class classX, class classY> void square2(class X and X, class Y and Y) and the template parameters classX and classY. The actual instantiation is done by the compiler based on the template we define.

2. Defining and Instantiating Class Templates

A class templates provides the foundation for designing and implementing classes with different member types. We define a class template in a way similar to the way we design and implement a normal, nontemplate class. The difference is that members of a class template can be parameterized. Here is an example of a template class definition:

```
template <class T>
Class MyClass
{
    public : // constructors and destructor
    MyClass(); //default constructor
    MyClass(T & newval);
            // copy constructor
    MyClass(); // destructor
    void SetVar (T & newval);
    void GetVar() const;
            // friend functions
    ostream & operator << (ostream & OS,
    constructor MyClass < T >and C);
    private:
    T mVar;
};
template <class T>MyClass <T> : : MyClass()
{
}
template<class T>MyClass<T>::MyClass(T &
newval);
mVar(newval)
{
}
template<class T>MyClass<T>::~MyClass()
{
}
tempalte<class T>void MyClass<T>::SetVar(T&
newval)
{
    mVar = newVal;
}
template <class T> void MyClass <T> : :
GetVal() const
{
    return mVar;
}
template <class T> ostream and MyClass
<T> : : Operator << (ostream & OS, const
Myclass (T & C)
{
}
```


OOP.88

```
OS << "member variable mVar = "<< C.mVar
<< "\n";
return OS;
}
```

Compare this to a normal class definition; the notable difference is that class members are now type parameterized. When a member is defined outside the class definition, we must add the following prefix to the function definition:

```
template <class T>
```

We must also parameterize Myclass as follows:

```
Myclass <T>
```

We instantiate a class template by generating a class definition from the template class and parameters. The following example shows the instantiation of two classes, Myclass <int> and Myclass <string> :

```
int main()
{
    Myclass <int> intobj;
    intobj.SetVar(5);
    cout<<"intobj.GetVar="<<intobj.GetVar()
    << "\n";
    cout << "intobj";
    Myclass<string>stobj("This is a string");
    cout<< "stobj.GetVar="<<stobj.GetVar()
    << "\n";
    cout << stobj;
    return 0;
}
```

Program for Swapping using Templates : Refer to Prob.1.

Sol.(b) Try : The try block contains the code in which exception condition can occur. To define try block we must use try keyword before starting the block. Then the entire body of try block is enclosed in braces. This block tells the compiler that if any exception is raised then program control should be transferred to respective exception handler. Exception can be raised in try block itself or in any function

called by try block. The syntax of try block is as follows:

```
try
{
    //code which can raise an exception
}
catch( exception_type)
{
    //statements to handle the exception
}
```

Throw : If try block generates an exception condition or raise an exception then it is thrown to exception handler through throw statement. A throw statement transfers the control of program from try block to a particular catch block which can handle that exception. The throw statement throws an exception object which is used to match the particular catch block. The syntax of throw statement is as follows:

```
throw exception_object;
```

Catch : The catch keyword is used to design catch block which is also called exception handler. Catch block must be followed by a try block. It handles the exception thrown by try block. A try block can have multiple catch blocks. Each handler will only evaluate an exception that matches. The syntax of catch block is shown below:

```
catch (exception_object)
{
    //statements to handle the exception
}
```

In some situations, we may not be able to handle all types of exceptions. So we may not be able to design independent catch blocks. To solve this problem we can design a universal catch block which can handle any type of exception thrown by try block. The following syntax is used to implement this catch block:

```
catch(...)
{
    //statements to handle all types of exceptions
}
```

□□□

- System design
- Software requirement analysis
- Preliminary software design
- Detailed software design
- Software integration and test
- System integration and test

Spiral Model

The Spiral model is an evolutionary software process

model that couples the iterative nature of prototyping with the controlled and systematic aspects of the linear sequential model. It provides the potential for rapid development of incremental versions of the software.

Software Engineering Approaches

To counter certain problems that usually arise in the waterfall approach to the software development process, the rapid throw away prototype approach was developed.

PREVIOUS YEARS QUESTIONS**PART-A**

Define verification and validation. [R.T.U. 2018]

Ans. Verification : Verification is a process of evaluating the intermediary work products of a software development lifecycle to check if we are in the right track of creating the final product.

Now the question here is : What are the intermediary products? Well, These can include the documents which are produced during the development phases like, requirements specification, design documents, data base table design, ER diagrams, test cases, traceability matrix etc. We sometimes tend to neglect the importance of reviewing these documents. We should understand that reviewing itself can find out many hidden anomalies when if found or fixed in the later phase of development cycle, can be very costly.

In other words we can also state that verification is a process to evaluate the mediator products of software to check whether the products satisfy the conditions imposed during the beginning of the phase.

Validation : Validation is process of evaluating the final product to check whether the software meets the business needs, in simple words the test execution which we do in our day to day life are actually the validation activity which includes smoke testing, functional testing, regression testing, systems testing etc.

2. Suggest a suitable life-cycle model for a software product which an organization has undertaken on behalf of a specific customer who is likely to change his requirements. justify your answer. [R.T.U. 2018]

Ans. Here are recommended criteria to select the correct life-cycle model:

Factors	Waterfall	V-Shape	Evolutionary Prototyping	Spiral	Iterative and Incremental	Agile
Under User Requirement	Poor	Poor	Good	Excellent	Good	Excellent
Unfamiliar Technology	Poor	Poor	Excellent	Excellent	Good	Poor
Complex System	Good	Good	Excellent	Excellent	Good	Poor
Reliable system	Good	Good	Poor	Excellent	Good	Good
Short Time Schedule	Poor	Poor	Good	Poor	Excellent	Excellent
Strong Project Management	Excellent	Excellent	Excellent	Excellent	Excellent	Excellent
Cost limitation	Poor	Poor	Poor	Poor	Excellent	Excellent
Visibility of Stakeholders	Good	Good	Excellent	Excellent	Good	Excellent
Skills limitation	Good	Good	Poor	Poor	Good	Poor
Documentation	Excellent	Excellent	Good	Good	Excellent	Poor
Component reusability	Excellent	Excellent	Poor	Poor	Excellent	Poor

For solve the given problem we can use spiral model because spiral mode give best solution for customer change requirement.

Q.3 Write advantages and disadvantages of waterfall model.

Ans. Waterfall Model :

Merits:

- Clear project objectives
- Stable project requirements
- Progress of system is measurable
- Strict sign-off requirements.

Demerits:

- Time consuming
- Never backward
- Little room for iteration
- Difficulty responding to charges.

Q.4 What are the differences between waterfall model and prototype model?

Ans. Differentiate between prototype model and waterfall model:

	Waterfall Model	Prototype Model
1.	Client can only preview the system only after final version of the software is developed because there is no feedback loop.	Client have a preview of the system from the 'quick design' and the prototype developed early at the process.
2.	Developers encounter a freezing requirement where they are not allow to modify the requirements or specification of the previous phase until the next iteration.	Developers can refine or add requirements and specification to the system after the prototype is built.
3.	The complexity of an error increases because of the nature of the model, each phase is sequential of the other.	The complexity of an error is low because the prototype enables the developer to detect any deficiency early at the process.
4.	Waterfall model is implemented in the projects where conventional software has to be delivered to the client sin such cases user is sure about his requirements.	Prototype model is usually used in online projects where client is not sure about his requirements.

sequence one after another i.e. they are organized in a linear fashion of consecutive phases. That's why it is also called as the Linear Sequential Model. It is the oldest model and a Classic Life Cycle, most widely used as the basis of the other models. Now, it has become a little outdated.

The waterfall model is an engineering model, designed to be applied to the development of the software products. The software development starts from the system level and develops step by step through the various phases. The main criteria behind it is that such steps are to be followed consecutively. Each step is itself made up of more detailed activities and produces an output which becomes the input of the next step and so on.

The sequence of major phases performed in a software project is :

Concept Formation, Developing the Software Specification, Software Design, Coding and Unit Testing, Software Integration and Testing and Release, Implementation and Maintenance. These various steps of model are shown in fig.2.

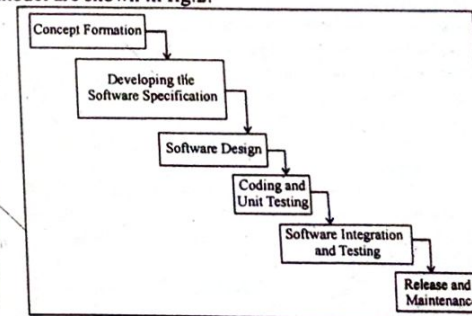


Fig. 2 : Major Phases of Waterfall Model

1. Concept Formation

Concept formation phase is divided into two phases which are as follows:

(i) Problem Definition : Many people consider this phase to be the most important phase. It provides a broad statement of user requirements or what the user expect from the software to do and thus sets the direction for the whole project. Additionally, this phase also sets the project scope, which defines what parts of the software can be changed and what parts should remain same. The resources to be made available to the project are often specified in this phase. Three important factors – The Project Goal, Project Scope and Resource Limits, generate the project terms of reference and are set by the organization's management.

(ii) Feasibility Study : Feasibility study proposes one or more conceptual solutions to the problem set for the project. Each conceptual solution gives an idea of what the new system will look like. These solutions also indicate what input will be needed by the software and what will be the output produced.

PART-B

Q.5 Explain the waterfall model. Suggest reasons, why it is not a true reflection of the activities which are involved in software development. Boehm came up with an alternative. Briefly explain it. [R.T.U. 2018]

Ans. Waterfall Model : Waterfall model is the simplest process model which has dominated software development for a number of years. It is a development process that centers around planned work and is best suited for projects where requirements can be clearly defined. It has been order of software development activities in the development of large software projects. It groups development activities into independent steps that are carried out in a systematic

SE.5

- (ii) Product testing includes the testing of product.
- (iii) Product development develops the software according to the mentioned specification.
- (iv) Product concept does the feasibility study and produces the product specification.

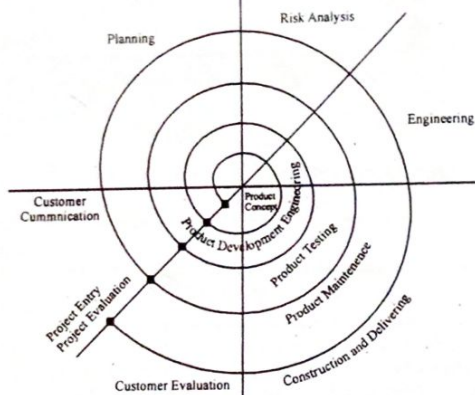


Fig. : Spiral Model

Additionally, a spiral model is partitioned into six task region.

- (1) Risk analysis
- (2) Engineering
- (3) Customer evaluation
- (4) Customer communication
- (5) Planning
- (6) Construction and delivery

Q.6 Explain the components of Software Requirement Specification (SRS). Write the IEEE recommended structure of SRS. [R.T.U. 2018]

Ans. Software Requirement Specification (SRS) document forms the basis for agreement between customers and the suppliers on what the software product should do.

There is not standard way of writing requirements specification document, but the following steps should be followed.

1. Creating SRS Outline : An SRS outline contains the following details :

- (a) Purpose of software
- (b) Scope of software
- (c) System overview
- (d) References
- (e) Definitions
- (f) Use cases
- (g) Functional Requirements
- (h) Non-Functional Requirements.

SE.6

Choosing SRS Writer : The SRS writer is a person or group of persons who understand the specifications of the software to present clear and unambiguous requirement from customer.

Provide only Relevant Information : The document should be proper and must be represented legibly in graphical tabular form.

4. Keep updating the SRS : As the task progresses, SRS updated in order to signify the changes that have been made.

The users related to SRS are :

- (1) Client
- (2) SRS writer
- (3) Designer / Developer
- (4) Tester.

The table listing the users and use of SRS with respect to that user has been shown in table.

S. No.	User	Use of SRS with respect to user
1.	Client	Specify the requirements to be met by the software.
2.	SRS Writer	Documents the requirements specified by the user in clear and unambiguous manner in form of SRS
3.	Developer / Designer	Uses SRS to translate the requirements in form of software by designing implementing it.
4.	Tester	Tests the software progress with SRS and mark revisions as per need.

The different users has been shown in order of occurrence:

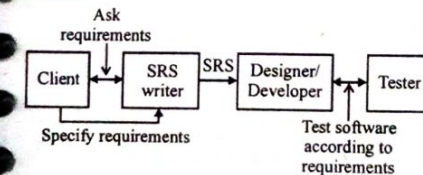


Fig.

SRS Documentation, following IEEE Std. 830:

General

- (a) All documents should have a title page (to include information such as: title of the project, course name and number, team members, place, date, and other relevant information).
- (b) Table of contents normally makes a lot of sense, so should be included as well.

B.Tech. (III Sem.) C.S. Solved Papers

(c) Number all sections in the document.

(d) Any reference correctly included in section (d) should be written as follows:

For books, reports, theses, standards, manuals, etc.:

[number] NameofAuthor(s), TitleofWork, Publisher, Place, Date

(if authors' or editors' names are not available, it can start with a title of the name of the Publisher)

For papers/articles:

[number] NameofAuthor(s), TitleofWork, JournalName, VolumeNumber, IssueNumber, Page Numbers (pp. first-last), Year (or Month and Year)

For papers in Conference Proceedings:

[number] NameofAuthor(s), TitleofWork, Phrase "Proceedings of the" Conference Name, Place, Date, Publisher, Place Date]

For websites:

[number] AuthorNames(s), TitleofWork, Company's Name, Place, Date, URL

Note : For names of authors never use full first names, only initials!

(e) All references from section (d) have to be referred to in the text (using [number] notation).

(f) Do not end section titles with colons.

(g) Every figure/diagram should have a caption (number and title). Place it underneath the figure/diagram.

(h) Every table should have a number and title, placed above the table.

(i) "Shall/Must" phraseology should not be used in unless it is requirement. This means that normally it is not used in sections 1 or 2.

2. Writing "Introduction"

• In Section (a) "Purpose", describe the purpose of this document, not the purpose of the software being developed.

• In Section (b) "Scope", describe the scope of this document, not the scope of the software being developed.

• In Section (e) "Overview", provide an overview of this document, not the overview of the software being developed.

• Definitions, in Section (d), should be write using the following template:

word_defined.<in lower case, ended with a dot ">"

Followed by a definition.

For example:

user. The person, or persons, who operate or interact directly with the product.

3. Writing "Overall Description"

• A context diagram is mandatory.

• Other important characteristics are: product perspective, product functions, user characteristics, constraints, assumptions and dependencies.

Software Engineering

4. Writing "Specific Requirements"

• Never specify the operating system or language in the SRS, unless the customer demands doing so. These are strictly implementation issues and well designed software can be implemented in any specific programming language to run under any specific operating system on any specific hardware platform.

• Specific requirements section should be split into:

- (i) "External Interfaces" derived from the context diagram
- (ii) "Functional Requirements" that should be further split into "Input Requirements" (related to user inputs, commands, etc.), "Output Requirements" (mostly related to the GUI), "Input/Output Requirements" (if they cannot be separated), and "Processing Requirements"
- (iii) "Non-Functional Requirements", such as performance, reliability, safety, security, etc.
- (iv) "Design Constraints", normally related to software and hardware limitations (OS, platform, stand-alone or networked, network protocols, standards, etc.);
- (v) "Database Requirements" – can be combined with "Design Constraints".

• Use case diagrams have to be included in most sections, specifically in the "Functional Requirements" section. Several use case diagrams have to be presented, including specific scenarios, how the system will respond to certain user/operator requests or commands, or network behavior.

5. Other

- End your SRS document by the following line (centered across the page):

*** End of the SRS ***

Q.7 Explain spe ialized process model. [R.T.U. 2017]

Ans. Specialized Process Models : Special process models take on many of the characteristics of one or more of the conventional models. However, specialized models tend to be applied when a narrowly defined software engineering approach is chosen.

1. Component-Based Development : Commercial off-the-shelf (COST) software components, developed by vendors who offer them as products, can be used when software is to be built. These components provide targeted functionality with well-defined interfaces that enable the component to be integrated into the software. The component-based development model incorporates many of the characteristics of the spiral model. It is evolutionary in nature, demanding an iterative approach to the creation of software. However, the model composes applications from prepackaged software components. Modeling and construction activities begin with

3. Aspect-Oriented Software Development : Regardless of the software process that is chosen, the builders of complex software invariably implement a set of localized features, functions and information content. These localized software characteristics are modeled as components and then constructed within the context of a system architecture. As modern computer-based systems become more sophisticated and complex, certain concerns, customer required properties or areas of technical interest, span the entire architecture. Some concerns are high-level properties of a system, others affect functions or are systemic.

When concerns cut across multiple system functions, features, and information they are often referred to as crosscutting concerns. Aspectual requirements define those crosscutting concerns that have impact across the software architecture. Aspects are mechanisms beyond subroutines and inheritance for localizing the expression of a crosscutting concern. Aspect-oriented software development (AOSD), often referred to as aspect-oriented programming (AOP), is a relatively new software engineering paradigm that provides a process and methodological approach for defining, specifying, designing, and constructing aspects.

A distinct aspect-oriented process has not yet matured. However, it is likely that such a process will adopt characteristics of both the spiral and concurrent process models. The evolutionary nature of the spiral is appropriate as aspects are identified and then constructed. The parallel nature of concurrent development is essential because aspects are engineered independently of localized software components and yet, aspects have a direct impact on these components.

Q.8 Discuss the merits & demerits of various model of software engineering models? [R.T.U. 2016]

OR

Discuss merits and demerits of various models of software development. [R.T.U. 2017]

Ans. Merits and demerits of various model of software engineering model.

(i) Waterfall model : Refer to Q.3.

(ii) Spiral model

Merits:

- Avoidance of Risk is enhanced.
- Strong approval and documentation control.
- Implementation has priority over functionality.
- Additional functionality can be added at a later date.

Demerits:

- Highly customized limiting re-usability.
- Applied differently for each application.
- Risk of not meeting budget or schedule.
- Possibility to end up implemented as the waterfall framework.

(iii) Prototype Model

Merits:

- Strong dialogue between user and developer.
- Confusing or difficult functions can be identified
- Missing functionality can be identified easily
- May generate specifications for a production application
- Encourages innovation and flexible designs.

Demerits:

- Identifying non-functional elements difficult to document
- Incomplete application may cause application not to be used as the full system was designed
- Incomplete or inadequate problem analysis
- Client may be unknowledgeable
- Requirements may frequently change significantly.

Q.9 Explain Spiral model in detail. Also differentiate between prototype model and water fall model. [R.T.U. 2016]

Ans. Spiral model : Refer to Q.5.

Differentiate between prototype model and waterfall model : Refer to Q.4.

Q.10 Explain the prototype model? Under what circumstances is it beneficial to construct a prototype model? [RTU 2016, 2013]

OR

Which of the software development process model you would suggest for a new railway reservation system that will generate useful stabilities and solution of the problems of the railway administration. Give justification also. [RTU 2010]

Ans. Prototype Model : Prototype model is suitable for new reservation system that will generate useful stabilities and solution of the problem of the railway administration.

Often designers are faced with the problem of specifying operations that can not be specified precisely. This often occurs in the organizations which are just starting to use computers or in novel applications where there is no precise specifications. Instead, it is more appropriate to develop the system gradually and learning about system capabilities as one goes along.

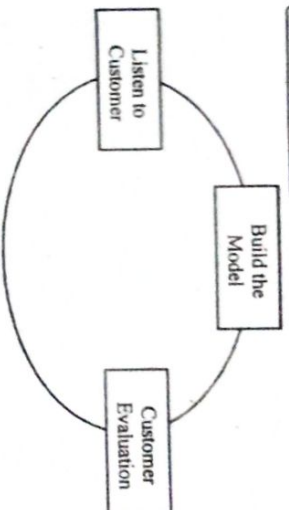


Fig. : Prototype Model

This model is advantageous to remove the limitations of the waterfall model. The basic principle of this model is that a throwaway prototype is built to help in understanding the requirement during the requirement analysis phase only. It presents one solution to the problem to be sure that the customer will get what he wants. It can be considered as the "Initial step" of the software.

(i) **Listen to Customer :** This model starts with the collection of needs and requirements of the customer. To fulfill such objective, initially a meeting is conducted between the customer and the developer. During this meeting they :

- Discuss the problem to be solved.
 - Set a particular, specific and static goal of the software.
 - Mention all the requirement which are known at this time.
- Such communication results to an agreement. In beginning, the customer presents only those requirements which are known by him at this time only i.e. he provides some partial requirements.

Phase Output : The output of this phase includes :

- an agreement document.
 - a list of all the requirements
- (2) **Build the Model :** In this phase, the developer gets ready to develop such model. He starts to build the model according to the given requirements of the customer and a "Quick Design" takes place.

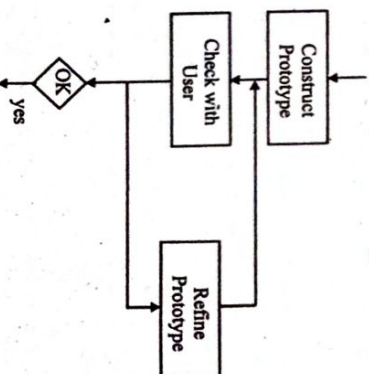


Fig. 2 : Built Model

Phase Output : This phase results that a raw and rough model is prepared which satisfies all the requirements mentioned by the customer and now it is ready to show to the customer as the intermediate work product.

(3) **Customer Evaluation :** In this, model is evaluated and tested by the customer. It is does not suits to the customer and he wants to have some more additions in it then he may give the feedback to the developer. It is quite possible that the customers wants to change all requirements, then that build model will be thrown away and again a new communication will be held between the customer and the developer. The customer can present his new requirement list and a new model is build according to this list. In this way, the repetitively loop will go and go until the customer tells all its requirements and at last get satisfied.

Phase Output : In the end of this phase, either the customer will give its feedback to the developer or user satisfaction and acceptance will be achieved.

Prototyping is the practice of building an early version of a software. This model is built at the early stages of the software development. It is built very rapidly and fast, so that it can be checked time-to-time by the customer. That is why, it is also as the "Rapid Prototyping". In such cases, if the model is to be built very quick, some reusable components can also be used to make it's task easy.

Advantages

- This misunderstandings between software developer and users may be identified.
- Missing facilities and features may be find out.
- Any non-confirm hard or confusing facilities or characteristics may be identified.
- Software developers may final the incomplete or inconsistent requirements as the prototype is developed.

Disadvantages

- The customer will want the early version of the model which may not have good quality contents and maintainability.
- As the model is to be built very quickly, during development if the developer use any inappropriate operating system or programming language which are available at this time, but not suitable for model. Later on, when almost whole model is built up, the developer forgets to remove that choices then the model which has been built doesn't satisfy the customer requirement fully.

Q.11 Explain the spiral model with advantages and disadvantages. [RTU 2015]

Ans. Spiral Model : Refer to Q.5.

Advantages of Spiral Model

- (1) Spiral Life Cycle model is one of the most flexible SDLC models in place. Development phases can be determined by the project manager, according to the complexity of the project.
- (2) Project monitoring is very easy and effective. Each phase, as well as each loop, requires a review from concerned people. This makes the model more transparent.
- (3) Risk management is one of the in-built features of the model, which makes it extra attractive compared to other models.
- (4) Changes can be introduced later in the life cycle as well. And coping with these changes isn't a very big headache for the project manager.
- (5) Project estimates in terms of schedule, cost etc become more and more realistic as the project moves forward and loops in spiral get completed.
- (6) It is suitable for high risk projects, where business needs may be unstable.
- (7) A highly customized product can be developed using this model.

Disadvantages of Spiral Model

- (1) Cost involved in this model is usually high.
- (2) It is a complicated approach especially for projects with a clear SRS.
- (3) Skills required, to evaluate and review project from time to time, need expertise.
- (4) Rules and protocols should be followed properly to effectively implement this model. Doing so, throughout the span of project is tough.
- (5) Due to various customizations allowed from the client, using the same prototype in other projects, in future, is difficult.
- (6) It is not suitable for low risk projects.
- (7) Meeting budgetary and scheduling requirements is tough if this development process is followed.
- (8) Amount of documentation required in intermediate stages makes management of project very complex affair.

Q.12 Describe how to prepare a Software Requirement Specification (SRS) document. List possible Users and Use of SRS for each User. [RTU 2014]

Ans. Refer to Q.6.

Q.13 Explain the term software engineering. Why it is important. Discuss in detail.

Ans. Software Engineering : Software engineering is an engineering discipline that is concerned with all aspects of

software production from the early stages of system specification through to maintaining the system after it has gone into use. In this definition, there are two key phrases:

1. Engineering discipline engineers make things work. They apply theories, methods and tools where these are appropriate. However, they use them selectively and always try to discover solutions to problems even when there are no applicable theories and methods. Engineers also recognize that they must work to organizational and financial constraints so they look for solutions within these constraints.
2. All aspects of software production software engineering is not just concerned with the technical processes of software development. It also includes activities such as software project management and the development of tools, methods and theories to support software production.

Engineering is about getting results of the required quality within the schedule and budget. This often involves making compromises-engineers cannot be perfectionists. People writing programs for themselves, however, can spend as much time as they wish on the program development.

In general, software engineers adopt a systematic and organized approach to their work, as this is often the most effective way to produce high quality software. However, engineering is all about selecting the most appropriate method for a set of circumstances so a more creative, less formal approach to development may be effective in some circumstances less formal development is particularly appropriate for the development of web-based systems, which required a blend of software and graphical design skills.

Software engineering is important for two reasons:

1. More and more, individuals and society rely on advanced software systems. We need to be able to produce reliable and trustworthy systems economically and quickly.
2. It is usually cheaper, in the long run, to use software engineering methods and techniques for software systems rather than just write the programs as if it was a personal programming project. For most types of systems, the majority of costs are the costs of changing the software after it has gone into use.

The systematic approach that is used in software engineering is sometimes called a software process. A software process is a sequence of activities that leads to the production of a software product. There are four fundamental activities that are common to all software processes. These activities are:

Software Engineering

1. Software specifications, where customers and engineers define the software that is to be produced and the constraints on its operation.
2. Software development, where the software is designed and programmed.
3. Software validation, where the software is checked to ensure that it is what the customer requires.
4. Software evolution, where the software is modified to reflect changing customer and market requirements.

Q.14 Write advantages & limitation of formal requirement specification.

Ans. Merits of formal requirements specification

Formal methods possess several positive features, some of which are discussed below:

- Formal specifications encourage rigour. Often, the very process of construction of a rigorous specification is more important than the formal specification itself. The construction of a rigorous specification clarifies several aspects of system behavior that are not obvious in an informal specification.

- Formal methods usually have a well-founded mathematical basis. Thus, formal specifications are not only more precise, but also mathematically sound and can be used to reason about the properties of a specification and to rigorously prove that an implementation satisfies its specifications.

- Formal methods have well-defined semantics. Therefore, ambiguity in specifications is automatically avoided when one formally specifies a system.

- The mathematical basis of the formal methods facilitates automating the analysis of specifications. For example, a tableau-based technique has been used to automatically check the consistency of specifications. Also, automatic theorem proving techniques can be used to verify that an implementation satisfies its specifications. The possibility of automatic verification is one of the most important advantages of formal methods.

- Formal specifications can be executed to obtain immediate feedback on the features of the specified system. This concept of executable specifications is related to rapid prototyping. Informally, a prototype is a "toy" working model of a system that can provide immediate feedback on the behavior of the specified system, and is especially useful in checking the completeness of specifications.

Limitations of formal requirements specification

It is clear that formal methods provide mathematically sound frameworks within large, complex systems can be specified, developed and verified in a systematic rather than in an ad hoc manner. However, formal methods suffer from several shortcomings, some of which are the following:

- Formal methods are difficult to learn and use.
- The basic incompleteness results of first order logic suggest that it is impossible to check absolute correctness of systems using theorem proving techniques.

- Formal techniques are not able to handle complex problems. This shortcoming results from the fact that, even moderately complicated problems blow up the complexity of formal specification and their analysis. Also, a large unstructured set of mathematical formulas is difficult to comprehend.

PART-C

Q.15(a) Explain RAD model in detail. Draw suitable diagram to show its activities.

(b) Explain why programs that are developed using evolutionary development are likely to be difficult to maintain. [R.T.U. 2018]

- **Ans.(a) RAD Model :** RAD, *Rapid Application Development*, a methodology that has the same objective of speedy response to user needs as does prototyping but is broader in scope. The RAD term was coined by computer consultant and author 'James Martin'. RAD refers to a development life cycle intended to produce systems (or softwares) quickly without sacrificing quality.

RAD model is an incremental model combining waterfall model with component based development model, which

- Consumes very less time to develop the software.
- Is the high speed, very fast processing model.
- Is the modified version of the waterfall model.
- Is used in the business oriented applications.
- Can use reusable components.
- Performs various business functions for which the different and independent teams are made for each function.

There are various independent teams to complete the workdone of various business functions. Each individual business function has a separate RAD team. When all the

functions are developed, they are integrated to form it as a whole. This model is divided into five phases, shown in fig. All the phases are explained below:

- 1. Business Modeling :** In this phase, the information flow among the various functions is described. This information will be generated and processed by software engineers. It is to be checked from where that information is created and where it will go.
- 2. Data Modeling :** To refine the information flow of the previous phase, it is partitioned into three different data objects. Each data object may have the various attributes and then the relationships and connections are made among these data objects.
- 3. Process Modeling :** The data objects of data modeling phase are now combined to provide the information flow of each business function. Some documents and reports are prepared for addition, updation, editing like, deletion, creation and retrieval of the data objects.

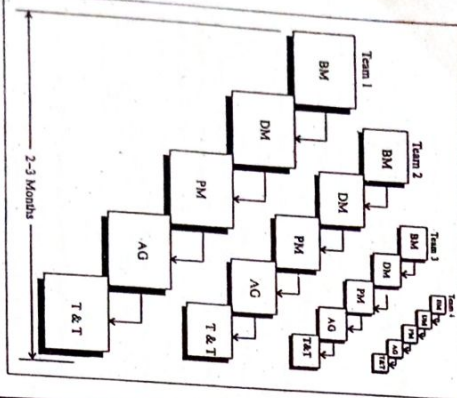


Fig. : Rapid Application Development Model
BM: Business Modeling
DM: Data Modeling
PM: Process Modeling
AG: Application Generation
T & T: Testing and Turnover
4. Application Generation : This model uses the concept of reusability. So the product is built using the new

built components as well as some already existing components. If any new-built component of this business application is found to be important then it can be stored in the library or data store for the later projects and future use. Now this component will act as reusable component in any other new project.

- 5. Testing and Turnover :** The unit testing is done, where all the new built components are tested. The existing ones need not to be tested because they have already been tested in their previous projects. So, this reduces the overall testing time. After that, all components are merged together to form an integrated one. Now, this integrated system will be tested against the mentioned requirements for their fulfillment and satisfaction. In this way, the whole business application is developed and it is now delivered to the customer. RAD model is built in the time constraints i.e. of 2 to 3 months. So, if a business application is divided into its various functions, each of the function must be completed within three months times. If it is not completed within this time limitation then that application is not RAD model based.

Advantages

1. Very less time consuming.
2. Good for the business-oriented applications.
3. Independent teams for separate functions.

Disadvantages

1. Human Resources : RAD requires the sufficient human resources to create the RAD teams.
2. Time Constraints : The RAD process must be completed within the time limits (60-90 days) otherwise this model is useless for the particular business requirement.
3. Performance : This model does not suit to all types of applications. Because this model is bounded in the time constraints, and it may or may not achieve good performance.

4. Risk Handling : RAD is not appropriate when the technical risks are high. These risks occur due to more use of new technology or when any new software requires high interoperability with the existing programs. So, when the technical risks become reality, implementation may become impossible.

Ans.(b) Evolutionary Development Model in Software Engineering

This evolutionary development model based on the initial implementation will result in the user comments, so it can be repaired through many versions until an adequate system can be developed. In addition to having separate activities, this model provides feedback to capabilities simultaneously. There are 2 types in the evolutionary development model:

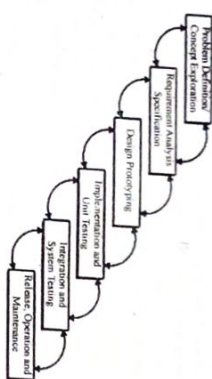


Fig. : The Waterfall Model

Evolutionary programming of this type, the goal is to work with consumers to produce and deliver the needs of end systems to the user/consumer. Development begins with the parts of the system are understood. The system was developed through the addition of appropriate features proposed by the consumer.

Modeling the second type, the goal of the process is knowing customer needs and develop a better definition of requirements for the system. Models or examples of research focused on the parts of consumers' needs are poorly understood important evolutionary programming when it is difficult to make the specification of the system in detail. Some people may agree that all systems fall into this type. However, evolutionary programming is widely used in the development of intelligence systems: building/using emulate human ability. We can not possibly make a detailed specification for software that rival humans because we do not understand how people carry out their tasks. Evolutionary approach is usually more effective than a waterfall approach to software development that it should immediately be able to meet the needs of consumers. However, in terms of technique and management, this method has a fundamental problem, namely:

The process is not visible manager requires "deliverables" that regularly to measure progress. If dikemangkan system will quickly be a waste on making each version of the document that describes the system. System-systems are typically less structured. The trend of continuous change will reduce the structure of the software. The evolution of software seem difficult and expensive.

Special skills are rarely held : it is unclear boundaries of normal skill in software engineering that may be effectively used in the model development. Most of the systems developed in this way has been implemented by a small group who have high skills and strong motivation. To solve these problems, sometimes the goal of evolutionary development is developing a system instance. This example is used to understand and menvalidasikan system specifications. This is where the evolutionary development is part of some larger process (such as the waterfall model). Due to these problems,

large-scale systems are usually not developed through this model.

Evolutionary development of more appropriate for:

- Development of a relatively small system. The problems of the existing system changes avoided by re-implementing the entire system whenever a significant change is needed. If modeling is used, not too expensive.
- Development of a system that has a relatively short masabidup. Here, the system was restricted to support some of the activities that are restricted by time. For example, a system may be developed specifically for the launch of new products. Development of the system or parts of large systems which do not allow to express the specification in detail. For example, AI systems and user interfaces.

When a system is produced using the evolutionary development model, features tend to be added without regard to an over riding design. With each modification, the software becomes increasingly disorganized. System maintenance (fixing bugs, or adding new features after the system has been released) is hampered by these problems, as it is harder to identify the source of bugs in poorly designed systems. Also, keeping the documentation up-to-date over successive "evolutions" is uncommon. Poor documentation also makes maintenance more difficult.

- (1) With evolutionary development strategies and technology used in software might be strictly experimental and not meant for widespread application or testing.
- (2) Software that is created with evolutionary development in mind might quickly find itself obsolete in the sense that the entire base code is too complex or outdated to assimilate into the technology of tomorrow.
- (3) The program has no documentation on the design and implementation of the code, so nobody understands what it does or how it does it.

Q.10 Explain the waterfall model with advantages and disadvantages.

[RTU 2015]

OR
Explain waterfall model in detail along with its various phases; also describe its advantages and disadvantages.
[R.T.U. 2017]

OR
With the help of a neat sketch, explain the waterfall model of software engineering. What are the drawbacks of this model?
[RTU 2013]

OR
Give with diagram all the steps of waterfall model.
[RTU 2011]

Give various step of a linear sequential SWDLC model.

[RTU 2010]

OR
Write merits and demerits of classical waterfall model.
[Raj. Univ. 2007]

As we can see in the fig 1 there are many process models mentioned which are used in the different areas.

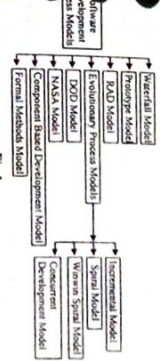


Fig.1

The waterfall model and the evolutionary models are widely used for the practical software development. The other models also have their specifications in their respective fields.

Waterfall Model : Refer to Q.5.

Advantages

- (i) The model has well-defined phase with well-defined inputs.
- (ii) It recognizes the sequence of software engineering activities that shall result in a software product.

Disadvantages

- (i) Generally user don't use this model because it takes the beginning stage.
- (ii) The customer must have patience because the model goes sequentially through phases.

Q.17 Write short notes on: (any two)

- (a) RAD model
- (b) Prescriptive model
- (c) Software characteristics

[RTU 2015]

Ans.(i) RAD Model : Refer to Q.15(a).

Ans.(ii) Prescriptive Models : Prescriptive models are related to both descriptive and predictive models. While descriptive models claim to provide insight into what has happened and predictive models help model and forecast what might happen, prescriptive models seek to determine the best solution or outcome among various choices, given the known parameters. In a way, it can be defined as the model which goes beyond predictive and descriptive models by recommending design one or more courses of action. A prescriptive model is designed from start to make

PART-A

Q.1 Write the objective of software project planning.

[R.T.U. 2018]

Ans. Software Project Planning : Project Planning is an aspect of project management that focuses a lot on Project Integration. The project plan reflects the current status of all project activities and is used to monitor and control the project.

The Project Planning tasks ensure that various elements of the Project are coordinated and therefore guide the project execution.

Project Planning helps in following :

- (i) Facilitating communication
- (ii) Monitoring/measuring the project progress, and
- (iii) Provides overall documentation of assumptions/planning decisions

The Project Planning Phases can be broadly classified as follows:-

- (i) Development of the Project Plan
- (ii) Execution of the Project Plan
- (iii) Change Control and Corrective Actions

Project Planning is an ongoing effort throughout the Project Lifecycle.

Objective Project Planning

"If you fail to plan, you plan to fail."

Project planning is crucial to the success of the project.

Careful planning right from the beginning of the project can help to avoid costly mistakes. It provides an assurance that the project execution will accomplish its goals on schedule and within budget.

Q.2 Write any two major differences between LOC and FP estimation techniques. [R.T.U. 2018]

Ans. Line of Code : The basis of the Measure LOC is that program length can be used as a predictor of program characteristics such as effort and ease of maintenance. The LOC measure is used to measure size of the software.

There are versions of LOC

DSI (Delivered Source Instructions) : It is used in COCOMO'81 as KDSI (Means thousands of Delivered Source Instructions). DSI is defined as follows :

- Only Source lines that are DELIVERED as part of the product are included - test drives and other support software is excluded
- SOURCE lines are created by the project staff - code created by applications generators is excluded
- One INSTRUCTION is one line of code or card image
- Declarations are counted as instruction
- Comments are not counted as instructions

Function Points (FP)

Functions points [Albrecht 1979] is basic data from which productivity metrics could be computed.

FP data is used in two ways:

1. as an estimation variable that is used to "size" each element of the software.
2. as baseline metrics collected from past projects and used in conjunction with estimation variable to develop cost and effort projections.

The approach is to identify and count a number of unique function types :

- external inputs (e.g. file names)
- external outputs (e.g. reports, messages)
- queries (interactive inputs needing a response)
- external files or interfaces (files shared with other software systems)
- internal files (invisible outside the system)

Ans. Input = 24
Output = 16
Inquiries = 22
Files = 4
External interfaces = 2
Average Complexity
FP = Count total $\times [0.65 + 0.01 \times \Sigma(FI)]$
Count Total = $24 \times 6 + 16 \times 7 + 22 \times 8 + 4 \times 15 + 2 \times 10$
= 468
Adjustment Factor = $\Sigma(FI)$
= 14×3
= 42
FP = $468 \times [0.65 + 0.01 \times 42]$
= $468 \times [0.65 + 0.42]$
= 468×1.07
= 500.76
Given Productivity = 6.4 FPPM
Estimated Effort = $6.4 \times 500.76 = 3204.864$

Q6. List any three common types of risks that typical software might suffer from. Explain how you can identify the risks that your project is susceptible to, support you are the project manager of a large software development project. Point out the main steps you would follow to manage the risks in your project. (16, 11, 2018)

OR
Explain risk analysis process in detail.

Ans. Risk Analysis: An engineering project is expected to produce a reliable product within a limited time, using limited resources. Any project, however, runs the risk of not producing the desired product, oversteering its allotted resource budget or overrunning its allotted time. Risk accompanies any human activity. Robert Charette has defined the risk in his book of 'Risk Analysis and Management' as:

"First, risk concerns further happenings. Today and yesterday are beyond active concern, as we are already reaping what was previously sown by our past actions. The question is, can we, therefore, by changing our actions today, create an opportunity for a different and hopefully better situation for ourselves tomorrow. This means, second, that risk involves change, such as in changes of mind, opinion, actions or places. Third, risk involves choice and uncertainty that choice itself entails. Thus paradoxically, risk, like death and taxes, is one of the few certainties of life."

So, Charette has included the three main concepts of risk: Future Happenings, Change and Choice. The future is our concern and it should be seen which risks can be high in future to manage the software. Change is our concern in the sense, that if any changes do occur, it must not go beyond the

Software Engineering

SE-21

mentioned time and cost and at last success must be achieved. Lastly, choice is our concern with respect to what methods and tools should must we use, requirements or human factors etc. having the idea of all these three concerns, the risk is defined as the possibility that the defined goals are not met.

It can be measured as the combined effect of the likelihood of the undesirable event occurring and the likelihood that a particular quantified, measured or assessed consequence will result, given the occurrence of the undesirable event. So, risk is considered as a probability that some adverse circumstances will actually occur. Risk may threaten the project, the software that is being developed for the organisation. The project manager also plays an important task to anticipate risks which might affect the project schedule or the quality of the software being developed and to take actions to avoid these risks.

Risk analysis and control is a topic of management theory. Several standard techniques exist for identifying project risks, assessing their impact and monitoring and controlling them. Knowledge of these techniques allows a project manager to apply them when necessary to increase the chances of success of a project. In the risk analysis, the likelihood and consequences of the risks are assessed.

Risk analysis can be categorized in four distinct areas:

1. Risk Identification
2. Risk Projection
3. Risk Assessment
4. Risk Management and Monitoring

1. Risk Identification
The first step towards the risk analysis is the risk identification. This step is essential so that the appropriate risk can be identified in the project. Peter Drucker has said - "while it is futile to try to eliminate risk and questionable to try to minimize it, it is essential that the risks taken be the right risks".

"Right risks" termed as the appropriate risks that are the basic causes of the failure of the project. Right risks are chosen but first it is important to identify all the risks that are obvious to both managers and practitioners.

Roughly, the risks can be partitioned into three types of risks:

- (a) **Project Risks**: These types of risks are related to the software project which affect the project schedule or resources like the delivery of the system may be delayed. For an example, the project completion time is three months. But the project is completed and delivered after four months then this risk is high.
- (b) **Product Risks**: Such risks are associated with the software product which affect the quality or performance of

the software being developed. For an example, the reliability of the software is mentioned as 75% to be accomplished but is not achieved then this risk will be high.

(c) **Business Risks**: Business risks are concerned with the business aspects of the software which affect the organisation developing or procuring the software. For an example, an inexperienced person in any business may get unsuccessful for the first time then this risk will be high. Some more business risks are:

- building an excellent product that no one really wants.
- building a product that no longer fits into the overall product strategy for the company.
- building a product that the sales force doesn't understand how to sell.
- losing the support of senior management due to a change in focus or a change in people.
- losing budgetary or personnel commitment.

2. Risk Projection
Risk projection is also known as 'Risk Estimation'. Risk projection attempts to rate each risk in two ways:

- the likelihood - risk is real
- consequences - the problems associated with the risk.

The project planner, with the management people and the technical staff members, performs the four risk projection activities:

- (a) Establishing a scale that reflects the perceived likelihood of a risk.
- (b) Delineating the consequences of the risk.
- (c) Estimating the impact of the risk on the project and the product.
- (d) Noting the overall accuracy of the risk projection so that there will be no misunderstandings.

3. Risk Assessment
Risk assessment is an activity that must be undertaken during project planning. It is one of the major planning activity in risk analysis management and consequent planning for risk control.

Due to the nature of a software project, uncertainties may be most near to the beginning of the project (just as for cost estimation). Risks can be assessed more precisely here. Due to this, although risk assessment should be performed throughout the project, it is most needed to be done in the starting phases of the project.

During this activity, three major tasks are performed:

- (a) Examining the accuracy of the estimates that were made during risk projection.
- (b) Attempt to prioritize the risks that have been uncovered.

SE-22

(i) Thinking about ways to control risks that are likely to occur.

At this point in the risk assessment, a set of triplets is established:

$$[r_i, p_i, s_i]$$

where r_i is risk,
 p_i is likelihood (probability) of the risk and
 s_i is the impact of the risk.

Risk Management and Monitoring
Risk management (also called as Risk Avoidance) is, apply an organized means of the identifying each risk area, quantifying the risk, developing ways to predict the occurrence of the undesirable event from attribute measurements, tracking the measurement on a continuing basis and developing planned alternatives or corrective actions in case the undesirable even occurs occur. It is the area that tries to ensure that the impact of risks on cost, quality and schedule is minimal.

Risk management is particularly important for software projects because of the inherent uncertainties which most projects face. These stem from loosely defined requirements, difficulties in estimating the time and resources required for software development, dependence on individual skills and requirements changes due to changes in customer needs. The project manager should anticipate risks, understand the impact of these risks on the project, the product and the business and take steps to avoid these risks.

Risk management deals with the possibility and the actual occurrence of those events that are not regular or commonly expected. The commonly expected events. Such as people going on leave or some requirements changing, are handled by normal project management. So in a sense, risk management begins, where normal project management ends. It deals with events that are infrequent, somewhat out of control of the project management and are large enough to justify special attention.

Explain project estimation technique.

Ans. Project Estimation: For an effective management accurate estimation of various measures is a must. With correct estimation managers can manage and control the project more efficiently and effectively.

Project estimation may involve the following:

- (i) **Software size estimation**: Software size may be estimated either in terms of KLOC (Kilo Line of Code) or by calculating number of function points in the software. Lines of code depend upon coding practices and Function points vary according to the user or software requirement.

(ii) **Effort estimation**: The managers estimate efforts in terms of personnel requirement and man-hour required to produce the software. For effort estimation software size should be known. This can either be derived by manager's experience, organization's historical data or software size can be converted into efforts by using some standard formulae.

(iii) **Time estimation**: Once size and efforts are estimated, the time required to produce the software can be estimated. Efforts required is segregated into sub categories as per the requirement specifications and interdependency of various components of software. Software tasks are divided into smaller tasks, activities or events by Work Breakthrough Structure (WBS). The tasks are scheduled on day-to-day basis or in calendar months.

The sum of time required to complete all tasks in hours or days is the total time invested to complete the project.

(iv) **Cost estimation**: This might be considered as the most difficult of all because it depends on more elements than any of the previous ones. For estimating project cost, it is required to consider -

- Size of software
- Software quality
- Hardware
- Additional software or tools, licenses etc.
- Skilled personnel with task-specific skills
- Travel involved
- Communication
- Training and support

Q8. Explain line of code and function point estimation techniques.

Ans. Line of Code : Refer to Q2

Advantages of LOC

1. Simple to measure
2. Drawbacks of LOC

1. It is defined on code. For example it cannot measure the size of specification.
2. IT characterise only one specific view of size, namely length, it takes no account of functionality or complexity.
3. Bad software design may cause excessive line of code.
4. It is language dependent.
5. Users cannot easily understand it.

Because of the critics above there have been extensive efforts to characterise other product size attributes, notably complexity and functionality.

Function Points (FP) : Refer to Q2.

collaboration which might help to eliminate surprises during project execution, but also ensures the sustainability of your professional status in the future.

PART-C

Q1) Consider a software project of full screen editor. The major components identified are :

- Screen Edit
 - Command Language Interpreter
 - File Input and Output
 - Cursor Movement and
 - Screen Movement
- The sizes for these components are estimated to be 6K, 7K, 2K, 4K and 3K LOC. Assume cost driver values as 1.0. Use Intermediate COCOMO model to determine:
- Final effort
 - Effort estimates for different phases
 - Total time
 - Time estimates for different phases

[R.T.U. 2018]

Ans. Size of 5 modules are :

Screen edit = 6 KLOC
Command language interpreter = 7 KLOC
File input and output = 2 KLOC
Cursor movement = 4 KLOC
Screen movement = 3 KLOC

Total = 22 KLOC

- Let us assume that significant cost drivers are
- Required software reliability is high i.e. 1.15
- Product complexity is high i.e. 1.15
- Analyst capability is high i.e. 0.86
- All other drivers are nominal i.e. 1.00

Hence

Table : Co-efficients for Intermediate COCOMO

Project	a	b	c	d
Organic mode	3.2	1.05	2.5	0.38
Semidetached mode	3.0	1.12	2.5	0.35
Embedded mode	2.8	1.20	2.5	0.32

$$EAF = 1.15 \times 1.15 \times 0.86 = 1.1373$$

The initial effort estimate for the project

$$E = a \cdot (KLOC)^b \cdot EAF$$

$$= 3.2(22)^{1.05} \times 1.1373 = 84.06 \text{ M}$$

Phase wise cost and schedule estimates
 $E_p = 1/p$

Since size is only 22 KLOC, it is an organic small model. Phase wise effort distribution is given below:

$$\begin{aligned} \text{System Design} &= 0.16 \times 84.06 = 13.45 \\ \text{Detailed Design} &= 0.26 \times 84.06 = 21.85 \\ \text{Module code and test} &= 0.42 \times 84.06 = 35.30 \\ \text{Integration and test} &= 0.16 \times 84.06 = 13.45 \\ D &= c/(E \cdot M) \\ &= 2.5/(84.06/0.38) \\ &= 79.85 \text{ M} \end{aligned}$$

(iv) Phase wise development time duration is:

$$\begin{aligned} \text{EDP} &= pD \\ \text{System Design} &= 0.19 \times 79.85 = 15.17 \\ \text{Detailed Design} &= 0.24 \times 79.85 = 19.16 \\ \text{Module code and test} &= 0.39 \times 79.85 = 31.14 \\ \text{Integration and test} &= 0.18 \times 79.85 = 14.37 \end{aligned}$$

Q2) Write a detailed note on COCOMO Model. Explain its types.

Ans. Software Engineering / COCOMO Model :

Cocomo (Constructive Cost Model) is a regression model based on LOC, i.e. number of Lines of Code. It is a procedural cost estimate model for software projects and often used as a process of reliable predicting the various parameters associated with making a project such as size, effort, cost, time and quality. It was proposed by Barry Boehm in 1970 and is based on the study of 65 projects, which make it one of the best documented models.

The key parameters which define the quality of any software products, which are also an outcome of the Cocomo model, are primarily Effort & Schedule :

- Effort : Amount of labor that will be required to complete a task. It is measured in person-months units.
- Schedule : Simply means the amount of time required for the completion of the job, which is, of course, proportional to the effort put. It is measured in the units of time such as weeks, months.

Different models of Cocomo have been proposed to predict the cost estimation at different levels, based on the amount of accuracy, and correctness required. All of these models can be applied to a variety of projects, whose characteristics determine the value of constant to be used in subsequent calculations. These characteristics pertaining to different system types are mentioned below.

Boehm's definition of organic, semi detached and embedded systems:

1. Organic : A software project is said to be an organic type if the team size required is adequately small, the problem is well understood and has been solved in the past and also the team members have a nominal experience regarding the problem.

2. Semi-detached : A software project is said to be a Semi-detached type if the vital characteristics such as team-size, experience, knowledge of the various programming environment lie in between that of organic and Embedded.

The projects classified as Semi-Detached are comparatively less familiar and difficult to develop compared to the organic ones and required more experience and better guidance and creativity. Eg: Compilers or different Embedded Systems can be considered of Semi-Detached type.

3. Embedded : A software project with requiring the highest level of complexity, creativity and experience requirement fall under this category. Such software requires a larger team size than the other two models and also the developers need to be sufficiently experienced and creative to develop such complex models.

All the above system types utilize different values of the constants used in Effort Calculation.

Types of models : COCOMO consists of a hierarchy of three increasingly detailed and accurate forms. Any of the three forms can be adopted according to our requirements.

These are types of COCOMO model:

- Basic COCOMO Model
- Intermediate COCOMO Model
- Detailed COCOMO Model

The first level, Basic COCOMO can be used for quick and slightly rough calculations of Software Costs. Its accuracy is somewhat restricted due to the absence of sufficient factor considerations.

Intermediate COCOMO takes these Cost Drivers into account and Detailed COCOMO additionally accounts for the influence of individual project phases, i.e. in case of Detailed it accounts for both these cost drivers and also calculations are performed phase wise henceforth producing a more accurate result. These two models are further discussed below.

Estimation of Effort : Calculations

1. Basic Model :

$$E = a \cdot (KLOC)^b$$

The above formula is used for the cost estimation of for the basic COCOMO model, and also is used in the subsequent models. The constant values a and b for the Basic Model for the different categories of system:

Software Projects	A	B
Organic	2.4	1.05
Semi Detached	3.0	1.12
Embedded	3.6	1.20

The effort is measured in Person-Months and as evident from the formula is dependent on Kilo-Lines of code. These formulas are used as such in the Basic Model calculations, as not much consideration of different factors such as reliability, expertise is taken into account, henceforth the estimate is rough.

2. Intermediate Model :

The basic COCOMO model assumes that the effort is only a function of the number of lines of code and some constants evaluated according to the different software system. However, in reality, no system's effort and schedule can be solely calculated on the basis of Lines of Code. For that, various other factors such as reliability, experience, Capability. These factors are known as Cost Drivers and the Intermediate Model utilizes 15 such drivers for cost estimation.

Classification of Cost Drivers and their attributes:

- Product attributes :**
 - Required software reliability extent
 - Size of the applications database
 - The complexity of the product
- Hardware attributes :**
 - Run-time performance constraints
 - Memory constraints
 - The volatility of the virtual machine environment
- Personnel attributes :**
 - Analyst capability
 - Software engineering capability
 - Applications experience
 - Virtual machine experience
 - Programming language experience
- Project attributes :**
 - Use of software tools
 - Applications of software engineering methods
 - Required development schedule

Cost drivers	Very Low	Low	Nominal	High	Very High
Product Attributes					
Required Software Reliability	0.75	0.88	1.00	1.15	1.40
Size of Application Database	0.94	1.00	1.08	1.15	1.16
Complexity of The Product	0.70	0.85	1.00	1.15	1.30
Hardware Attributes					
Runtime Performance Constraints			1.00	1.11	1.30
Memory Constraints			1.00	1.06	1.21
Volatility of the virtual machine environment	0.87	1.00	1.15	1.30	
Required turn about time	0.94	1.00	1.07	1.15	

Personnel Attributes	Very Low	Low	Nominal	High	Very High
Analyst capability	1.46	1.19	1.00	0.86	0.71
Software engineer capability	1.29	1.13	1.00	0.91	0.82
Software engineer capability	1.42	1.17	1.00	0.86	0.70
Virtual machine experience	1.21	1.10	1.00	0.90	
Programming language experience	1.14	1.07	1.00	0.95	
Project Attributes					
Application of software engineering methods	1.24	1.10	1.00	0.91	0.82
Use of software tools	1.24	1.10	1.00	0.91	0.83
Required development schedule	1.23	1.08	1.00	1.04	1.10

The project manager is to rate these 15 different parameters for a particular project on a scale of one to three (written, depending on these ratings, appropriate cost driver values are taken from the above table. These 15 values are then multiplied to calculate the EAF (Effort Adjustment factor).

The Intermediate COCOMO formula now takes the form :

$$E = (a(KLOC)^b) \cdot EAF$$

The values of a and b in case of the intermediate model are as follows :

Software Projects	A	B
Organic	3.2	1.05
Semi Detached	3.0	1.12
Embedded	2.8	1.20

3. Detailed Model : Detailed COCOMO incorporates all characteristics of the intermediate version with an assessment of the cost driver's impact on each step of the software engineering process. The detailed model used different effort multipliers for each cost driver attribute. In detailed COCOMO, the whole software is divided into different modules and then we apply COCOMO in different modules to estimate effort and then sum the effort.

The Six phases of detailed COCOMO are

- Planning and requirements
- System design
- Detailed design
- Module code and test
- Integration and test
- Cost Constructive model

PREVIOUS YEARS QUESTIONS

PART-A

Q.1 List the requirement analysis tasks. [R.T.U. 2018]

Ans. Requirement Analysis Tasks : The requirements analysis task is a process of discovery, refinement, modelling and specification. The software scope is initially established by the software engineer and refined during software project planning in detail. Models of the information, operation behaviour and control flow and data domains are created and at last specification is made.

To initiate this task, first alternative solutions are analyzed and allocated to various software elements. After allocation, the following steps are performed one by one.

- Problem Recognition
- Problem Evaluation and Synthesis
- Modelling
- Specification
- Validation Criteria
- Review

Q.2 Why is accuracy important in data dictionary? [R.T.U. 2018]

Ans. Data Dictionary : Data dictionary is very important for representing the characteristics of every data object and control items that acts as the building block of analysis model. This is done by data dictionary it is a quasi formal grammar for describing the contents of a data object.

Data dictionary is an organized listing of all data elements that are pertinent to the system. With precise, vigorous definitions so that both user and system analyst will have a common understanding of input, output components of stores and intermediate calculation. CASE tools are used for this purpose.

Data dictionary is an essential and important tool in the software development process. As its name indicates, it is a comprehensive definition of all the data (and control) items (elements) in a given software or it can stated as-it is a structured central repository of data about data. In a data dictionary one should find a clear and complete definition of each data item and its synonyms. The data dictionary often becomes a source document for specification and design of input processing, files and data structures, processing algorithms and output processing. Typically, data dictionary development begins very early in the development process

and expands as the software development process continues and more detail becomes available. To describe the data dictionary, sufficient information should be collected about data items. As such, there are a number of methods to collect the information about data items. In some methods, the data dictionary as a repository is not developed and the related information about the data items is collected in the requirements, design and interface specifications only. In the development methods, a separate document as a data dictionary is made to reside information.

Q.3 Write short note on Control and Process Specification. [R.T.U. 2017]

Ans. Control Specification : Control specification contains a State Transition Diagram (STD) that is the sequential represented model of the behavior of the system. It also contains a Process Activation Table (PAT). PAT represented information contained in the STD in the form of process. It indicates which process will be invoked whenever an event occurs. It is like the manual of the STD.

Process Specification : Process Specification is used to describe all flow model process that appear at the final level of analytical modelling. It includes text, information in program defined language mathematical equations and charts.

Q.4 Explain the behavior modelling as a part of Structured Analysis of Software Development. [R.T.U. 2016]

OR
Write short note on behavioural modelling. [R.T.U. 2017]

Ans. Behavior Modelling : Behavior modelling is the minimal form of logical abstraction of a software system that fits nicely into how the human mind thinks and allows for one to understand complex systems and increases one's ability to understand and solve difficult problems.

Behavioral Modelling contains a set of tools for performing a wide variety of analysis on a model and incorporating the analysis results into the model. Behavioral Modelling lets you modify the model design to reflect the desired solution. Behavioral Modelling tools can be used to:

- Create feature parameters based on measurements and analysis of the model.
- Create geometric entities based on measurements and analysis of the model.
- Create new types of measurements tailored to application-specific needs.

Software Engineering

- Analyze the behavior of measured parameters as variable dimensions and parameters change.
- Automatically find dimension and parameter values that achieve a desired behavior of the model.
- Analyze the behavior of measured parameters within a specified design space.

The basic building blocks of Behavioral Modelling are:

- Field point
- Analysis feature
- Persistent analysis display
- User-Defined Analysis (UDA)
- Sensitivity, Feasibility, and Optimization Studies
- Optimization feature
- Multi-Objective Design studies
- External Analysis
- Motion Analysis

Q.5 Describe Facilitated Application Specification Techniques (FAST). [R.T.U. 2013]

Ans. Facilitated Application Specification Technique ("FAST")

It is a technique for requirements elicitation for software development. The objective is to close the gap between what the developers intend and what users expect. It is a team-oriented approach for gathering requirements.

Goal: The main goal is to identify the problem, propose solutions, negotiate different approaches, and specify a preliminary set of software requirements in an atmosphere that is conducive to accomplish the goal.

Tasks to be performed

- Meetings are conducted at a neutral site attended by both developers and users
- The group establishes rules for preparation and participation.
- A facilitator controls the meeting.
- A definition mechanism is used.
- An agenda is suggested that with enough formality to cover all important points but informal enough to encourage the free flow of ideas.

Output

After initial meeting, user and developer should write a one or two product request form. Before the next meeting it is distributed to all other attendees. Each attendee is asked to make the following lists:

- List of objects
- List of services
- List of constraints
- Performance criteria

PART-B

Q.6 Draw Data Flow Diagram of Traffic Control System for all 3 levels: A-level, 1-level and 2-level. [R.T.U. 2018]

Ans. Traffic Control System Level-0 DFD:

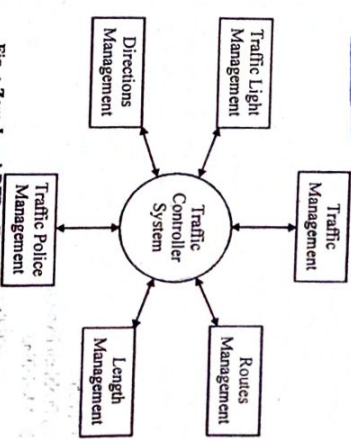


Fig. : Zero Level DFD - Traffic Controller System

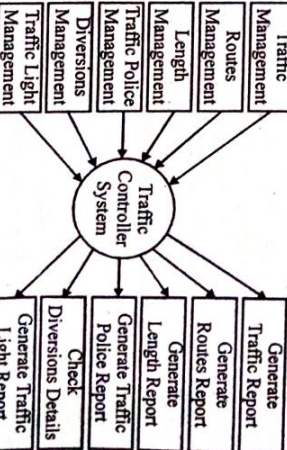


Fig. : First Level DFD - Traffic Controller System

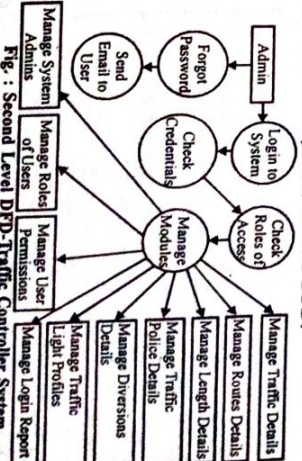


Fig. : Second Level DFD-Traffic Controller System

- Each external file type may have an associated tool which may be applied to the file
- Each external file type may be represented as a specification on the user's display
- Facilities should be provided for the icon representing an external file to be defined by the user
- When a user selects an icon representing an external file, the effect of that selection is to apply the tool associated with the type of external file to the file represented by the selected icon.

Q.8 Write short note on Finite State Machine (FSM).

[RTU 2017, 2016, 2013, 12]

OR

Explain Finite State Machine (FSM) Models.

[RTU 2014]

Ans. Finite State Machine (FSM) : Finite state machine model is a simple, widely known and important model for describing control aspects. It is a model of the system to describe an entity which is characterized by its operations and behavior. Such machines are basically appropriate tools to express the requirements and design specifications of software product.

FSM models are represented by a set of states and specifies transitions between the states. The transition from one state to another state is based on the input. So, such models show system states and events which cause transition from one state to another.

A Finite State Machine (FSM) consists of a finite set of symbols that are given below.

J = a finite set of states

ϵ = a finite set of input

S = the starting state

δ = a transition, from (input $\times$ state) to 2 state ξ , which can be a partial function, i.e. can be undefined for some values of its domains.

$\delta : \text{Input} \times \text{state} \rightarrow \text{state}$

$\delta : \epsilon \times J \rightarrow J$

These notations are used to make the finite state machine diagram or transition tables. To draw a diagram, first, we draw a circle for each state in the diagram. Then for each state and possible input, we draw a directed arrow to indicate the transition from one state to another.

Example 1 : Draw a FSM model for the following description :

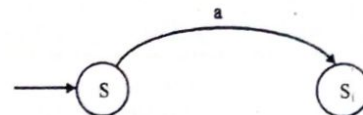
$T = \{S, S_1\}$

$\epsilon = \{a\}$

S = starting state

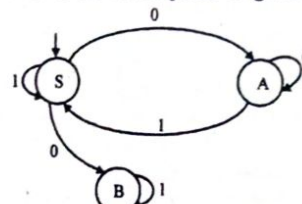
$\delta(S, a) = S_1$

Sol.



When an input does not result in a state change, we can indicate this in the diagram by showing an arrow originating and ending in the same state.

Example 2 : Give the description of given FSM :



Sol.

$J = \{S, A, B\}$

$\epsilon = \{0, 1\}$, S = Starting state

$\delta(S, 0) = A$, $\delta(S, 1) = S$

$\delta(A, 0) = A$, $\delta(A, 1) = S$

$\delta(B, 0) = B$, $\delta(B, 1) = B$

Example 3 : The FSA in given fig. recognizes the following set of words : {bet, ball, beg} :

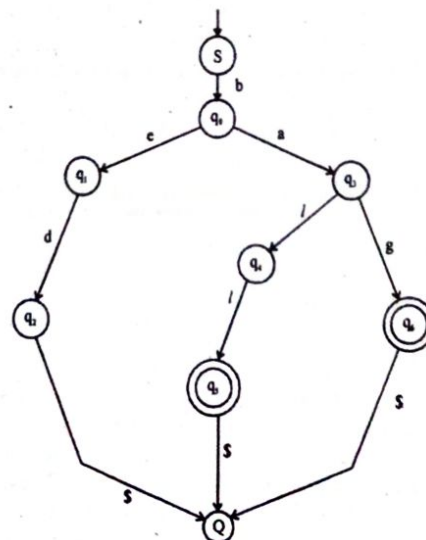


Fig.

Sol.

$\epsilon = \{b, e, d, a, l, g\}$

$J = \{S, q_0, q_1, q_2, q_3, q_4, q_5, Q\}$

S = Starting state

$A = Q$ (Accepting state)

$\delta(S, b) = q_0$, $\delta(q_3, l) = q_4$

$\delta(q_0, e) = q_1$, $\delta(q_3, g) = q_5$

$\delta(q_0, a) = q_3$, $\delta(q_4, l) = q_5$

$\delta(q_1, d) = q_2$, $\delta(q_5, s) = Q$

$\delta(q_2, s) = Q$, $\delta(q_5, s) = Q$

and

$f = \{q_2, q_3, q_4\}$

Q.9 Write short note on QFD (Quality, Function, Deployment) and use-cases. [R.T.U. 2016]

Ans. QFD (Quality, Function, Deployment) : QFD is a process used to determine product development characteristics that combine technical requirements with customer performances. It is a process that is used in product design to identify the priorities for customer needs and expectations and to transfer these priorities into the technical design of product. QFD use the house of quality as a visual model.

Use-case- an abstraction that describes a set of scenarios

Use-case is an effective tool to capture, represent and analyze the functional requirements of a system. Use-cases provide the external view of what the system does, and feed into OOA and design. The use-case model specifies the behavior of a system.

A system gets input from some external entities. It provides output to some external entities. Hence, the system interacts with various external entities. In use-case terminology, any entity that interacts with the system is termed as actor. An actor is usually a role played by a human user. However, it can also be any entity such as a third-party organization or another computer system. A single user may play different roles at different times, and thus be represented as different actors. An actor in a use-case diagram is analogous to an external entity in a data flow diagram.

The use-case approach requires the analyst to determine all the potential actors involved in a system. Actors are external to the system and make use of it. An actor makes use of a system in different ways. Each use-case describes a sequence of transactions between one or more actors and the system. A use-case may involve a number of actors and an individual actor may also make use of several use-cases.

The use-case is graphically depicted by a use-case diagram. In the use-case diagram, the use-cases are represented by ovals (ellipse) and actors are represented by stick-man. Let us consider interactions between a banking system and its customers. A customer withdraws money from the bank. He can also deposit money in the bank. Hence, we can represent:

'Customer' as an 'Actor'

'Withdraw money' as a 'Use-case'

'Deposit money' as another 'Use-case'

to fully develop the context around a product and what it must look like to help developers implement their work.

- Interview various sources. Get information for the requirements document from business leaders, engineers, developers, sales reps, customers or anyone else with important information about needs for product development.
- List system requirements or properties. One of the important elements of requirements is the system requirements, or how the product will interact with a given system for a workstation or network.
- Identify any constraints for the project. Explaining restrictions or constraints within the requirements document will help further guide those who are working on the software or IT product.
- Consider any interface requirements. Interface requirements are an important part of this document because they determine how the end-user will view the product. They often have a critical influence on the user-friendliness of a product.
 - Identify color schemes, command button requirements and any other part of a successful interface.
 - Keep in mind the programming tools that will be used to develop the project or product when listing interface requirements. This will provide more guidance for developers and others.
- Identify parameters like cost and scheduling. These practical considerations should also be part of a requirements document.
- Work up a development plan. Along with the above information, a good requirements document will often include further instructions about development of the product. This can take one of a number of forms and may require a bit of creativity.
- Insert visuals. Graphic mockups of a product keep the project fresh and appealing to the eye, while providing more detail about how something will look.
- Categorize and organize requirements. Find a way to neatly fit each of these categories of requirements into a single document.
- Write the requirements document. Utilize good document planning strategies in constructing something that reads well, where individual areas of the requirements document are easy to access.

Q.1 (a) What is structured analysis? Explain control flow diagram and control process specification by giving a suitable example.

(b) Describe software prototyping in detail.

[R.T.U. 2018]

Ans.(a) Structure Analysis : Structured analysis is a software engineering technique that uses graphical diagrams to develop and portray system specifications that are easily understood by users. These diagrams describe the steps that need to occur and the data required to meet the design function of a particular software. This type of analysis mainly focuses on logical systems and functions and aims to convert business requirements into computer programs and hardware specifications. Structured analysis is a fundamental aspect of system analysis.

There are three orthogonal views related to structured analysis:

- Functional View :** This involves box diagrams, which define the work that has been done and the flow of data between things done, thereby providing the primary structure of a solution.
- Data View :** This comprises the entity relationship diagram and is concerned with what exists outside the system that is being monitored.
- Dynamic View :** This includes state transition diagrams and defines when things happen and the conditions under which they may happen.

Control Flow Diagrams : A control flow diagram helps us understand the detail of a process. It shows us where control starts and ends and where it may branch off in another direction, given certain situations. Let's say you are working on software to start a machine. What happens if the engine is flooded, or a spark plug is broken. Control then changes the flow to other parts of the software. We can represent these branches with a diagram. The flow diagram is helpful because it can be understood by both stakeholders and systems professionals. Although some of the symbols might not be fully understood by the layperson, they can still grasp the general concept.

Example

Before we cover all of the symbols in the diagram, let's look at an example. Keeping with the idea of an engine-start program, the following shows flow through the software:

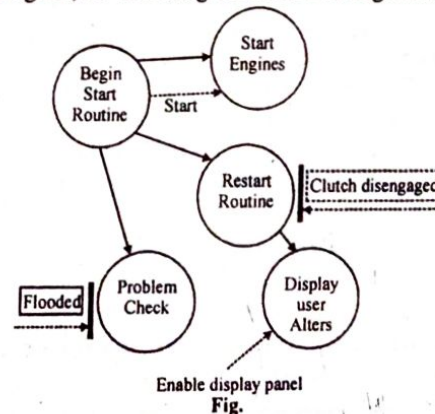


Fig.

Even to the non-IT person, it is fairly clear that the software will attempt to start engines. If errors occur, then it will branch off into different directions. The symbols might not be completely clear, so we'll look at those next.

Symbols

The general flow should make sense:

- try to start the engines
- run a problem check
- display alerts if it can't start
- disengage the clutch
- try restarting

The vertical lines and dashed arrows :

Vertical Bar



The vertical bar indicates input or output from the current control specification, that is the diagram we are working on. Think of the bar as a window INTO or OUT OF the entire specification/flow. Additional notes, including an arrow, indicate further detail of the flow through that window.

Dashed Arrow

The input to the problem check step is Flooded. The input into the restart routine is clutch disengaged. These inputs are noted by the dashed arrow:



Control Process Specifications (CSPEC) : The control process specifications (CSPEC) is used to indicate how the software behaves when an event or control signal is sensed and which processes are invoked as a consequence of the occurrence of the event. The control process specification (CSPEC) contains a number of important modeling tools. The control specification represents the behavior of the system in two ways. The CSPEC contains a state transition diagram that is sequential specification of behavior. It also contains a process activation table (PAT) - a combinatorial specification of behavior.

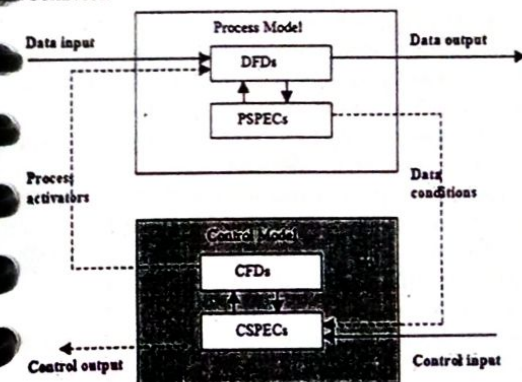


Fig. : The relationship between data and control models

Ans.(b) Prototyping : Prototyping is method that can be used for problem analysis, which is getting a lot of attention now-a-days. It takes a very different approach to problem analysis to conceptual modeling-based approaches (example - structured analysis, object-oriented modeling).

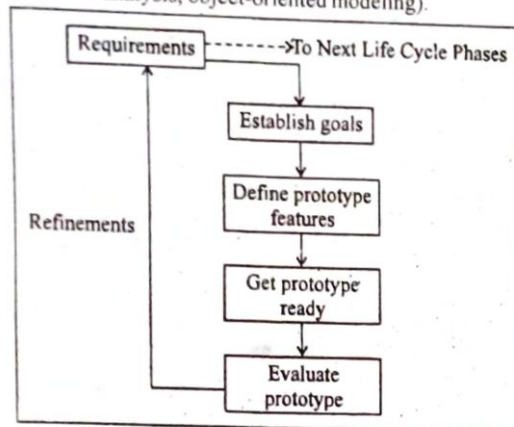


Fig. 1 : Use of Prototype of Requirements

In prototyping, a partial system is constructed as a partial result of the communication of requirements between the users and the analysts. The prototype helps to the clients, users and developers to gain a better understanding of the problem and the needs. Prototyping is frequently used to provide early feedback to customers and users and to improve communication of requirements between the analysts and the users. A prototype can effectively provide the users a 'look-and-feel' and convey a sense of how the system will work.

The idea behind prototyping is that clients and the users can assess their needs much better if they can see the working of a system, even if the system is only a partial system. Prototyping emphasizes that actual practical experience is the best aid for understanding needs. By actually experimenting with a system people can say, "I don't want this feature" or "I wish it had this feature" or "This is wonderful".

Prototyping Scenario

For maximum benefits, prototyping should be carried out in a systematic manner, as shown in fig.1 with the following steps :

Step-1 : Establish goals for the prototype : First, the objectives of the prototype should be laid down and agreed on. Prototypes may be developed with different objectives, for example, to discuss the user interface, to discuss functionality from a given context etc. If the developers or the end-users donot know the objectives, the prototype may create further misunderstandings.

Step-2 : Define prototype features and functionality :

In this step, the detailed list of features and functions that are to be demonstrated in the prototype are documented,

discussed and agreed on. This document should state the two aspects that what will be in the prototype and what will not be included in the prototype.

Step-3 : Get the prototype ready

In this step, the prototype is developed and reviewed against the list of features and functionality, to ensure that all the required features and functionality are developed.

Step-4 : Evaluate the prototype

This step starts with reviewing the goals, features and functions defined for the prototype to establish the expectation of the prototype. The prototype is then executed. User's and developer's feedback is collected. This feedback typically includes new requirements, changes to user interface, dropping of requirements etc. The evaluation report is then used to elicit and analyze new requirements.

Approaches of Prototyping

There are types of approaches to prototyping - **The Throwaway Approach and The Evolutionary Approach.** In the Throwaway, Approach prototyping, the use of the prototype ends, once all the requirements are clearly documented. This prototype is not used for the actual construction of the system. Typical characteristics of this approach are :

- The prototype is built using an environment and tools different from the tools and environment that will be used to build the system.
- Not much importance is given to prototype's "internals" such as code.

- The prototype is usually built very quickly as compared to the evolutionary prototype.
- The throwaway prototype converts the poorly understood requirements into the well understood requirements, as shown in fig.2.

Throwaway prototyping

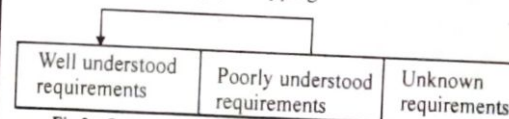


Fig.2 : Conversion of Poorly Understood into Well Understood Requirements

In the **Evolutionary Approach**, the prototype is built in such a way that it can be used in the construction phase. The evolutionary approach requires the following :

- The tools, languages and platforms used to develop the prototype be the same as the actual development platform selected.
 - To minimize the reuse of the prototype, important conventions for the design and programming be laid down before the development of the prototype begins.
 - The prototype should be reviewed both externally (behavior) and internally (design and code).
- The evolutionary approach develops a considerable part of the system during the requirements and the design stages but slows down the requirements definition process and also detracts the attention of the developers to internal details.

Table : Comparison between Prototyping Model and Waterfall Model

S.No.	Prototyping Model	Waterfall Model
1	The basic principle of this model is that a throwaway prototype is built to help in understanding the requirements during the requirement analysis phase only.	The Waterfall Model is an engineering model, designed to be applied to the development of the software products. The software development starts from the system level and develops step-by-step through the various phases.
2	This model follow the three step to build a software : (1) Listen to Customer (2) Build the Model (3) Customer Evolution	This model follows the six step to build a software : (1) Concept formation (2) Developing the software specification (3) Software design (4) Coding and unit testing (5) Software integration and testing (6) Release, Implementation and Maintenance.
3	The misunderstanding between software developer and user may be identified. Missing features and facilities may defined out.	It recognizes the sequence of Software Engineering activities that shall result in a Software Product.
4	Any non-confirm hard or confusing facilities or characteristics may be identified.	It is quite difficult all the requirement at the beginning stage.
5	Generally, user use this model because the development of software is faster.	Generally, user don't use this model because it takes time.
6	Maintenance effort is reduced because the systems meet the user needs.	Maintenance effort is high because the systems not meet clearly to the user.

7	This model has the good interaction with the users.	This model developer have very little interaction with the users.
8	It enables develop to cope and lack of clarity in requirement, it gives the users the opportunity to change his mind before communication to the final system.	This model is essentially inflexible because it assumes that software creation activities can be compartmentalize.
9	In the end of model either the customer will give its feedback to the developer or user satisfaction and acceptance will be achieved.	This model is an ideal model. It can be categorized as linear rigid and monolithic. The monolithic waterfall model in the sense that all planning is oriented to a single delivery date.
10	Example : Online-Reservation System	Example : Railway Reservation System

Q.14 Explain Major elements of DFD's and CFD's.

[R.T.U. 2016]

OR

Describe the structural analysis for the system and differentiate the DFD and CFD in detail with example. [R.T.U. 2017]

What do you understand by structured analysis. Explain data and control flow diagrams. [RTU 2015]

Ans. Refer to Q.10.

Q.15 Write short note on data dictionary.

[RTU 2017, 2016, 2012]

OR

What do you understand by data dictionary? Where and how it is used? [RTU 2015, 13, 10, 2009, Raj, Univ. 2004, 2002, 2000, 1995]

Ans. Data Dictionary : Refer to Q.2.

Format of Data Dictionary : Although the format of data dictionaries varies from method to method, most use the following table, called as Project Data Element Sheet and information about the data item :

Name	Alias	Use	Context	Supplemental Information
Table 1 : Project Data Element Sheet	Table 2 : Content Description Symbols and their Definitions	Table 3 : Project Data Element Sheet of a Telephone Number	Table 4 : Data Dictionary	Table 5 : Data Dictionary

In this table :

Name : a format or primary name of the data or control item, data storage or external entity.

Alias : other names or acronyms of data item.

Use : a listing of processes which use the data or control item and how or when it is used.

Constant Description : standard format for representing the contents of data or control item.

Supplement/Additional Information : other information such as limitations, restrictions, defaults or initial value etc.

The description format is frequently given a highly structured form but it is to be noted that consistency in naming

must be followed. Except the content description, all the information may be provided directly. To develop the content description of a data item, some notations (symbols) are used, as shown in table-2.

Table 2 : Content Description Symbols and their Definitions

Symbol	Definition
=	Is composed of
+	and
{ }	either
{ }	"n" repetitions of
{ }	optional data
..	delimits comments

As an example, using the tables 1 and 2, a telephone number might appear in a data dictionary as, shown in table 3 :

Table 3 : Project Data Element Sheet of a Telephone Number

Name	Alias	Use	Context	Supplemental Information
Telephone Number	None	Telephone number= (local) *dial phone	* Phone Setup	number/long distance (number) local number= prefix+access number long distance number= 0 + area code = local number prefix= [2520/5224] access number = "any three-digit string" area code = T141, 1511

In this way, a data dictionary can be maintained.

Classes of Data Items

Basically, there are three classes of data items to be considered, as follows :

1. **Data Element** : The smallest unit of data that can not be further decomposed.

Software Engineering

2. **Data Structure** : Once data items have been defined, the data structure required by the particular system under consideration can be identified. The terms normally used to define data structures are :

Group Items : Combinations of data items for some logical purpose. For an example, the data item "Customer address" might well be combined with the data item "Customer name" to form a group item i.e. "Customer name & address".

Records : Collections of data items or group items which are gathered to define an entity for a particular processing purpose. For an example, the data item "Customer number" and the group item "Customer name and address" might be brought together to form a customer name and address record: where by in a sales invoicing application, the customer number input to the system on a delivery note should be matched with the customer number in the customer name and address record to pick up the name and address.

Files : Collections of records which are gathered together to allow processing of data about several entities. For an example, if in the invoicing system, all the deliveries for a particular day were processed to produce invoices then a file of customer name and address records would be required to match against all the different customers to whom a delivery had been made.

Database : A collections of records which are interrelated for purposes of processing to meet user requirements. For an example, in a deliveries database, the delivery record would be interrelated with the order record, product record, customer record and invoice record, in such a way as to minimize duplication of data.

3. **Data Stores and Data Flows** : Data stores and data flows are made up of data structures and elements. Consequently, data store and flow entries contain data structures and data elements. The data in these entries is described using any of the above notations. Entries about data stores and flows also include other references. For example, the data flow description contains the origin and destination of the data flow. It also contains the physical description of the data flow, in this case stating that it takes the form of paper invoices. Data store descriptions also contain physical information, such as the medium used to store the data and where it is located.

Advantages of Data Dictionary

The advantages of using a data dictionary are :

1. It is a valuable reference in any organization because it provides documentation.
2. It improves the communication between system analyst and user by establishing consistent definitions of various items and procedures.
3. It is a good tool for managers, operators and other members of the development team to understand requirements and designs.

4. It helps the analyst to simplify the structure for meeting the data requirements of the system.

5. It is just like a store of all data elements information that can link all phases of software development life cycle.

6. It is used to remove the redundancy in data definition.

7. It is an important step in building a database. Most database management systems have a data dictionary as a standard feature.

8. During implementation, it serves as a base against which developers compare their data descriptions.

Disadvantages of Data Dictionary

1. There are some disadvantages which are given below :
1. It does not provide functional details.
2. It is not acceptable to many nontechnical users.

Q.16 Explain the requirement management phase of the requirement engineering.

[R.T.U. 2016]

OR

What is requirement analysis. Explain the requirement analysis tasks and principles. [RTU 2015]

Ans. Requirement Analysis : Requirement analysis is a correct system that work according to user requirements.

Correct systems can only be built if it is known exactly what are the user requirements and what system must do. One of the most important factors in building correct systems, therefore, is to clearly define what the system must do. There are two major steps to be considered. The first step is to agree on a Broad Conceptual Solution. The next step is a detailed analysis of user requirements followed by a system specification.

Requirement Fundamentals**(1) Software Requirements**

(a) A condition or a capability needed by a user to solve a problem or to achieve an objective.

(b) A condition or a capability that must be met or possessed by a system to satisfy a contract, standard, specifications or other formally imposed document.

The requirements which are coined during the customer developing communication should be fulfilled or satisfied. As the requirement phase proceeds, it ultimately ends with the Software Requirement Specification (SRS).

(2) Characteristic Requirements :

To ensure that both developers and customers understand the requirements properly, it is important that the requirements be of high quality.

- (a) **Requirements Correctors** : To be sure about requirements correctors, both developer and customer should review them that they are stated without any error.
- (b) **Consistency of Requirements** : Consistency of requirements does mean no conflicting or ambiguous

requirement. For an example, if one requirement states that a maximum of 5 users can use system at one time and another requirement says that in a certain situation there may be 20 simultaneous uses. Those requirements are said to be inconsistent.

- (c) **Requirement Completeness** : The set of requirement is complete if all possible states, state changes, inputs, products and constraints are described in the form of requirements. A system description is externally complete if the system description contains all the information about the environment desired by the customer. A requirement description is internally complete if there are no undefined references among the requirements.
- (d) **Realistic Requirements** : Realistic requirements are the one which are possible to be performed practically. For example, suppose a system requires user to connect to a main computer located several thousand miles away and the response time for remote users is the same as for local users.
- (e) **Appropriateness of Requirements** : Sometimes a requirement restricts the developers unnecessarily or includes functions that are not directly related to the problem at hand. For example, a General may decide that a tank's new software system should allow soldiers to send and receive electronic mail, even though the main purpose of the tank is to traverse uneven terrain.
- (f) **Verification of Requirements** : We must be able to write test cases to demonstrate that the requirements have been met.
- (g) **Traceability of Requirements** : Each system function should be traced to a set of requirements that modulate it. It will easy to find the set of requirements that deals with a specific aspect of the system.

(3) Types of Requirements :

- | | |
|-----------------------------|-------------------|
| (a) Physical Environment | (b) Interface |
| (c) Users and human factors | (d) Functionality |
| (e) Documentation | (f) Data |
| (g) Resources | (h) Security |
| (i) Quality Assurance | |

Requirement Analysis Tasks : Refer to Q.1.

(i) **Problem Recognition** : In the starting, the system analyst studies the system specification and software project plan to have the basic idea of the problem to understand it. So, it is essential for him, to have the complete base about the software. Then he reviews the scope which was used to make the planning estimations.

After having such overview of the software, to be built, a communication must be established to be sure about the problem recognized. So, the analyst will communicate with the management and the technical staff of the user/customer organization and the software development

organization. The project manager acts as a coordinator of the meeting of the software to discuss the important aspects of the software to be followed. The goal of the analyst is recognition of the basic problem elements as perceived by the user/customer.

(ii) **Problem Evaluation and Synthesis** : This is the next step to be followed, where the analyst must evaluate :

- the flow and content of information
- define and describe all software functions
- understand software behaviour that affect the system
- various system interface characteristics and
- uncover design constraints

Each of these tasks are used to describe the problem so that the overall approach or solution may be synthesized. Once problems have been identified, the analyst determines the system. After getting the desired information (input and output), the analyst begins to synthesize one or more solutions. This process of evaluation and synthesis step will be continued until both analyst and customer feel confident that software can be adequately specified for subsequent development steps. During this task the analyst's main concern is on "what" to be accomplished not, "how" to accomplish.

(iii) **Modelling** : During this task, the analyst creates one or more models of the system. A model presents an abstract description of the system. These models also act as a bridge between the users, customers and the developers, as the models are easily understandable by all the parties. Generally, models simplifies the situation and picks out most important aspects of a situation. It is imperative that the analyst selects the most suitable model for the situation at hand and uses more than one model, if necessary.

(iv) **Specification** : It is an agreement between the end user and the system developer. Generally, the specification is a document that completely describes what the proposed software should do without describing how the software will do it. The basic goal of the requirement phase is to produce the SRS, which describes the complete external behaviour of the proposed software.

(v) **Validation Criteria** : Once basic information, functions, performance, behaviour and interfaces are described, validation criteria are specified to demonstrate understanding of a successful software implementation.

(vi) **Review** : As the last step, the review is conducted to check the progress of the project and to check whether all the work (module or software) is performed well according to the user requirements. During the review, the review members may give their comments, feedbacks or praise on the work done. And as a result of review, modify function, performance, information representations, constraints or validation criteria.

SOFTWARE DESIGN 4

CHAPTER IN A NUTSHELL

Design Fundamentals

The following set of fundamental software design concepts have evolved over the last four decades :

1. Abstraction
2. Refinement
3. Modularity
4. Software Architecture
5. Control Hierarchy
6. Structural partitioning
7. Data Structure
8. Software procedure
9. Information Hiding

Effective Modular Design

A modular design reduces complexity, facilitates change and results in easier implementation by encouraging parallel development of different parts of a system. An effective modular design features are as follows :

→ Functional independence → Cohesion → Coupling

Data Design

Data design creates a model of data and/or information that is represented at a high level of abstraction. The structure of data has always been an important part of software design.

Architectural Design

Architectural design represents the structure of data and program components that are required to build a computer based system. It considers the architectural style that the system will take, the structure and properties of the components that constitute the system and the interrelationships that occur among all architectural components of a system.

Design Documentation

The design specification addresses different aspects of the design model and is completed as the designer refines his representation of the software.

PREVIOUS YEARS QUESTIONS

PART-A

Q.1 How are coupling and software portability related?

Give an example.

[R.T.U. 2018]

Ans. The concept of coupling embodies the reliability and dependability (strength) among all relationships between functional units. The concept of software portability is the usability of same components/software in different environments. For optimal software, it is generally the case where coupling has to be minimized and portability has to be maximized (for sustainability and effectiveness in the long run). Having said that, strong software that contains low levels of coupling could easily and efficiently be portable. Similarly,

if the software is portable it is probably due to the low levels of coupling that it has.

For example: If you want a working building, then all those nooks and crannies should not be visible (implying some coupling). That means that they should be stuck together pretty tightly (coupling). At the same time, if you want to remove a part of your structure like a room or a floor, then the tightness of your pieces (coupling) shouldn't be as strong. Ultimately, your structures portability is only dependent on the tightness of the pieces.

Q.2 What is a "Design Walk through"? Is it different from a "Design Inspection"?

[R.T.U. 2018]

Ans. Design Walk Through : A design walk through is a quality practice that allows designers to obtain an early validation of design decisions related to the development and

(Software Engineering)
treatment of content, design of the graphical user interface and the elements of product functionality. Design walk throughs provide designers with a way to identify and assess early on whether the proposed design meets the requirements and addresses the project's goal.

Design Inspection different from Design walk

Through:

- Walk through is an informal meeting for evaluation. No preparation is required.
- Inspection is a method that deserves careful consideration by an organization concerned with the quality of the product. It is conducted by quality control members.

Q.3 What are the importance of design documentation in software engineering?

Ans. Importance of Design Documentation

1. Clarify our business goals, requirements and activities : With a proper documentation, we can share the business goals and requirement with our managers and team mates so that they have a clear vision and goals and the activity they perform will be more towards the success.
2. Design and specify our product : This comes in Architectural/Design documents and it gives us complete overview of how our products look like.
3. Everything is clearly explained : When we makes End User documentation of the product of software, we have to explain each and everything about its working. It describes each feature of the program, and assists the user in realizing these features.
4. Any body can work on other's code : If we are a developer, it is not sufficient to write good codes only but we also need to take cares about the documentation part, which can be helpful to other developers while working in a team.
5. Helpful in proper communication : A good software documentation is helpful in proper communication. The written procedure helps we to make interaction within several departments.

Q.4 Write short note on fundamentals of software design.

Ans. Fundamentals of Software Design : Every intellectual discipline is characterized by fundamental concepts and specific techniques. They provide the underlying basis for development and evaluation of techniques. A set of fundamental software design concepts has evolved over the past three decades. Each concept provides the software designer with a base so that he can use good design methods.

SE-47
Fundamental software design concept provide the necessary framework for 'getting it right'. There are various design concepts, which are given below :

1. Decomposition and Modularity
2. Abstraction
3. Stepwise Refinement
4. Software Architecture
5. Control Hierarchy
6. Data Structure
7. Software Procedure
8. Structural Partitioning
9. Information Hiding

PART-B

Q.5 Explain Cohesion and Coupling.

[R.T.U. 2017]

OR
Explain different types of cohesion that a module must exhibit. How is it different from coupling?

[R.T.U. 2018]

OR
Explain cohesion and coupling in modular design.

[R.T.U. 2013]

OR

Distinguish between cohesion and coupling. Explain both cohesion and coupling spectrum.

[R.T.U. 2016, 2011]

OR

Explain effective modular design in context of the following : Cohesion and Coupling ?

[RTU 2010, 2009; Raj. Univ. 2007, 2006, 2005]

Ans. Cohesion : Cohesion is the extension of the information hiding concept. Cohesion refers to the internal "glue" with which a module is constructed. It is the concept that tries to capture the intra-module bond. The intra-module bond refers to strengthen the bond between elements of the same module by maximizing the relationship between them. So this bond tells how closely the elements of a module are related to each other.

Cohesion of a module represents how tightly bonds of the internal elements of a module have made with each other. A module is cohesive if all elements of the module are directed towards and essential for performing the same

task. So, a cohesive module performs a single task within a software procedure, requiring little interaction with procedures being performed in other parts of a program.

Ideally, a cohesive module should do just one thing. In cohesion of a module, the designer must have an idea about whether the different elements of a module belong to the same module. Cohesion and coupling are closely related, usually in practice the greater the cohesion of each module in the system, the lower the coupling between modules. So, the more cohesive a module, the more related are the internal parts of module to each other and its overall purpose.

Fig. 1 shows the example of various types of cohesion.

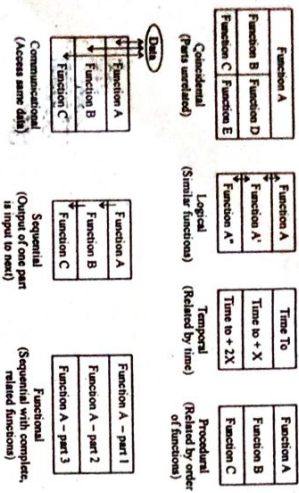


Fig. 1 : Examples of Cohesion

It can be seen that there are total seven levels of cohesion :

- (1) Coincidental Cohesion
- (2) Logical Cohesion
- (3) Temporal Cohesion
- (4) Procedural Cohesion
- (5) Communicational Cohesion
- (6) Sequential Cohesion
- (7) Functional Cohesion

(1) **Coincidental Cohesion** : The worst degree of cohesion is coincidental, which is placed at the lowest level. It occurs when no meaningful relationship exists among the elements of a module. So, a module perform the tasks which are loosely related or almost unrelated to each other. It can occur, if an existing program is "modularized" by breaking it into pieces and making different pieces of module. If a module is created to save duplicate code by combining some part of code which occurs at many different places, that module is likely to have coincidental cohesion. For example, a module that checks a user's security classification and also prints this week's payroll is coincidentally cohesive.

(2) **Logical Cohesion** : Logical cohesion is the next higher level of cohesion, where several logically related elements or data elements are placed in the same component.

For an example, one module may read all kinds of inputs (from tape, disk etc.) regardless of where the input is coming from or how it will be used; input is the glue that holds this module together. Although more reasonable than coincidental one, the elements of a logically cohesive module are not related functionally. In our example, since the input can have different purposes for different modules, we perform many unrelated functions in one place.

(3) **Temporal Cohesion** : Temporal cohesion is same as logical cohesion, except that, here the elements are related with time and executed together. So, the time concern is associated over here. When a module performs tasks that must be executed with same duration of time that module is called a **Logically Cohesive Module**. Elements of the module in this case are time related. Temporal cohesion is higher than logical cohesion because the elements are executed simultaneously.

(4) **Procedural Cohesion** : When elements are grouped together in a module just to follow a sequential order, the component is procedurally cohesive. So, the elements of a module are evaluated with each other and must be executed in a specific order and belong to a common procedural unit. For example, a loop of decision statements in a module may be combined to form a separate module. Procedurally cohesive modules often occur when modular structure is determined from some form of flowchart.

(5) **Communicational Cohesion** : Communicational cohesion has elements that are related by a reference to the same input or output data. Here, the elements concentrate on one area of a data structure. In a communication bound module, the elements are together because they operate on the same input or output. Such modules may perform more than one function. This cohesion is sufficiently high and generally acceptable if alternative structures with higher cohesion cannot be easily identified. However, communicational cohesion often destroys the modularity and functional independence of the design.

(6) **Sequential Cohesion** : If the output from one part of a module is input to the next part, the module has sequential cohesion. So, all the elements are related in such a way in a module so that the output of one forms the input to another. But the sequential cohesion doesn't provide any guidelines on how to combine elements of a module. A sequentially bound module may contain several functions or parts of different functions.

(7) **Functional Cohesion** : Our ideal is the functional cohesion which is the strongest cohesion. Here, all the elements of the module are related to perform a single whole-scope function. In this cohesion every processive element is

essential to the performance of a single function and all essential components are contained in one module. For an eg., to find the factorial of the number, can be considered as a single function to be performed.

Coupling : Coupling between modules is the strength of interconnections between modules or a measure of interdependence among modules. All the modules in a system can not be independent of each other, as they must interact with each other to produce the desired external behaviour of the system. The more connections between modules, the more dependent they are, in the sense, that more knowledge about one module is required to understand or solve the other module.

We must strive for low coupling, so that less interconnections among modules are there. Hence, the coupling is used to see "how strongly" different modules are interconnected. It depends on the interface complexity between modules, the point at which entry or reference is made in a module and what data pass across the reference.

So, we say that two modules are highly coupled when they are joined by strong interconnections and there is a great deal of dependence between them. Loosely coupled modules have some dependence but the interconnections among modules are weak. Uncoupled modules have no interconnections at all, they are completely independent, as shown in fig. 2.

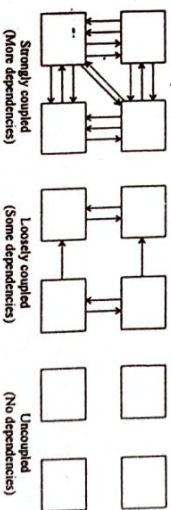


Fig. 2 : Module Coupling

The main goal is to keep the degree of coupling as low as possible. That is, it is required to minimize the dependence among modules. There are many ways that modules can be dependent on each other, so coupling depends on several things :

- The references made from one module to another.
- The amount of data passed from one module to another.
- The amount of control one module has over the other.
- The degree of complexity in the interface between modules.

Thus, we can measure coupling along a range of dependence, from complete dependence to complete independence. There are total seven types of couplings, given below and shown in fig. 3:

- (1) No Direct Coupling
- (2) Data Coupling
- (3) Stamp Coupling
- (4) Control Coupling
- (5) External Coupling
- (6) Common Coupling
- (7) Content Coupling

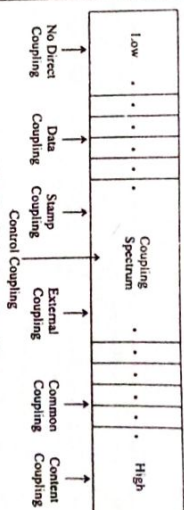


Fig. 3 : Cohesion Coupling

(1) **No Direct Coupling** : No direct coupling refers to the modules in which there is no connection in between. So modules are unrelated with each other and have no interconnections. Such modules are called as the Independent Modules. No direct coupling is shown in fig. 4.

There is no direct coupling between the module 1 and module 2 and they perform their functions independent to each other and are subordinates of module 3.

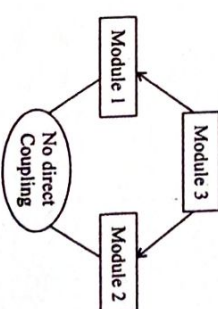


Fig. 4 : No Direct Coupling

(2) **Data Coupling** : Data coupling exists when data are passed from one module to another module via argument list. So, simple data are passed and a one-to-one correspondence of items exists. As such, data coupling is simpler and does not make much errors. The low coupling is exhibited in fig. 5.

In the fig. 5, the data coupling exists between module 1 and module 2 where some data are passed as an arguments from module 1 to module 2. So module 1 requires module 2 to be performed.

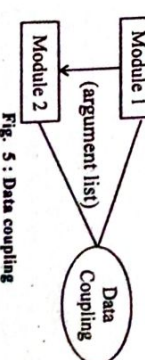


Fig. 5 : Data coupling

(3) **Stamp Coupling** : Stamp coupling is a variation of the data coupling. It is used when a data structure is used to pass information from one module to another. Stamp coupling is shown in fig. 6.

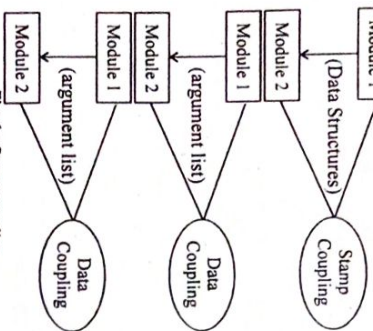


Fig. 6 : Stamp coupling

In fig. 4 module 1 is sending some data structure as an argument to the module 2.

(4) **Control Coupling** : When one module passes parameters to control the activity of another module, it is said that there is control coupling between the two. In fig. 7 control coupling is shown, it occurs when module 1 passes control data to module 2. Then the module 2 will perform its complete function. So, control is passed via a "flag" on which decisions are made from a super ordinate module to a subordinate module.

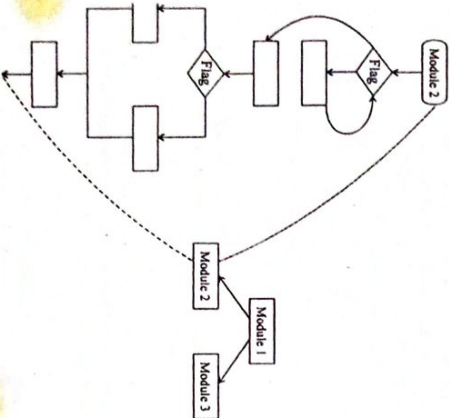


Fig. 7 : Control Coupling

In this coupling, it is impossible for the controlled module to function without direction from the controlling one. In a design with control coupling, there is an advantage that each module perform only one function or execute one process. This restriction minimizes the amount of controlling information that must be passed from one module to another and localizes control to a fixed and recognizable set of parameters forming well-defined interface.

(5) **External Coupling** : External coupling is the coupling in which high coupling occurs. Modules are tied to an environment external to software. It is essential but should be limited to small number of modules within a structure.

(6) **Common Coupling** : In common coupling, to reduce the amount of coupling, data is accessible from a common data store (global data area). However, dependence still exists, since making a change to the common data means tracking back to all modules that access that data to evaluate the effect of that change. This kind of dependence is called Common Coupling.

With common coupling, it can be difficult to determine which component is responsible to set a variable to a particular value. In the fig. 8 it is shown how common coupling works.

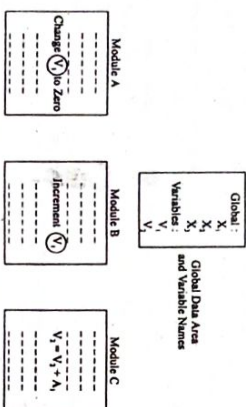


Fig. 8 : Common Coupling

As shown in the fig. 8 modules make use of data and variable names from a global data area.

(7) **Content Coupling** : Content coupling is the least desirable coupling.

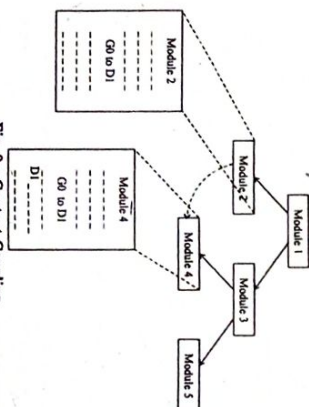


Fig. 9 : Content Coupling

This coupling occurs when one module makes use of data or control information maintained within the boundary of another module. That is, when one module actually modifies another module, then the modified module is completely dependent on the modifying one. And this is called as the Content Coupling.

Content coupling might occur when one module modifies an internal data item in another module or when one module branches into the middle of another module. Fig. 9 shows an example of content coupling.

As it can be seen in the fig. 9 module 2 branches into module 4, even though module 4 is supposed to be under the control of module 3.

We should strive to achieve low coupling. As such, it is an abstract concept that is not easily quantifiable. So, no formulas can be given to determine the coupling between two modules.

Q.6 What is design documentation in software engineering? Explain along with its importance in detail. [R.T.U. 2017]

Ans. Documentation :

- An important product of the design processors is a set of documents that describe the system to be built.
- The design documents should contain a section, called the design rationale, outlining the critical issues and trade off that were considered in generating the design.

Software design document outline

- Reference documents**
 - Existing software documentation
 - System documentation
 - Vendor (hardware or software) documents
 - Technical reference
- Modules for each module**
 - Processing narrative
 - Interface description
 - Design language (or other) description
 - Modules used
- Scope**
 - System objective
 - Hardware, software and human interfaces
 - Major software functions
 - Externally defined database
 - Major design constraints, limitations
- Design description**
 - Data description
 - Review of data flow
 - Review of data structure
 - Derived program structure
 - Interfaces within structure

- Test provisions**
 - Test guidelines
 - Integration strategy
 - Special considerations
 - Packaging**
 - Special program overlay provisions
 - Transfer considerations
 - File structure and global data**
 - External files structure :
 - Logical Record Description
 - Access method
 - Global data
 - File and data cross-reference
 - Requirements cross-reference**
- Importance of Design Documentation : Refer to Q.3.**

Q.7 Describe the top down and bottom up approach in effective modular design with example. [R.T.U. 2017]

[R.T.U. 2017]

Ans. Top Down Approach : In top down strategy we start by testing the top of the hierarchy and we incrementally add modules that it calls and then test the new combined system. This approach of testing requires stubs to be written. A stub is a dummy routine that simulates a module.

In the top-down approach, a module cannot be tested in isolation because they invoke some other modules. To allow the modules to be tested before their subordinates have been coded, stubs simulate the behavior of the subordinates.

Bottom-up Approach : The bottom-up approach starts from the bottom of the hierarchy. First the modules at the very bottom, which have no subordinates, are tested. Then these modules are combined with higher-level modules for testing. At any stage of testing all the subordinate modules exist and have been tested earlier.

To perform bottom-up testing, drivers are needed to set up the appropriate environment and invoke the module. It is the job of the driver to invoke the module under testing with the different set of test cases.

Example to understand the working of top down and bottom up approach:

Top-down vs. Bottom-up object database design

There are two approaches for developing any database, the top-down method and the bottom-up method. While these approaches appear radically different, they share the common goal of uniting a system by describing all of the interaction between the processes. Let's examine each approach:

The top-down method starts from the general and moves to the specific. Basically, we start with a general idea of what is needed for the system and then ask the end-users

what data they need to store. The analyst will then work with the users to determine what data should be kept in the database. Using the top-down method requires that the analyst has a detailed understanding of the system. The top-down method also can have shortcomings. In some cases, top-down design can lead to unsatisfactory results because the analyst and end-users can miss something that is important and is necessary for the system.

The bottom-up approach begins with the specific details and moves up to the general. To begin a bottom-up design, the system analyst will inspect all the interfaces that the system has, checking reports, screens, and forms. The analyst will work backwards through the system to determine what data should be stored in the database.

To understand the differences between these approaches, let's consider some jobs that are bottom-up in nature. In statistical analysis, analysts are taught to take a sample from a small population and then infer the results to the overall population. Physicians are also trained in the bottom-up approach. Doctors examine specific symptoms and then infer the general disease that causes the symptoms.

An example of jobs that require the top-down approach include project management and engineering tasks where the overall requirements must be specified before the detail can be understood. For example, an automobile manufacturer must follow a top-down approach to meet the overall specifications for the car. If a car has the requirement that it cost less than 15,000 dollars, gets 25 miles per gallon, and seating five people. In order to meet these requirements the designers must start by creating a specification document and then drilling down to meet these requirements.

The analyst will have no choice but to talk and work with the users to determine what is important to the users and as a result determines what data should be stored in the database. What the analyst usually does is create some prototype reports, screens, and forms to help the users visualize what the system will look like and how the system will work.

Q.8 *Explain the programming style and program quality.* [R.T.U. 2015]

Ans. Software coding is a process of implementing the software in terms of code. As the complexity of the software increases, the lines of code also increases, hence size of software increases. It becomes very difficult to understand and change the software, so there is a need to bring clarity in code and improve the efficiency and quality of code with reduction in time and cost associated with software

Code with bad programming style	Code with good programming style
<pre>void main () { int a, b, c, d; printf("Enter two no."); scanf("%d %d", &a, &b); if (b = 0) printf("division is : %d", a/b); }</pre>	<pre>void main () { int number 1, number 2, div; printf("Enter two integers"); scanf("%d %d", &a, &b); if (b = 0) // denominator // should not be zero printf("division is : %d", a/b); }</pre>
Bad program quality	Good program quality

Q.9 Explain the design documentation with example

[R.T.U. 2015]

OR
What do you mean by design documentation?
Explain in brief.

OR

Write short note on "Design Documentation".

[Raj. Univ. 2004]

Ans. Design Documentation : Refer to Q.6.

Q.10 What is software design? How will you translate the analysis model into design model? Also explain the design principles in brief.

[RTU 2013]

Ans. A software decomposition into modules, description of what each module is intended to do and of the relationship among the modules.

Such type of description is called as the Software Architecture or the Software Structure. Thus, the goal of the design activity is the definition of the software architecture.

Development of Software Design Model

The design model is developed by the software designer only whose main goal is to develop a model or abstraction of the product based on experience, design principles, its guidelines, heuristics and fundamental concepts. This model is then evaluated with respect to the requirements (or the analysis model), and operating and economic environments. The design process goes through some stepwise refinements until a physical realization comes into existence.

The fig. shows, how the analysis model developed during analysis phase is translated into the design model. The design model consists of four design models (elements) in the form of a pyramid. All design model are developed by using the elements of the analysis model as given below:

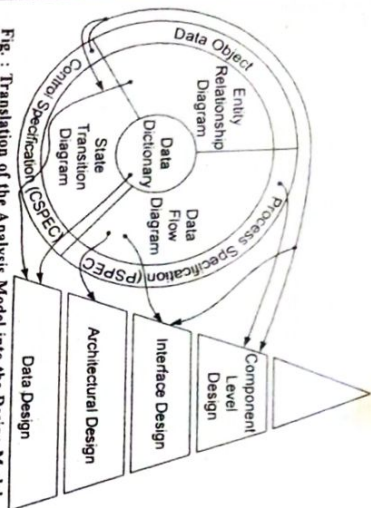


Fig. : Translation of the Analysis Model into the Design Model

The data objects and their relationship obtained by entity-relationship diagrams and the data details from the data dictionary made during the analysis phase provides the ground for the creation of the data model of a software. This information serves as the base for the creation of data structures of the software solution.

- **Architectural design** consists of data flow diagrams. It depicts the relationships between major "structural elements" of the software, various design patterns and various design pattern constraints.

- **Interface design** requires control specifications, state transition diagrams and data flow diagrams. It depicts the inter-communication path of the software i.e. it describes this user-computer interaction, the communication with any other required system or within itself.

- **Component level design** involves the control specification, state transition diagram and process specification.

It converts the structural components of the software architecture into a procedural description of the software components.

In this manner, the requirements through analysis model can be translated into the software design. This model serves as the base of the coding phase. So, design is the only tool that depicts the transformations of customer's requirements into finished software product. Design also describes the quality of the software work product, which goes throughout the life cycle phases—coding, testing and maintenance.

Characteristics of A Good Design

There are some characteristics which a good design must have to achieve good quality products. McGlashan has suggested three characteristics of a good design :

Q.11

B.Tech. (III Sem) C.S. Solved Papers

The design must implement all of the explicit requirements contained in the analysis model, and it must accommodate all of the implicit requirements desired by the user.

The design must be a readable, understandable guide those who generate code and for those who test and frequently support software.

The design should provide a complete picture of the software addressing the data, functional and behavioral domains from an implementation perspective.

Design Principles

The design process should not suffer from "tunnel vision." A good designer should consider alternative approaches, judging each based on the requirements of the problem, the resources available to do the job.

The design should be traceable to the analysis model. Because a single element of the design model often traces to multiple requirements, it is necessary to have a means for tracking how requirements have been satisfied by the design model.

The design should not reinvent the wheel. Systems are constructed using a set of design patterns, many of which have likely been encountered before. These patterns should always be chosen as an alternative to reinvention. Time is short and resources are limited. Design time should be invested in representing truly new ideas and integrating those patterns that already exist.

The design should "minimize the intellectual distance" between the software and the problem as it exists in the real world. That is, the structure of the software design should (whenever possible) mimic the structure of the problem domain.

The design should exhibit uniformity and integration. A design is uniform if it appears that one person has developed the entire thing. Rules of style and format should be defined for a design team before design work begins. A design is integrated if care is taken in defining interfaces between design components.

The design should be structured to accommodate change. The design concepts discussed in the next section enable a design to achieve this principle.

The design should be structured to degrade gently, even when aberrant data, events, or operating conditions are encountered. Well-designed software should never "bomb." It should be designed to accommodate unusual circumstances, and if the processing needs to be terminated, do so in a graceful manner.

Design is not coding, coding is not design. Even when detailed procedural designs are created for program components, the level of abstraction of the design model

is higher than source code. The only design decisions made at the coding level address the small implementation details that enable the procedural design to be coded.

- The design should be assessed for quality as it is being created, not after the fact. A variety of design concepts and design measures are available to assist the designer in assessing quality.

- The design should be reviewed to minimize conceptual (semantic) errors. There is sometimes a tendency to focus on minutiae when the design is reviewed, missing the forest for the trees. A design team should ensure that major conceptual elements of the design (omissions, ambiguity, inconsistency) have been addressed before worrying about the syntax of the design model.

Q.11 Explain modularity in context of software design.

[RTU 2013]

Ans. **Modularity** : "Modularity can be stated as-

"Modularity is one single attribute of software that allows a program to be intellectually manageable". The main benefit of modularity is that it allows the principle of separation of concerns to be applied in two phases : when dealing with the details of each module in isolation (while ignoring details of other modules) and when dealing with the overall characteristics of all modules and their relationships in order to integrate them to make it as a whole system. The systems which follow the concept of modularity are called as the Modular Systems. Modular Systems consist of well-defined, manageable unit with well-defined interfaces among the units. Desirable properties of a modular system include :

- (1) Each processing abstraction is a well defined subsystem that is potentially useful in other applications.
- (2) Each functions in each abstraction has a single, well-defined purpose.
- (3) Each function manipulates not more than one major data structure.
- (4) Functions share global data selectively, it is easy to identify all routines that share a major data structure.
- (5) Functions that manipulate instances of abstract data types are encapsulated with the data structure being manipulated.

Need of Modularity : Modularity is required to build the system. We consider two problems, P_1 and P_2 , a complexity function $C(x)$ and an effort function $E(x)$ to solve a problem P . As in a general case, if the complexity of problem P is more, then the required effort will be more for it, stated as :

4. Carefully demonstrating that each successive step in the refinement process is a faithful expansion of previous steps.

So, refinement defines the internal functioning details. The refinement process may be started during requirement analysis and completed when the detail of design is sufficient for conversion into code. Refinement is actually a process of elaboration.

3. **Information Hiding** : Information hiding permits modules to be designed and coded without concern for the internal details of other modules. Paras suggest the components hide the internal details and processing from one another. In information hiding, each component hides a design decision from others.

Architectural Styles and Strategies

Software architectural design is next to be followed after data design. It refers to a pattern for construction. A style involves its components (modules), connectors and constraints to combine components. The software designer (or architect) usually follows the top-down manner. There are many architectural styles from which any architectural style can be used to build the software (or system).

Various Styles

Shaw and Garlan has suggested seven commonly used styles:

1. Pipes and Filters
2. Object-oriented Architectures
3. Implicit Invocation
4. Layering
5. Repositories
6. Interpreters
7. Process Control

Two architectural styles of software design given below :

1. Pipes and Filters

A component in a pipe-and-filter system, as shown in Fig. 1, has streams of data called pipes, for input and output. Often, the transformation of the data from input to output is begun before the component, called a filter, finishes reading the input stream. In this type of system, each filter works independently of those components of upstreams and downstreams. Each filter doesn't require knowledge of the working of its neighboring filters. Additionally, the correctness of the system's output doesn't depend on the order in which the filters are applied.

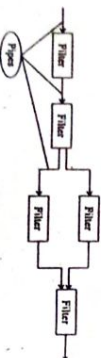


Fig. 1: Pipes and Filters

There are several properties of pipe-and-filter system:

- The designers can understand the entire system's effects on input and output as the composition of the filters.
- Since any two filters can be linked together, the filters can be reused easily on other systems.
- System evolution is simple as new filters can be added and old filters can be removed easily.
- Because of filter independence, designers can simulate system behavior and analyze properties such as throughput.
- They allow concurrent execution of filters.
- Drawbacks of this Architecture
 1. They encourage batch processing and they are not good for handling interactive applications.
 2. When two data streams are related, the system must maintain a correspondence between them.

2. Layering

In the layered architecture, the number of layers are defined, as shown in Fig. 2.



Fig. 2: Layered Security Architecture

In the fig, the layers are hierarchical; each layer provides services to the one outside it, and acts as a layer client to the inside it. In some systems, every layer has access to some or all the other layers; whereas in other systems, a given layer has access only to its adjacent layers. The description of various layers is given below:

- (i) **Cryptography Layer** is the inner layer, includes functions to encrypt and decrypt a key that is used in the basic encryption scheme for the system.
- (ii) **File Interface Layer** encrypts and decrypts a file.
- (iii) **Key Management Layer** allows a component to "sign" for file, verifies the signature and computes a hash code to gain access to the file.
- (iv) **Authentication Layer** manages a password file that is stored in encrypted form and it requests users to provide identification and a password.

Advantages of Layering

1. Layered architectures use the notation of abstraction. So, each layer can be considered to be an increasing level of abstraction and the designers can use the layers to decompose a problem into a sequence of abstract steps.
2. It is relatively easy to add or modify a layer as the need arises.

Disadvantages of Layering

1. It is not always easy to structure a system in layers.
2. The system performance degrades due to the extra coordination among layers.

Procedural Design

Procedural design occurs after data and program structure have been established. In an ideal world, the procedural specification required to define algorithmic details could be stated in a natural language such as English. Procedural design must specify procedural detail which must be clear, understandable and unambiguous. Using a natural language, we can write a set of procedural steps in so different ways.

Structured Programming

In 1960, Dijkstra and others proposed the use of a set of simple logical constructs from which any program could be formed. The constructs are sequence, condition and repetition. These three constructs are fundamentals to structured programming.

Graphical Design Notation

There are two graphical design notations to be used—Flow Chart and Box Diagram, that readily depict procedural details. If graphical tools are misused, the wrong picture may lead to the wrong software.

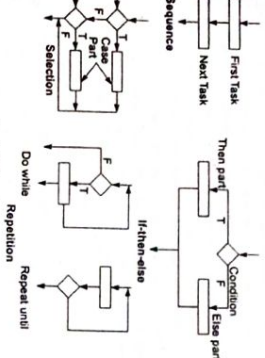


Fig. 3: Flowchart

The flowchart is the most widely used graphical representation for procedural design. The flowchart is quite simply pictorial. A box is used to indicate a processing step. A diamond represents a logical condition and arrows show the flow of control. Sequence is represented as two processing boxes connected by a line (arrow) of control. Condition, also called if-then-else, is depicted as a decision diamond which if true causes then-part processing to occur, and if false, invokes else-part processing. Repetition is represented using two slightly different forms.

Tabular Design Notation

Decision tables provide a notation that translates action and conditions into a tabular form.

Rule Numbers	Decision Table									
	1	2	3	4	5	6	7	8	9	10
Condition										
Rows										
Action										

Fig. 4: Decision Table Representation

The table is difficult to misinterpret and may even be used as a machine readable input to a table-driven algorithm. Referring to the Fig. 4, double lines divide the table into four sections. The upper left-hand section contains a list of all conditions whereas lower left-hand section contains a list of all possible actions on combinations of conditions. The right-hand sections from a matrix to indicate condition combinations and the corresponding actions that will occur for a specific combination. Therefore, each column of the matrix may be interpreted as a processing rule.

The following steps are applied to develop a decision table:

- (1) List all actions that can be associated with a specific procedure (or module).
- (2) List all conditions (or decisions made) during execution of procedure.
- (3) Associate specific sets of conditions with specific actions, eliminating impossible combinations of conditions; alternatively, develop every possible permutation of conditions.
- (4) Define rules by indicating what actions occur for a set of conditions.

Program Design Language

Structured English or pseudocode, PDL, is used as a generic reference for a design language. The difference between PDL and a real high-order programming language lies in the use of narrative text (example English) embedded directly into a syntactical structure. PDL cannot be compiled (at least not yet). However, PDL "Processors" currently exist to translate PDL into a graphical representation (example, a flow chart) of design, and produce nesting maps, a design operation index, cross reference tables and a variety of other information. A design language should have the following characteristics:

- (a) A fixed syntax of keywords that provide for all structural constructs, data declaration, and modularity characteristics.
- (b) A free syntax of natural language that describe processing features.

- (c) Data declaration facilities that should include both simple (scalar, array) and complex (linked list or tree) data structures.
- (d) Subprogram definition and calling techniques that support various modes of interface description.

Flowchart : A flowchart is made to break down a process into simple steps by writing down the steps in boxes that are connected with arrows. A flow chart starts from the first step and ends at the last step with all the actions to be performed in middle. Flow chart also trouble shoots the problems by providing solution if an error occurs during any step. The biggest advantage of a flow chart is that it gives an overall view of the process at a single glance, to understand it better. There are different types of flowcharts like

- (i) System flow chart
- (ii) Data flow chart
- (iii) Document flow chart
- (iv) Program flow chart

Box Diagram : A box diagram is a representation of data flow through a system during which it is processed as well. The flow of data from an external source or internal source to its destination is shown by a diagram. Where the data will end up after being processed is also shown in a box diagram. The processes through which data will go through are shown in these diagrams. These processes can be a sequence or act simultaneously while data is going through the system.

Flowchart versus Box Diagram in Procedural design:

- (1) The main difference between flow chart and box diagram is that flow chart presents steps to complete a process whereas Box diagram presents the flow of data.
- (2) Flow chart does not have any input from or output to external source whereas box diagram describes the path of data from external source to internal store or vice versa.
- (3) The timing and sequence of the process is aptly shown by a flow chart whereas the processing of data is taking place in a particular order or several processes are taking simultaneously is not described by a box diagram.
- (4) Box diagrams define the functionality of a system where as flow diagram shows how to make a system function.
- (5) Flow charts are used in designing a process but box diagram are used to describe the path of data that will complete that process.

Q.14 Explain problem partitioning and abstraction as the major aspects of design fundamentals. [K.T.U. 2016]

OR

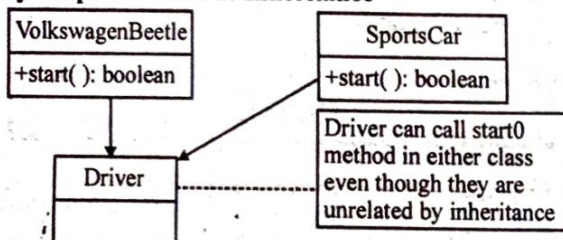
PREVIOUS YEARS QUESTIONS

PART-A

Q.1 "Can we have inheritance without polymorphism?"
Comment on it. [R.T.U. 2018]

Ans. Inheritance and polymorphism are independent but related entities – it is possible to have one without the other. If we use a language that requires variables to have a specific type (C++, C#, Java) then we might believe that these concepts are linked.

Polymorphism without Inheritance



There are languages where you have polymorphism without using inheritance. Some examples are JavaScript, Python, Ruby, VB.NET, and Small Talk.

In each of these languages it is possible to write `car.start()` without knowing anything about the object `car` and its method.

Python polymorphism

```

class VolkswagenBeetle(Car):
def start(): # Code to start Volkswagen
class SportsCar(Car):
def start(): # Code to start SportsCar
# Invocation of polymorphism
cars = [ VolkswagenBeetle(), SportsCar() ]
for car in cars:
car.start()
    
```

The ability to get pure polymorphism stems from these languages only have a single variable type prior to runtime i.e. `var` in JavaScript, `def` in Python, `dim` in VB.NET. With only one variable type there cannot be type error prior to runtime.

Q.2 List object oriented design approaches. [R.T.U. 2018]

Ans. Object Oriented Design : Object oriented design works around the entities and their characteristics instead of

functions involved in the software system. This design strategies focuses on entities and its characteristics. The whole concept of software solution revolves around the engaged entities.

Let us see the important concepts of object oriented design:

- (1) **Objects** : All entities involved in the solution design are known as objects. For example, person, banks, company and customers are treated as objects. Every entity has some attributes associated to it and has some methods to perform on the attributes.
- (2) **Classes** : A class is a generalized description of an object. An object is an instance of a class. Class defines all the attributes, which an object can have and methods, which defines the functionality of the object.

In the solution design, attributes are stored as variables and functionalities are defined by means of methods or procedures.

- (1) **Encapsulation** : In OOD, the attributes (data variables) and methods (operation on the data) are bundled together is called encapsulation. Encapsulation not only bundles important information of an object together, but also restricts access of the data and methods from the outside world. This is called information hiding.
- (2) **Inheritance** : OOD allows similar classes to stack up in hierarchical manner where the lower or sub-classes can import, implement and re-use allowed variables and methods from their immediate super classes. This property of OOD is known as inheritance. This makes it easier to define specific class and to create generalized classes from specific ones.
- (3) **Polymorphism** : OOD languages provide a mechanism where methods performing similar tasks but vary in arguments, can be assigned same name. This is called polymorphism, which allows a single interface performing tasks for different types. Depending upon how the function is invoked, respective portion of the code gets executed.

Q.3 What is sequence diagram in context of UML?

Ans. Sequence Diagram : Sequence diagram is an interaction diagram that models dynamic aspect of a system. It captures flow of messages from one object to other.

Components of Sequence Diagram

- Objects
- Messages between objects
- Message synchronization

The above figure shows that customers first order the meal by choosing from menu and then placing orders. Further the customer pays the bill and gets the receipts.

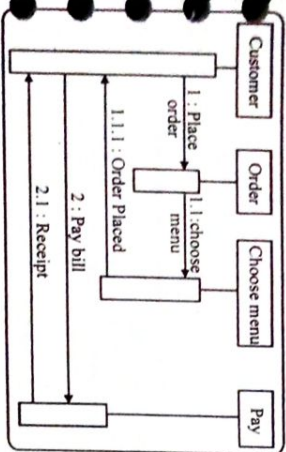


Fig.

2.4 Define class and object in context of UML.

Ans. Classes and Objects : Class is an UML representation which has basic three section.

- (1) Class name
- (2) Class attributes
- (3) Class functions

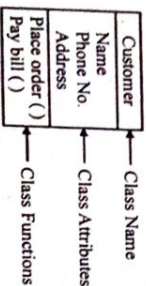


Fig.

Object is an UML representation that is used to represent objects. It has two sections.

- (1) Object name
- (2) Class name

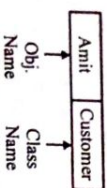


Fig.

Amit is one of the customers.

1.5 Write short note on state diagram.

Ans. State Diagrams : State diagram models the dynamic behavior of a system. It describes the various states and the events that changes the state. So, it controls the flow of events from one state to the other.

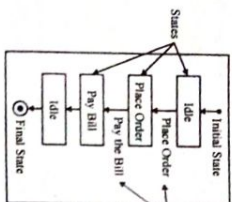


Fig.

Components of state Diagram

- (1) States
- (2) Events triggering to a state.

The above figure shows three states idle, place order and pay bill. Place order is the event that triggers from idle to place order state where if pay the bill event triggers then it leads to pay Bill state.

PART-B

Q.6 Define Unified Approach? Explain UML with all its diagram and advantages.

[R.T.U. 2016]

What is UML? Explain how it is useful in object oriented modeling.

[R.T.U. 2018]

Explain unified modelling language diagrams with the help of appropriate example.

[R.T.U. 2017]

What is UML? How it is useful in OO modelling?

[R.T.U. 2013]

What is UML?

[R.T.U. 2014]

Draw and explain class diagram, object diagram and process diagram. List all the diagram available in UML.

[R.T.U. 2011]

Explain different types of diagrams proposed by UML.

[R.T.U. 2010]

What is Unified Modelling Language? Explain all the diagrams of UML?

[R.T.U. 2009, 2008, Raj, Univ. 2007]

Ans. Unified Approach : A unified approach evolved from the combined efforts of Booch, Rumbaugh and Jacobson. This unified approach is known as UML.

UML is organized into two major design activities System design and Object design.

Modelling is simplification of reality. It is a central part of all the activities that leads to the deployment of good software. We build models to communicate the desired structure and behaviour of the system. In short, building models help in :

- (a) Visualization and control of system's architecture.
- (b) Exposure of opportunities for simplification and reuse.
- (c) Managing risk.
- (d) Communication for the desired structure and behaviour of the system.

So, modelling is very important to be done. In the concern of complexity of a program, the modelling plays an important and effective role to reduce it. As we know that when the levels of abstraction increases, there are more chances of failure and it will be very difficult to solve the problem.

The Unified Modelling Language (UML) is a standard language for writing software blueprints. UML may be used to visualize, specify, construct and document the artifacts of a software-intensive system.

Conceptual Model of UML

UML requires three major elements to build its conceptual model. It includes :

- UML's basic building blocks,
- Rules that dictate how those building blocks may be put together and
- Some common mechanisms that apply throughout UML.

One more term required to build UML is view. A view is simply a subset of UML. Modelling construct, that represents one aspect of the system. One or two kinds of diagrams provide a visual notation for the concepts in each view.

UML can be used for domains like :

- Enterprise Information Systems
- Banking and Financial Services
- Telecommunications
- Transportation Management
- Defense/Aerospace and Navigation
- Scientific and Engineering Applications

Building Blocks in UML

To build the blocks in UML, for an application, things are defined and the relationships among them are defined. At last complete UML diagram is made.

(A) Things : Things are the abstractions that are present as an entity in a model. There are four kinds of things in UML

1. **Structural Things :** Structural things are the nouns of UML models. They are mostly the static parts of a model. There are seven kinds of structural things :

- Class
- Interface

- Collaboration
- Use-Case
- Active Class
- Component
- Node

2. **Behavioural Things :** Behavioural things are the dynamic parts of UML models. These are the verbs of a model, representing behaviour over time and space. There are two kinds of behavioural things :

- **Interaction :** Interaction is a behaviour that comprises a set of messages exchanged among a specific objects within a particular context to accomplish a specific purpose.
- **State machine :** A state machine shows behaviour that specifies the sequence of states of an object.

3. **Grouping Things :** Grouping things are the organizational parts of UML models. These are the boxes into which a model can be decomposed. There is only one kind of grouping things i.e. package. Package is a mechanism for organizing elements into groups.

4. **Annotational Things :** Annotational things are the explanatory parts of UML models. These are the comments applied to describe, illuminate and remark about any element in a model. It should be noted that it is simply a symbol for rendering constraints and comments attached to an element or a collection of elements.

(B) **Relationships :** There are basically four kinds of relationships in UML. They are described below one-by-one.

1. **Dependency :** Dependency is a semantic relationship between two things in which a change to one thing (the independent thing) may affect the semantics of the other thing (the dependent thing). It is shown in fig. 1 as a dotted line with an arrow.

Fig. 1 : Dependency

2. **Association :** Association is a structural relationship that describes a set of links, a link being a connection among objects. It is a solid line, possibly directed and may contain label. It may contain multiplicity and role names as shown in fig. 2



Fig. 2 : Association

3. **Generalization :** Generalization is a relationship in which objects of the element that is specialized (may be considered as child) are substitutable for objects of the generalized element (parent). So, a child is sharing the behaviour and structure of parent. It is represented as shown in fig. 3. A hollow arrowhead is pointing to the parent.

Fig. 3 : Generalizations

4. **Realization :** Realization is a semantic relationship between classifiers. One classifier specifies a contract that

another classifier guarantees to carry out. Its representation is shown in fig.4

Fig. 4 : Realization

UML Diagrams

Diagrams are the graphical representations of a set of elements. They are represented as a connected graph of vertices (which are things) and arcs (relationships).

So, a diagram is a projection into a system. The same element may appear in all diagrams, in some of the diagrams or in no diagrams at all. A diagram may contain any combination of things and relationships. So, these UML diagrams are the means by which the building blocks can be viewed and these blocks are most often rendered as a connected graph (things) and arcs (relationships).

So UML can be useful in OO modelling as:

- UML defines a number of diagrams so that we can focus on different aspects of the system independently.
- We can have several hundred classes in the design of a corporate enterprise information system. We can never visualize the structure or behaviour of that system by staring at one large diagram containing all the classes and their relationships.

So, if we want to create several diagrams each focused on one view, that's why, different kinds of diagrams are made available in UML to:

- achieve any combination of elements in the same diagram. For example, to depict both classes and objects in the same diagram.
- decide which views we need to best express the architecture of the system and expose the technical risks to the project if any.

The UML includes nine such diagrams:

- Class Diagrams
- Object Diagrams
- Use Case Diagrams
- Sequence Diagrams
- Collaboration Diagrams
- State Chart Diagrams
- Activity Diagrams
- Component Diagrams
- Deployment Diagrams

1. Class Diagrams

All these diagrams explained as follows:

Class Diagrams : Class diagrams show a set of classes, interfaces, collaborations and their relationships. Class diagrams are specifically used to model the static design view of a system.

Consider an example of house building. During its building, we include basic building blocks such as wall, floors, doors etc. These are inherently structural in nature as they are related with height, width etc. But they are somewhat behavioural also (different kinds of walls can support different loads). We can't consider these features independently. The process of house construction involves assembling these things.

So, with UML, we use class diagrams to visualize the static aspect of the building blocks and their relationships and to specify their details.

The various contents of a class diagram are:

- Interfaces
- Collaborations
- Relationships

The class can be better understood with the help of examples.

Example-1 : In fig.5 a class diagram is shown for the withdrawal of money from the account.

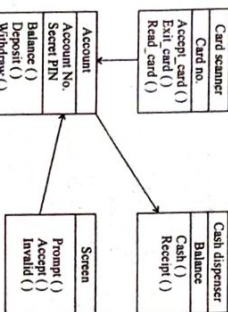


Fig. 5 : Class Diagram for Withdrawal of Money

Example-2 : This example shows the class diagram of an organization, in which department, office, headquarters etc. are shown as things. Its diagram is shown in fig.6

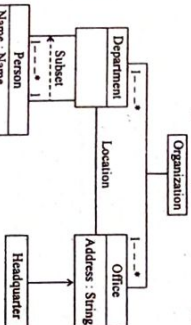


Fig. 6 : Class Diagram for an Organization

2. Object Diagrams

Object diagrams models the instances of things contained in a class diagram, shows set of objects and their relationships at any time and. Modelling a snapshot of system at that moment in time and rendering a set of objects, their state and their relationships.

To understand how the object diagram works, we can visualize a game of soccer. Its overview looks as if it is a

terribly simple sport with a mob of people running wildly chasing a ball. There hardly seems any systematic sense in it.

Now, for a moment, freeze the motion. Classify the individual players. We are able to distinguish their particular positions like forwards, modified, left wing, full back, right back, defenders, goal keepers etc. Further insight reveals how these players collaborate following strategies for attacking and defending. Similar to this, we ought to have the snapshot of the system under concern to look at the objects present, their neighbours and their relationships to these neighbours.

Contents of Object Diagram

There are two contents of an object diagram. They are:

- Links
- Objects

An object diagram expresses the static part of an interaction, consisting of the objects that collaborate but without any of the messages passed among them.

Example-3 : This example tells the object diagram for a company as illustrated in fig.7

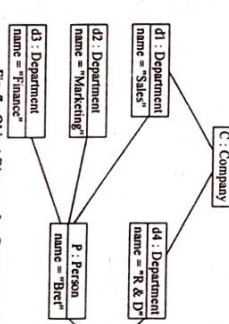


Fig. 7 : Object Diagram for Company

Example-4 : Another example is the objects involved in the mechanism used by the robot to calculate a module of the world in which it moves. Element represents instances that the robot has identified but not yet assigned in its world view.

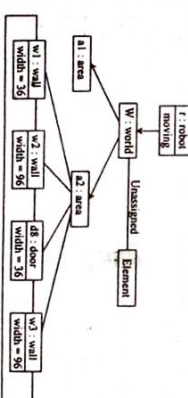


Fig. 8 : Object Diagram Showing Exploration of Robot

Robot has recognized the enclosed area which has walls on 3 sides and a door on fourth side.

3. Use Case Diagrams : Various aspects related to use case diagrams are given below :

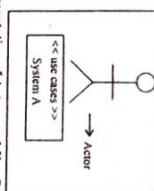
Use case diagrams model the dynamic aspects of a system.

A use case diagram depicts:

1. Use cases : Use cases are the description of a set of sequences of actions, that a system performs yielding an observable result of value to an actor.
2. Actors : Actors are the set of roles that users of use cases play when interacting with the use cases.

Use case diagrams help in understanding how the elements of the system may be used in the context. This enables the user to comprehend how to use that element effectively.

Use case diagrams are represented as shown in fig.9



Use cases are represented by ovals. Labels with ovals represent the functions of the system.

Representation of Use Case is shown in fig.10



Fig. 10 : Representation of Use Case

Actors are the users of a system. They represent who will access the functions of the system. In fig.11 an actor is shown who exercise the use cases.

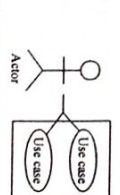


Fig. 11 : Actors Exercising the Use Cases

Relationship between an actor and a use case is illustrated with a signal line. Relationships between use cases are labeled either "uses" or "extends" "uses" indicates that one use case is needed by another in order to perform a task.

Use case diagrams model the use case view of a system. This view supports the behaviour of a system. This represents the visible services that the system provides in context of its environment. Use case diagrams are applied in the following ways:

- Modelling the context of a system : It involves defining a system and asserting which actors the outside the system and interact with it.

Modelling the requirements of a system : It involves specifying what that system should do.

Example-5 : Consider, to model behaviour of a cellular phone. The actors are cellular network and user. And the use cases are making a call, sending message and capturing photos. The use case diagram of cellular telephone is shown in fig.12

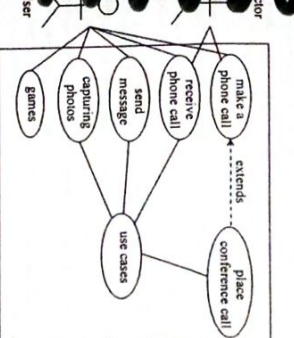


Fig. 12 : Cellular Telephone - Use Case Diagram

Example-6 : Consider an example to process a credit card, using the use cases. It is illustrated in fig.13

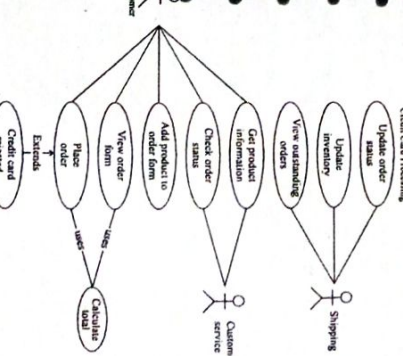


Fig. 13 : Use Case Diagram for Credit Card Processing

- Identify constraints
- Specify optimization criteria

Q.10 Explain the following in context of UML

- (i) Use case diagrams.
- (ii) Sequence diagram
- (iii) Classes and objects.
- (iv) Interfaces.
- (v) State diagrams.

[RTU 2014]

Ans. (i) Use Case Diagrams : Refer to Q.6.

(ii) Sequence Diagram : Refer to Q.3.

(iii) Classes and Objects : Refer to Q.4.

(iv) Interfaces : Interface is an UML representation that is denoted by a circle and name of interface is written below the circle.

Interface describes functionality without implementation. Implementation of an interface is done by the class through the function according to the requirement.

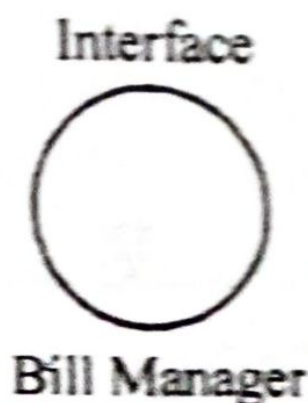


Fig.

(v) State Diagrams : Refer to Q.5.

administrator wants to get maximum possible output from its machines and adjusters. The objective of the simulation is to see how average machine and adjuster utilization depends on:

- Number of Machines
- Number of Adjusters
- Reliability of Machines in terms of MTTF
- Productivity of Adjusters

[R.T.U. 2018]

Ans. (a) Difference Functional-Oriented Design versus Object-Oriented Design

S. No.	Functional-Oriented Design (FOD)	Object-Oriented Design (OOD)
1.	The basic abstractions, which are given to the user, are real world functions.	The basic abstractions are not the real world functions but are the data abstraction where the real world entities are represented.
2.	Functions are grouped together by which a higher level function is Page on obtained. aneg of this technique is SA/SD.	Functions are grouped together on the basis of the data they operate since the classes are associated with their methods.
3.	In this approach the state information is often represented in a centralized shared memory.	In this approach the state information is not represented in a centralized memory but is implemented or distributed among the objects of the system.
4.	FOD approach is mainly used for computation sensitive application.	Whereas OOD approach is mainly used for evolving system which mimics a business process or business case.
5.	We decompose in function/procedure level.	We decompose in class level.
6.	Top down approach.	Bottom up approach.
7.	It views system as black box that performs high level function and later decompose it detailed function so to be mapped to modules.	Object-oriented design is the discipline of defining the objects and their interactions to solve a problem that was identified and documented during object-oriented analysis.
8.	Begins by considering the use case diagrams and scenarios.	Begins by identifying objects and classes.

(b) "Fail uniformly" probably means that they fail at a constant rate (with a different rate for each type of machine). Uniformly is not a good word in this context as it implies the machine failure time follows a uniform distribution (this would mean roughly the same number of machines failing in every period, which is not realistic).

Failing at a constant rate λ per unit time happens when the number of fails X in the interval from 0 to t (so λt fails on average) follows a poisson distribution

$$P(X = k) = (\lambda t)^k \exp(-\lambda t) / k!$$

If roughly half of these machines have failed up to time t_0 then half of the remaining will fail in the next interval up to t_0 and so on, decreasing all the time. If T is the time to failure of a machine, which is a random variable, then the probability that the machine is still working after time t is the same as the probability that X is 0 i.e. there has been no fail up to time t . Then

$$P(T > t) = P(X = 0) = \exp(-\lambda t)$$

This is the reliability of the machine type, which is the proportion of machines which are still working after time t . The failure rate λ is $1/\text{MTTF}$ for that type of machine. So for given MTTF we can work out the proportion of machines which will fail up to time t using the reliability. e.g If lightbulbs have a MTTF of 300hrs then the proportion which are still working after 200hrs is

$$\exp(-1/300 \times 200) = 0.513 \text{ or } 51\%$$

The proportion which have failed up to time t is $1 - \text{reliability}$. This can be written as an integral up to time t of the probability density for T . Differentiation will then give the density for the variable T , but the reliability is a more meaningful thing to focus on for practical purposes.

5 Write short notes on :

(a) Object Oriented Analysis modelling

(b) OOD concepts and methods

[Note : 'OOD' read as 'OOD'.]

[R.T.U. 2017]

OR

Explain Object oriented design concepts and methods.

[RTU 2015, 2014]

(a) Refer to Q.9(ii).

(b) Refer to Q.11.

6 Explain Object Oriental Analysis and its Approach and Explain class and Object Relationship Model.

[R.T.U. 2016]

OR

Discuss Object Oriented Analysis (OOA) and modelling in detail.

[RTU 2014]

OR

Describe the Class-Responsibility-Collabrator (CRC).

[RTU 2013]