

Chapter-1

TYPICAL CONFIGURATION OF COMPUTER SYSTEM

➤ Introduction:

- “Computer is an electronic machine that can store, recall and process data. It can perform tasks or complex calculation according to a set of instructions or programs.
- The terms and definitions in the study of computer system are:
- **Hardware:** The physical parts of a computer system called as hardware. The hardware components can be seen, touch and feel. Ex: Keyboard, Mouse, Monitor, RAM, CPU etc.
- **Software:** A Set or collection of programs is known as software. Example Operating System,
- **Data:** Data is the raw information or basic facts that computer can process.
- **User(s):** People who use the computers are called users.

➤ Block diagram of a computer:

- A computer is designed using four basic units. They are:
 1. Input Unit
 2. Central Processing Unit(CPU)
 - Control Unit
 - Arithmetic and Logic Unit (ALU)
 3. Memory Unit
 4. Output Unit

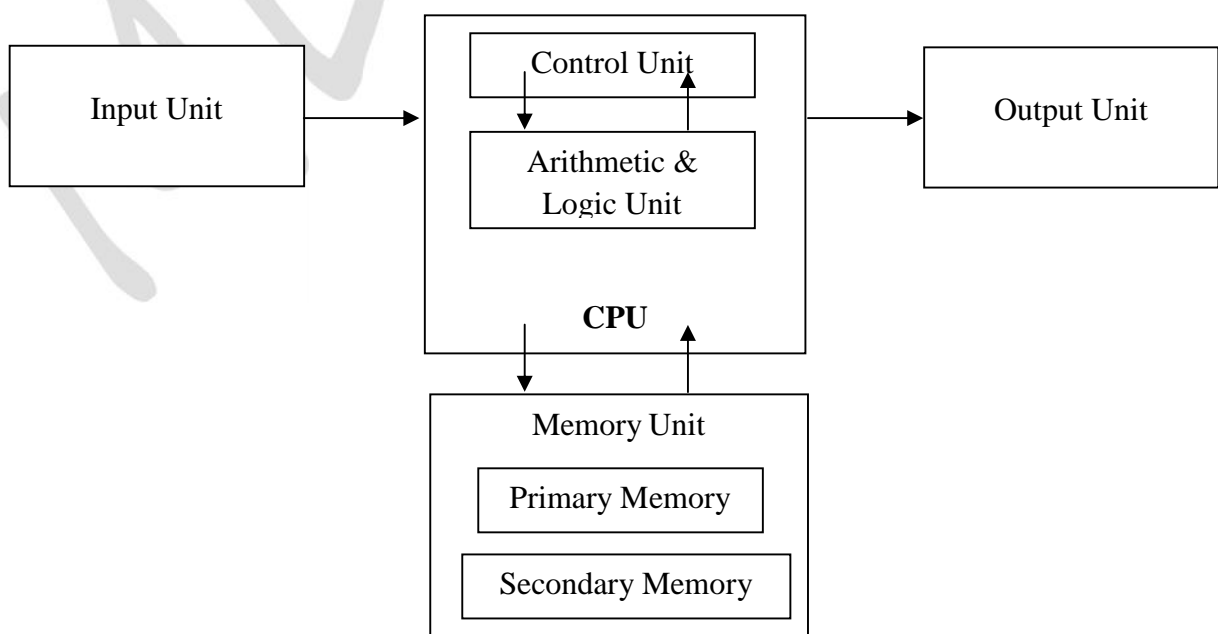


Fig: Block Diagram of Computer

➤ **Input Unit:**

- Computers need to receive data and instructions in order to solve a problem. The Input unit performs this operation.
- The Input Unit basically links the external world or environment to the computer system.
- The input unit may consist of one or more input devices.

➤ **Central Processing Unit (CPU):**

- The function of the CPU is to interpret the instructions in the program and execute them one by one. It consists of two major units.

1. **Control Unit:** It controls and directs the transfer of program instructions and data between various units.
2. **Arithmetic and Logic Unit (ALU):** Arithmetic and Logic Unit performs arithmetic and logical operations and controls the speed of these operations.

➤ **Memory Unit:**

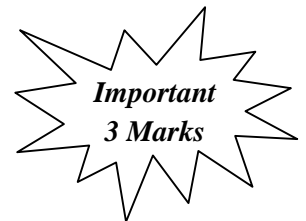
- The results generated from processing have to be preserved before it is displayed.
- The memory units thus provide space to store input data, intermediate results and the final output generated.
- **Note:** The input unit, an output unit, and secondary storage devices are together known as Peripheral Devices.

➤ **Output Unit:**

- It is used to print or display the results, which are stored in the memory unit. The actual function of the output unit is just the reverse of the input unit.
- Thus, the output unit links the computer to the outside world.

➤ **Motherboard:**

- *The motherboard is a main circuit board of the computer, which contains the CPU, memory, expansions slots, bus, video controller and other electronic components.*
- It is a large Printed Circuit Board (PCB).



➤ **Characteristics of Motherboard:**

The motherboard may be characterized by the form factor, chipset and type of processor socket used.

- **Form factor** refers to the motherboard's geometry, dimensions, arrangement and electrical requirements.
- **Chipset** controls the majority of resources of the computer. The function of chipset is to coordinate data transfer between the various components of the computer.

- The **processor socket** may be a rectangular connector into which the processor is mounted vertically, or a square shaped connector with many small connectors into which the processor is directly inserted.

➤ **Types of Motherboard:**

There are four different types in motherboard:

- **XT Motherboards:**

- XT stands for **Extended Technology**.
- These are old model motherboards. In this we find old model processor socket like LIF (Low Insertion Force) sockets, RAM slots: DIMM (Dual Inline Memory Modules) and ISA (Industry Standards Architecture) slots, 12 pin power connector. They have slot type processors and no ports.
- Ex: Pentium-I, Pentium-MMX, Pentium-II and Pentium-Pro.

- **AT Motherboards:**

- AT stands for **Advanced Technology**.
- AT Motherboards have PGA (Pin Grid Array) socket, SD RAM slots, 20 pin power connector PCI slots and ISA slots.
- Full AT is 12” wide X 13.8” deep. AT has 5-pin large keyboard connector.
- Ex: Pentium-III Processors.

- **Baby AT Motherboards:**

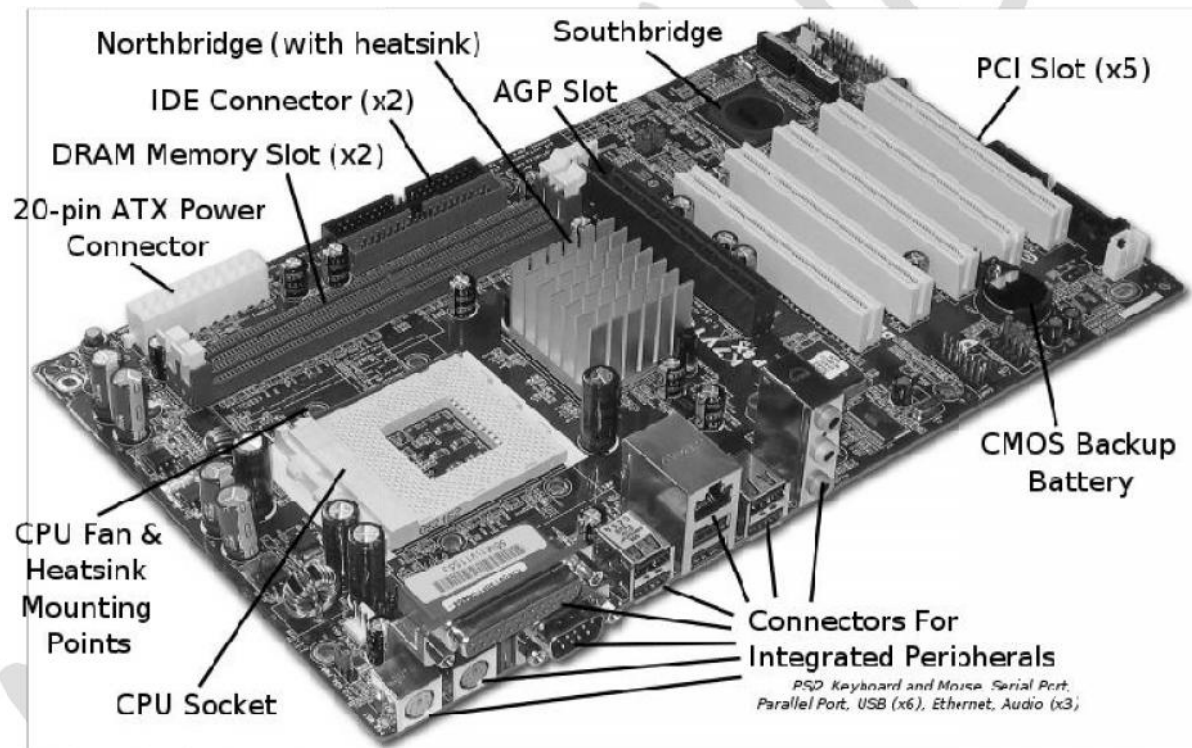
- Baby AT motherboards have the combination of **XT and AT**.
- It was the first PC motherboard to build in sockets for I/O ports, which were cabled to connectors on the back of the case.
- Ex: Pentium-III and Pentium-IV

- **ATX Motherboards:**

- ATX motherboard stands for **Advanced Technology Extended** Motherboard.
- Latest Motherboard all are called as ATX motherboard, designed by ATX form factor.
- In this motherboard, MPGA Processor sockets, DDRRAM Slots, AGP Slots, SATA Connectors, 20 pin and 24 pin ATX power connector and ports
- It is a full size board measuring 12” wide by 9.6” deep. Micro ATX is a small motherboard size of 9.6” X 9.6”.
- Ex: Pentium-IV, Dual Core, Core 2 Duo, Quad Core, i3, i5 and i7.

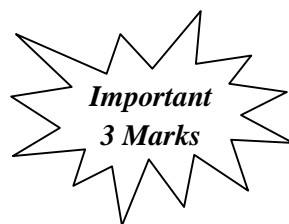
➤ General Structure of Motherboard

- The primary function of the processor is to execute the instructions given to it & produce the results.
- It fetches instructions and data from the primary memory and performs the required operations.
- **North Bridge** and **South Bridge** are two chips in core logic chipset on PC motherboard.
- **North Bridge** or **north chipset** is responsible for control of high speed components like CPU, RAM, Chipset, BUS speed control and switch control data, ensuring data back and forth between the components in a smooth and continuous, fully exploit the speed of the CPU and RAM.
- **South Bridge** or **south chipset** is similar as north chipset, but the south bridge driver chipset components slower as: Sound Card, Net Card, hard disk, etc.



➤ Components of Motherboard:

- The motherboard components are:
 - Processors (CPU)
 - BIOS
 - CMOS
 - Slots
 - Disk Controllers
 - I/O Ports and Interfaces
 - BUS



➤ Processors (CPU):

- *The processors or CPU is the main component on the motherboard and is called the brain of the computer.*

- CPU consists of 1) ALU 2) CU 3) Registers
- Arithmetic and logic unit performs all the arithmetic and logic operations on data.
- CU is responsible for organizing the processing of data and instructions.
- Registers is a temporary storage areas for holding data and instructions.

Note:

- **Clock Speed:** *A measure of a processor's operating speed, currently measured in MHz (Megahertz) and GHz (Gigahertz).*
- A CPU's performance is measured by the number of instructions executed per second i.e. MIPS & BIPS
- **Microprocessor:** *It is an electronic component. It is a single integrated circuit (IC) Chip. This tiny chip contains the entire computation engine. Example: Intel, AMD, Celeron.*

➤ BIOS (Basic Input Output System):

- BIOS is a small chip on the motherboard that holds a set of instructions to load the hardware settings required like keyboard, monitors or disk drives.
- The BIOS runs when the computer is switched ON.
- **POST (Power On Self Test)**
- It checks if the hardware devices are present and functioning properly.
- BIOS include the *bootstrap loader* to load the OS into memory.

➤ CMOS (Complementary Metal Oxide Semiconductor):

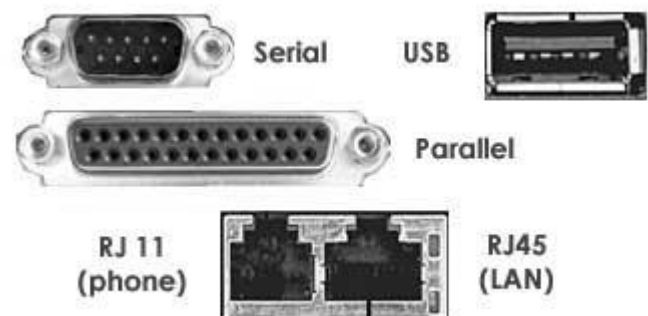
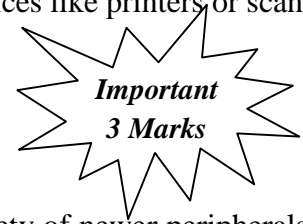
- It is a type of memory chip to store date, time and system setup parameters.
- These parameters are loaded every time the computer is started.
- BIOS & CMOS are kept powered by a small **lithium Ion battery** located on motherboard.

➤ Slots:

- **Slot:** *A slot is an opening space in a computer where we can insert a printed circuit board.*
- Slots are often called expansion slots.
- There are several types of slots are:
- **ISA (Industry Standard Architecture):**
 - ISA slot is used to connect modem and input devices.
- **PCI (Peripheral Component Interconnect):**
 - PCI slots are used to connect graphics accelerators cards, sound card, internal modems or SCSI cards.
 - They are much faster than ISA cards.

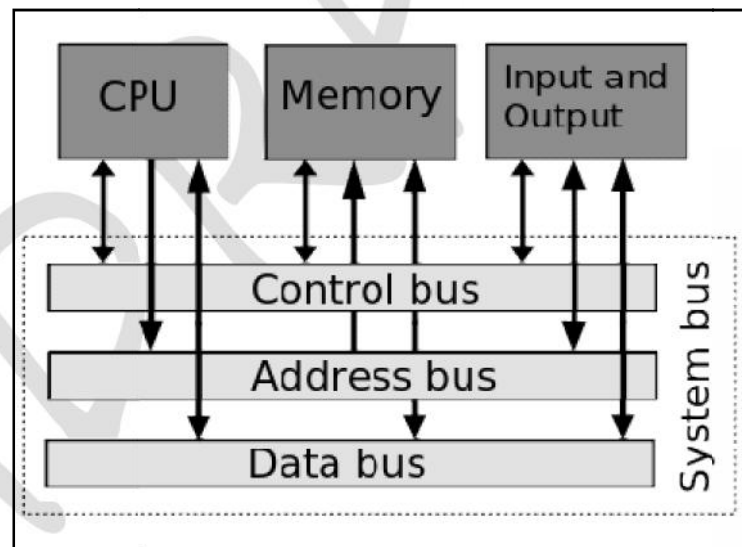
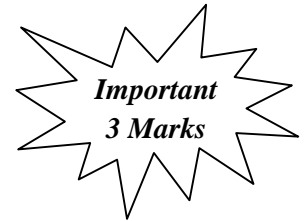
- **AGP (Accelerated Graphic Port):**
 - AGP slot is an advanced port designed for video cards and 3D accelerators.
 - **RAM Slot:**
 - RAM slot is used to install memory to store programs and data currently being used by CPU.
 - RAM is measured in units called bytes.
 - Two types of RAM slot
 - **SIMM (Single Inline Memory Module)**
 - **DIMM (Dual Inline Memory Module)**
 - **Processor Slot:**
 - Processor slot is used to insert the processor chip which is the largest chip on the motherboard.
- **Disk Controllers:**
- *A device that connects a disk drive to the computer's bus enabling the drive to exchange data with other device.*
 - **Hard Disk Controller (HDC)**
 - The HDC is the interface that enables the computer to read and write information to the hard disk drive.
 - This connector is used to insert an Integrated Digital Electronics (IDE) cable.
 - IDE cables connect devices such as hard drives, CD drives and DVD drives.
 - **Floppy Disk Controller (FDC)**
 - FDC is the interface that directs and controls reading from and writing to computer floppy disk drive.
 - FDC usually performs data transmission in Direct Memory Access (DMA) mode.
- **I/O Ports and Interfaces:**
- **Port:** *A port is a socket on the computer used to connect external device to the computer.*
 - It is used to connect external device like printer, keyboard or scanner.
 - The different types of I-O ports are Serial port, Parallel port, USB port and VGA port.
 - **Serial Port:**
 - Serial Port, also known as communication port or Rs-232 c ports, is used for connecting communication devices like mouse and modem.
 - They are used for connecting communication devices like mouse, modem.
 - This port transfers data serially one bit at a time.

- One main advantage is that data is sent and received over only two lines.
- **Parallel Port:**
 - Parallel ports are used to connect external input/output devices like printers or scanners.
 - Also known as printer port
 - They carry 8 bit (one byte) at a time.
- **USB port :**
 - USB (Universal Serial BUS) port is used to connect a variety of newer peripherals like printers, scanners, digital cameras, web cameras, speakers, etc. to a computer.
 - USB port gives a single, standardized, easy-to-use way to connect a variety of newer peripherals to a computer.
 - **USB is a plug-and-play interface between a computer and add-on devices such as audio players, modem, scanner etc.**
 - **With USB, a new device can be added to your computer without adding a adapter card or even turning the computer off**
 - USB supports a data speed of 12 megabits per second
 - USB supporting up to 127 devices.
- **PS-2 (Personal System) port:**
 - PS-2 port was developed by IBM to interface keyboards and pointing devices like mouse, trackballs and touch pads.
- **IDE (Integrated Digital Electronics) port :**
 - It connects IDE devices like CD-ROM drives or hard disk drives to the motherboard.
- **AGP (Accelerated Graphics Port) port:**
 - It is used to connect to graphic card that provides high-speed video performance typically required in games and other multimedia applications.
- **VGA (Visual Graphics Adaptor) port:** It connects monitor to a computer's video card.
- **Modem (Modulator demodulator)** connects a PC to the telephone network.
- **Ethernet port** connects to a network and high speed Internet. It connects network cable to a computer.
- **MIDI (Musical Instrument Digital Interface) port** is a system designed to transmit information between electronic musical instruments.



➤ Bus:

- *A bus is a collection of parallel wires that form a pathway to carry address, data and control signal.*
- The functional features of bus are:
 - A bus is a set of wire and each wire can carry one bit of data.
 - A bus width is defined by the number of wires in the bus
- A computer bus can be divided into two types
 - **Internal Bus:**
 - It connects major computer components like processor, memory & I/O.
 - It is also called as system bus.
 - **External Bus:**
 - It connects the different external devices peripheral, expansion slots, I/O ports to the rest of the computer.
 - It is also called the expansion bus and is slower than the system (internal) bus.
- A system bus or expansion bus comprise of three kinds of buses:



- **Data Bus:**
 - It provides a path to transfer data between CPU and memory.
 - The data bus may consists of 32, 64, 128 lines of wire.
- **Address Bus:**
 - It connects CPU & RAM with a set of lines similar to data bus.
 - The address bus width determines the maximum number of memory location the computer can address.
- **Control Bus:**
 - It is used to control the access to and the use of the data and address lines.

➤ Memory:

- *A computer memory refers to the electronic storing space for instructions and data.*
- Two kinds of memory are commonly used:
 - Primary or Main Memory
 - Secondary Memory
- **Primary Memory:**
 - *Primary memory is the main memory of the computer.*
 - It stores programs and data which are currently needed by CPU.
 - Functions of primary memory:
 - To contain a copy of the main software program i.e. operating system. This program is loaded into the primary memory when the computer is turned on.
 - Temporarily store a copy of the application program.
 - Temporarily store the data input from the keyboard.
 - Temporarily store the result which is generated from processing until it is transferred to output device.
 - Primary memory is of two types.
 1. RAM (Random Access Memory)
 2. ROM (Read Only Memory)

➤ Random Access Memory (RAM):

- *RAM is also called as the main memory of a computer.*
- Ram temporarily stores the computer operating system, application program and current data so that the processor can reach them quickly.
- RAM is faster memory.
- RAM is a volatile in nature i.e. when the power is switched off; the data in this memory is lost.
- Types of RAM
 - **Static RAM (SRAM)**
 - Static RAM chip is usually used in cache memory due to its high speed.
 - It stores information as long as the power supply is on.
 - SRAM is more expensive than DRAM and it takes up more space.
 - **Dynamic RAM (DRAM)**
 - It is the most common type of memory chip.
 - DRAM is cheaper and they consume less power.
 - It uses transistor and capacitors.

- They are further classified as
- **SDRAM**
 - SDRAM stands for *Synchronous Dynamic RAM*.
 - It is synchronized to the system clock.
 - Since it is synchronized to the CPU, it known's when the next cycle is coming and has the data ready when the CPU requests it.
- **DDR RAM**
 - DDR RAM stands for *Double Data Rate RAM*.
 - It works same as SD RAM but the data transfer rate is double when compared to SD RAM.
 - Different types of DDR RAM are DDR1, DDR2, and DDR3.

➤ **Read Only Memory (ROM):**

- ROM stands for “**Read Only memory**”.
- ROM is **non-volatile memory** i.e. the information stored in it is not lost even when the power supply goes off.
- It is used for permanent storage of information.

➤ **Difference between RAM and ROM:**

RAM	ROM
RAM stands for Random Access Memory	ROM stands for Read-Only Memory
Volatile Memory	Non-volatile Memory
RAM require a flow of electrically to retain data	ROM will retain data without the flow of electricity
Type: DRAM, SRAM	Type: PROM, EPROM

➤ **Secondary Memory:**

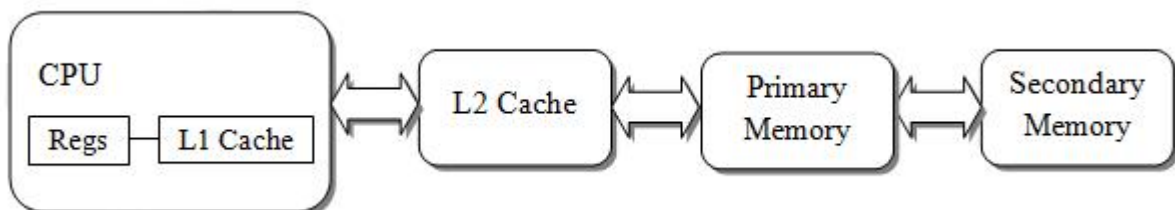
- The secondary memory is used as auxiliary memory. The secondary memory is used for bulk storage or mass storage of programs, data, and other information.
- It has much larger capacity than main memory.
- The secondary memory retains the information once stored on it.
- The magnetic memory such as Hard Disk Drive (HDD), Compact Disk, Pen Drive, Memory cards is the most commonly used secondary memory in the computer.

➤ Difference between Primary and Secondary Memory:

Primary Memory	Secondary Memory
Semi conductor Memory	Magnetic or Optical Memory
Volatile	Non-Volatile
Expensive	Less Expensive
Faster	Slower
Main Memory	Auxiliary Memory
Example: RAM, ROM	Example: HDD, Pen drive etc

➤ Cache Memory:

- The cache memory is a very high speed memory placed in between RAM and CPU.
- Cache memory stores data that is used more often, temporarily and makes it available to CPU at fast rate. Hence it is used to increase the speed of processing.
- Cache memory is very expensive, so it is smaller in size.
- Cache memory of sizes 256 KB to 2 MB.
- It is categorized as “levels”.
- **Level 1 (L1) cache:** It is extremely fast but relatively small and is usually present inside the CPU. The size of L1 cache varies from 32 KB to 512 KB
- **Level 2 (L2) cache:** It may be located outside the CPU on a separate chip a high speed system bus interconnecting the cache to the CPU. The size of L2 cache varies from 1MB to 2MB.
- **Level 3 (L3) cache:** It is typically specialized memory that works to improve the performance of L1 and L2. It is slower the L1 or L2 but it is usually double the speed of RAM.



➤ Switch Mode Power Supply:

- SMPS stands for *Switch Mode Power Supply*.
- An SMPS converts AC power from an electrical outlet to DC power needed by system components.
- The SMPS contains the power card plug, a fan for cooling because it generates a lot of heat.

➤ **UPS:**

- UPS stands for “*Uninterruptible Power Supply*”.
- An UPS is a power supply that includes a battery to maintain power in the event a power failure.
- An UPS keeps a computer running for several minutes to few hours after a power failure.
- There are two types of UPS
- **Offline UPS:**
 - Also known as “Standby UPS”
 - Offline UPS monitors the power line and switches to battery power as soon as it detects a problem.
- **Online UPS:**
 - An online UPS continuously provides power from its own inverter, even when the power line is functioning properly.
 - Online UPS is more costly than offline UPS.



CHAPTER 1 – TYPICAL CONFIGURATION OF COMPUTER SYSTEM BLUE PRINT				
VSA (1 marks)	SA (2 marks)	LA (3 Marks)	Essay (5 Marks)	Total
01 Question	-	01 Question	-	02 Question
Question no 1	-	Question no 19	-	04 Marks

Important Questions

➤ **1 Marks Question:**

1. What is Microprocessor?
2. What is Motherboard? [June 2015]
3. Expand USB.
4. What is data bus? [March 2015]
5. Which memory is known as main memory?
6. Which memory is considered as working memory of CPU?
7. What is Cache Memory? [June 2016]
8. What is Bus? [March 2017]
9. Expand SDRAM.
10. Expand ISA [March 2016]
11. Expand SMPS.

➤ 3 Marks Question:

1. Explain the characteristics of motherboard. [March 2017]
2. Explain three types of motherboard. [June 2016]
3. Mention the components of the motherboard.
4. What is port? Explain serial port. [March 2015]
5. What is meant by plug and play device? Explain.
6. Write any three features of USB.
7. Explain Cache memory.
8. What is the function of UPS? Mention the different types of UPS. [June 2015, March 2016]

MDRPUC

Chapter-2

BOOLEAN ALGEBRA

➤ Introduction:

- An algebra that deals with binary number system is called “Boolean Algebra”.
- It is very power in designing logic circuits used by the processor of computer system.
- The logic gates are the building blocks of all the circuit in a computer.
- Boolean algebra derives its name from the mathematician **George Boole** (1815-1864) who is considered the “**Father of symbolic logic**”.
- Boolean algebra deals with truth table TRUE and FALSE.
- It is also called as “**Switching Algebra**”.

➤ Binary Valued Quantities – Variable and Constants:

- A variable used in Boolean algebra or Boolean equation can have only one of two variables. The two values are FALSE (0) and TRUE (1)
- A Sentence which can be determined to be TRUE or FALSE are called **logical statements** or **truth functions** and the results TRUE or FALSE is called **Truth values**.
- The variables which can store the truth values are called **logical variables or binary valued variables**. These can store one of the two values 1 or 0.
- The decision which results into either YES (TRUE or 1) or NO (FALSE or 0) is called **Binary decision**.

➤ Truth Table:

- A truth table is a mathematical table used in logic to computer functional values of logical expressions.
- A truth table is a table whose columns are statements and whose rows are possible scenarios.
- Example: Consider the logical expression

Logical Statement: Meals = “Ram prefer rice and roti for the meal”

$Y = A \text{ AND } B$ (Logical Variables: Y, A, B, Logical Operator AND)

Ram Prefer Rice	Ram Prefer Roti	Meals
FALSE	FALSE	FALSE
FALSE	TRUE	FALSE
TRUE	FALSE	FALSE
TRUE	TRUE	TRUE

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	0

- If result of any logical statement or expression is always TRUE or 1, it is called **Tautology** and if the result is always FALSE or 0, it is called **Fallacy**.

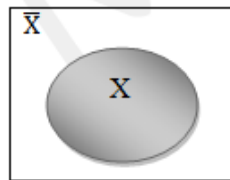
➤ Logical Operators:

- There are three logical operator, NOT, OR and AND.
- These operators are now used in computer construction known as switching circuits.

➤ NOT Operator:

- The Not operator is a unary operator. This operator operates on single variable.
- The operation performed by Not operator is called **complementation**.
- The symbol we use for it is bar.
- \bar{X} means complementation of X
- If $X=1, \bar{X}=0$ If $X=0, \bar{X}=1$
- The Truth table and the Venn diagram for the NOT operator is:

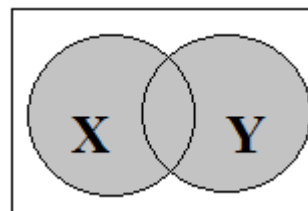
X	\bar{X}
1	0
0	1



➤ OR Operator:

- The OR operator is a binary operator. This operator operates on two variables.
- The operation performed by OR operator is called **logical addition**.
- The symbol we use for it is '+’.
- Example: $X + Y$ can be read as **X OR Y**
- The Truth table and the Venn diagram for the NOT operator is:

X	Y	$X + Y$
0	0	0
0	1	1
1	0	1
1	1	1

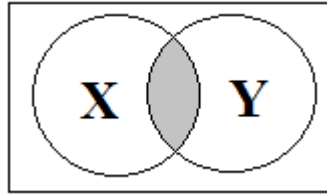


➤ AND Operator:

- The AND operator is a binary operator. This operator operates on two variables.
- The operation performed by AND operator is called **logical multiplication**.
- The symbol we use for it is '·’.
- Example: $X \cdot Y$ can be read as **X AND Y**

- The Truth table and the Venn diagram for the NOT operator is:

X	Y	$X \cdot Y$
0	0	0
0	1	0
1	0	0
1	1	1



➤ Evaluation of Boolean Expression using Truth Table:

- To create a truth table, follow the steps given below.
- Step 1: Determine the number of variables, for n variables create a table with 2^n rows.
 - For two variables i.e. X, Y then truth table will need 2^2 or 4 rows.
 - For three variables i.e. X, Y, Z, then truth table will need 2^3 or 8 rows.
- Step 2: List the variables and every combination of 1 (TRUE) and 0 (FALSE) for the given variables
- Step 3: Create a new column for each term of the statement or argument.
- Step 4: If two statements have the same truth values, then they are equivalent.

➤ Example: Consider the following Boolean Expression $F = X + \bar{Y}$

- Step 1: This expression as two variables X and Y, then 2^2 or 4 rows.
- Step 2: List the variables and every combination of X and Y.
- Step 3: Create a new column \bar{Y} of the statement, and then fill the truth values of Y in that column.
- Step 4: The final column contain the values of $X + \bar{Y}$.

X	Y	\bar{Y}	$X + \bar{Y}$
0	0	1	1
0	1	0	0
1	0	1	1
1	1	0	1

➤ Exercise Problems:

- Prepare a table of combination for the following Boolean algebra expressions.
 - $\bar{X}\bar{Y} + \bar{X}Y$
 - $XY\bar{Z} + \bar{X}\bar{Y}Z$
- Verify using truth table for the following Boolean algebra.
 - $X + XY = X$
 - $\overline{X + Y} = \bar{X} \cdot \bar{Y}$

➤ Boolean Postulates:

- The fundamental laws of Boolean algebra are called as the postulates of Boolean algebra.
- These postulates for Boolean algebra originate from the three basic logic functions AND, OR and NOT.
- **Properties of 0 and 1:**
 - I. If $X \neq 0$ then $X = 1$, and If $X \neq 1$ then $X = 0$
 - II. OR relation (Logical Addition)

a. $0 + 0 = 0$	c. $1 + 0 = 1$
b. $0 + 1 = 1$	d. $1 + 1 = 1$
 - III. AND relation (Logical Multiplication)

a. $0 \cdot 0 = 0$	c. $1 \cdot 0 = 0$
b. $0 \cdot 1 = 0$	d. $1 \cdot 1 = 1$
 - IV. Complement Rules

a. $\bar{0} = 1$	b. $\bar{1} = 0$
------------------	------------------

➤ Principle of Duality Theorem:

- This is very important principle used in Boolean algebra.
- Principle of Duality states that;
 - Changing each OR sign (+) to an AND sign (.)
 - Changing each AND sign (.) to an OR sign (+)
 - Replacing each 0 by 1 and each 1 by 0.
- The derived relation using duality principle is called dual of original expression.
- Example: Take postulate II, related to logical addition:
 - 1) $0 + 0 = 0$ 2) $0 + 1 = 1$ 3) $1 + 0 = 1$ 4) $1 + 1 = 1$
- 2. Now working according to above relations, + is changed to . and 0's replaced by 1's
 - a) $1 \cdot 1 = 1$ b) $1 \cdot 0 = 0$ c) $0 \cdot 1 = 0$ d) $0 \cdot 0 = 0$
- which are nothing but same as that of postulate III related to logical multiplication.
- So 1, 2, 3, 4, are the duals of a, b, c, d.
- Example: Find the duals for the following Boolean Expression

Sl No	Boolean Expression	Duals
1	$X + 0 = X$	$X \cdot 1 = X$
2	$X + 1 = 1$	$X \cdot 0 = 0$
3	$X \cdot \bar{X} = 0$	$X + \bar{X} = 1$
4	$X \cdot (Y + Z)$	$X + (Y \cdot Z)$
5	$X + X \cdot Y = X + Y$	$X \cdot (X + Y) = X \cdot Y$

➤ **Boolean Theorems:**

- Boolean Theorem can be proved by substituting all possible values of the variable that are 0 and 1.
- This technique of proving theorem is called **Proof by perfect induction.**

Sl No	Theorem	Sl No	Theorem
Properties of 0 and 1		Associative Law	
1	$0 + X = X$	12	$X \cdot (Y \cdot Z) = (X \cdot Y) \cdot Z$
2	$1 + X = 1$	13	$(X + Y) \cdot Z = X + (Y \cdot Z)$
3	$0 \cdot X = 0$	Distributive Law	
4	$1 \cdot X = X$	14	$X \cdot (Y + Z) = X \cdot Y + X \cdot Z$
Idempotence Law		15	$X + Y \cdot Z = (X + Y) \cdot (X + Z)$
5	$X + X = X$	Absorption Law	
6	$X \cdot X = X$	16	$X + XY = X$
Complementary Law		17	$X(X + Y) = X$
7	$X + \bar{X} = 1$	18	$XY + X\bar{Y} = X$
8	$X \cdot \bar{X} = 0$	19	$(X + Y)(X + \bar{Y}) = X$
Involution Law		20	$X + \bar{\bar{X}}Y = X + Y$
9	$\bar{\bar{X}} = X$	21	$X(\bar{X} + Y) = XY$
Commutative Law			
10	$X + Y = Y + X$		
11	$X \cdot Y = Y \cdot X$		

➤ **Theorem 1: $0 + X = X$**

Proof: If $X = 0$ then LHS $= 0 + X$ $= 0 + 0$ $= 0$ $= \text{RHS}$	Proof: If $X = 1$ then LHS $= 0 + X$ $= 0 + 1$ $= 1$ $= \text{RHS}$	Using Truth Table <table border="1"> <thead> <tr> <th>0</th> <th>X</th> <th>0+X</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	0	X	0+X	0	0	0	0	1	1
0	X	0+X									
0	0	0									
0	1	1									

➤ **Theorem 2: $1 + X = 1$**

Proof: If $X = 0$ then LHS $= 1 + X$ $= 1 + 0$ $= 1$ $= \text{RHS}$	Proof: If $X = 1$ then LHS $= 1 + X$ $= 1 + 1$ $= 1$ $= \text{RHS}$	Using Truth Table <table border="1"> <thead> <tr> <th>1</th> <th>X</th> <th>1+X</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	1	X	1+X	1	0	1	1	1	1
1	X	1+X									
1	0	1									
1	1	1									

➤ **Theorem 3:** $0 \cdot X = 0$

Proof: If $X = 0$ then LHS $= 0 \cdot X$ $= 0 \cdot 0$ $= 0$ $= \text{RHS}$	Proof: If $X = 1$ then LHS $= 0 \cdot X$ $= 0 \cdot 1$ $= 0$ $= \text{RHS}$	Using Truth Table <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <th>0</th> <th>X</th> <th>0.X</th> </tr> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> </table>	0	X	0.X	0	0	0	0	1	0
0	X	0.X									
0	0	0									
0	1	0									

➤ **Theorem 4:** $1 \cdot X = X$

Proof: If $X = 0$ then LHS $= 1 \cdot X$ $= 1 \cdot 0$ $= 0$ $= \text{RHS}$	Proof: If $X = 1$ then LHS $= 1 \cdot X$ $= 1 \cdot 1$ $= 1$ $= \text{RHS}$	Using Truth Table <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <th>1</th> <th>X</th> <th>0.X</th> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </table>	1	X	0.X	1	0	0	1	1	1
1	X	0.X									
1	0	0									
1	1	1									

➤ **Idempotence Law:** “This law states that when a variable is combined with itself using OR or AND operator, the output is the same variable”.

➤ **Theorem 5:** $X + X = X$

Proof: If $X = 0$ then LHS $= X + X$ $= 0 + 0$ $= 0$ $= \text{RHS}$	Proof: If $X = 1$ then LHS $= X + X$ $= 1 + 1$ $= 1$ $= \text{RHS}$	Using Truth Table <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <th>X</th> <th>X</th> <th>X+X</th> </tr> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </table>	X	X	X+X	0	0	0	1	1	1
X	X	X+X									
0	0	0									
1	1	1									

➤ **Theorem 6:** $X \cdot X = X$

Proof: If $X = 0$ then LHS $= X \cdot X$ $= 0 \cdot 0$ $= 0$ $= \text{RHS}$	Proof: If $X = 1$ then LHS $= X \cdot X$ $= 1 \cdot 1$ $= 1$ $= \text{RHS}$	Using Truth Table <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <th>X</th> <th>X</th> <th>X.X</th> </tr> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </table>	X	X	X.X	0	0	0	1	1	1
X	X	X.X									
0	0	0									
1	1	1									

➤ **Complementary Law:** “This law states that when a variable is AND ed with its complement is equal to 0 and a variable is OR ed with its complement is equal to 1”.

➤ **Theorem 7:** $X + \bar{X} = 1$

Proof: If $X = 0$ then LHS $= X + \bar{X}$ $= 0 + 1$ $= 1$ $= \text{RHS}$	Proof: If $X = 1$ then LHS $= X + \bar{X}$ $= 1 + 0$ $= 1$ $= \text{RHS}$	Using Truth Table <table border="1" style="width: 100%;"> <thead> <tr> <th>X</th> <th>\bar{X}</th> <th>$X + \bar{X}$</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> </tbody> </table>	X	\bar{X}	$X + \bar{X}$	0	1	1	1	0	1
X	\bar{X}	$X + \bar{X}$									
0	1	1									
1	0	1									

➤ **Theorem 8:** $X \cdot \bar{X} = 0$

Proof: If $X = 0$ then LHS $= X \cdot \bar{X}$ $= 0 \cdot 1$ $= 0$ $= \text{RHS}$	Proof: If $X = 1$ then LHS $= X \cdot \bar{X}$ $= 1 \cdot 0$ $= 0$ $= \text{RHS}$	Using Truth Table <table border="1" style="width: 100%;"> <thead> <tr> <th>X</th> <th>\bar{X}</th> <th>$X \cdot \bar{X}$</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> </tbody> </table>	X	\bar{X}	$X \cdot \bar{X}$	0	1	0	1	0	0
X	\bar{X}	$X \cdot \bar{X}$									
0	1	0									
1	0	0									

➤ **Involution Law:** “This law states that when a variable is inverted twice is equal to the original variable”.

➤ **Theorem 9:** $\bar{\bar{X}} = X$

Proof: If $X = 0$, then $\bar{X} = 1$ Take complement again, then $\bar{\bar{X}} = 0$ i.e. X If $X = 1$, then $\bar{X} = 0$ Take complement again, then $\bar{\bar{X}} = 1$ i.e. X	Using Truth Table <table border="1" style="width: 100%;"> <thead> <tr> <th>X</th> <th>\bar{X}</th> <th>$\bar{\bar{X}}$</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> </tbody> </table>	X	\bar{X}	$\bar{\bar{X}}$	0	1	0	1	0	1
X	\bar{X}	$\bar{\bar{X}}$								
0	1	0								
1	0	1								

➤ **Commutative Law:** “This law states that the order in which two variable are Or ed or AND ed make no difference”.

➤ **Theorem 10:** $X + Y = Y + X$

Proof: If $Y = 0$ then LHS $= X + Y$ $= X + 0$ $= X$ RHS $= Y + X$ $= 0 + X$ $= X$ Therefore LHS = RHS	Proof: If $Y = 1$ then LHS $= X + Y$ $= X + 1$ $= 1$ RHS $= Y + X$ $= 1 + X$ $= 1$ Therefore LHS = RHS	Using Truth Table <table border="1" style="width: 100%;"> <thead> <tr> <th>X</th> <th>Y</th> <th>$X+Y$</th> <th>$Y+X$</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	X	Y	$X+Y$	$Y+X$	0	0	0	0	0	1	1	1	1	0	1	1	1	1	1	1
X	Y	$X+Y$	$Y+X$																			
0	0	0	0																			
0	1	1	1																			
1	0	1	1																			
1	1	1	1																			

➤ **Theorem 11:** $X \cdot Y = Y \cdot X$

<p>Proof: If $Y = 0$</p> <p>then LHS $= X \cdot Y$ $= X \cdot 0$ $= 0$</p> <p>RHS $= Y \cdot X$ $= 0 \cdot X$ $= 0$</p> <p>Therefore LHS = RHS</p>	<p>Proof: If $Y = 1$</p> <p>then LHS $= X \cdot Y$ $= X \cdot 1$ $= X$</p> <p>RHS $= Y \cdot X$ $= 1 \cdot X$ $= X$</p> <p>Therefore LHS = RHS</p>	<p>Using Truth Table</p> <table border="1"> <thead> <tr> <th>X</th> <th>Y</th> <th>X.Y</th> <th>Y.X</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	X	Y	X.Y	Y.X	0	0	0	0	0	1	0	0	1	0	0	0	1	1	1	1
X	Y	X.Y	Y.X																			
0	0	0	0																			
0	1	0	0																			
1	0	0	0																			
1	1	1	1																			

➤ **Associative Law:** “This law allows the removal of brackets from an expression and regrouping of the variables”.

➤ **Theorem 12:** $X \cdot (Y \cdot Z) = (X \cdot Y) \cdot Z$

<p>Proof: If $Y = 0$</p> <p>LHS $= X \cdot (Y \cdot Z)$ $= X \cdot (0 \cdot Z)$ $= X \cdot 0$ $= 0$</p> <p>RHS $= (X \cdot Y) \cdot Z$ $= (X \cdot 0) \cdot Z$ $= 0 \cdot Z$ $= 0$</p> <p>Therefore LHS = RHS</p>	<p>Proof: If $Y = 1$</p> <p>LHS $= X \cdot (Y \cdot Z)$ $= X \cdot (1 \cdot Z)$ $= X \cdot Z$ $= XZ$</p> <p>RHS $= (X \cdot Y) \cdot Z$ $= (X \cdot 1) \cdot Z$ $= X \cdot Z$ $= XZ$</p> <p>Therefore LHS = RHS</p>	<p>Using Truth Table</p> <table border="1"> <thead> <tr> <th>X</th> <th>Y</th> <th>Z</th> <th>XY</th> <th>YZ</th> <th>X.(Y.Z)</th> <th>(X.Y).Z</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>0</td> <td>1</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> <td>0</td> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> <td>1</td> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> <td>1</td> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	X	Y	Z	XY	YZ	X.(Y.Z)	(X.Y).Z	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	1	1	0	1	0	0	1	0	0	0	0	0	0	1	0	1	0	0	0	0	1	1	0	1	0	0	0	1	1	1	1	1	1	1
X	Y	Z	XY	YZ	X.(Y.Z)	(X.Y).Z																																																											
0	0	0	0	0	0	0																																																											
0	0	1	0	0	0	0																																																											
0	1	0	0	0	0	0																																																											
0	1	1	0	1	0	0																																																											
1	0	0	0	0	0	0																																																											
1	0	1	0	0	0	0																																																											
1	1	0	1	0	0	0																																																											
1	1	1	1	1	1	1																																																											

➤ **Theorem 13:** $X + (Y + Z) = (X + Y) + Z$

<p>Proof: If $Y = 0$</p> <p>LHS $= X + (Y + Z)$ $= X + (0 + Z)$ $= X + Z$</p> <p>RHS $= (X + Y) + Z$ $= (X + 0) + Z$ $= X + Z$</p> <p>Therefore LHS = RHS</p>	<p>Proof: If $Y = 1$</p> <p>LHS $= X + (Y + Z)$ $= X + (1 + Z)$ $= X + 1$ $= 1$</p> <p>RHS $= (X + Y) + Z$ $= (X + 1) + Z$ $= 1 + Z$ $= 1$</p> <p>Therefore LHS = RHS</p>	<p>Using Truth Table</p> <table border="1"> <thead> <tr> <th>X</th> <th>Y</th> <th>Z</th> <th>X+Y</th> <th>Y+Z</th> <th>X+(Y+Z)</th> <th>(X+Y)+Z</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>0</td> <td>1</td> <td>0</td> <td>1</td> <td>1</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> <td>1</td> <td>1</td> <td>1</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> <td>1</td> <td>1</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> <td>1</td> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> <td>1</td> <td>1</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> <td>1</td> <td>1</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> <td>1</td> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	X	Y	Z	X+Y	Y+Z	X+(Y+Z)	(X+Y)+Z	0	0	0	0	0	0	0	0	0	1	0	1	1	1	0	1	0	1	1	1	1	0	1	1	1	1	1	1	1	0	0	1	0	1	1	1	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1
X	Y	Z	X+Y	Y+Z	X+(Y+Z)	(X+Y)+Z																																																											
0	0	0	0	0	0	0																																																											
0	0	1	0	1	1	1																																																											
0	1	0	1	1	1	1																																																											
0	1	1	1	1	1	1																																																											
1	0	0	1	0	1	1																																																											
1	0	1	1	1	1	1																																																											
1	1	0	1	1	1	1																																																											
1	1	1	1	1	1	1																																																											

➤ **Distributive Law:** “This law allows the multiplying or factoring out an expression”.

➤ **Theorem 14:** $X.(Y+Z) = XY + XZ$

Proof: If $X = 0$ LHS = $X.(Y+Z)$ $= 0.(Y+Z)$ $= 0$ RHS = $XY + XZ$ $= 0.Y+0.Z$ $= 0$ Therefore LHS = RHS	Proof: If $X = 1$ LHS = $X.(Y+Z)$ $= 1.(Y+Z)$ $= Y+Z$ RHS = $XY + XZ$ $= 1.Y+1.Z$ $= Y+Z$ Therefore LHS = RHS
--	--

➤ **Theorem 15:** $(X + Y) (X + Z) = X + YZ$

$$\begin{aligned}
 \text{LHS: } & (X + Y) (X + Z) = XX + XZ + XY + YZ \\
 & = X + XZ + XY + YZ \\
 & = X(1 + Z) + XY + YZ \\
 & = X + XY + YZ \\
 & = X(1 + Y) + YZ \\
 & = X + YZ \\
 & = \text{RHS}
 \end{aligned}$$



➤ **Absorption Law:** “This law enables a reduction of complicated expression to a simpler one by absorbing common terms”.

16) $X+XY = X$ LHS = $X + XY$ $= X(1 + Y)$ $= X$ $= \text{RHS}$	17) $X(X+Y) = X$ LHS = $X(X+Y)$ $= XX + XY$ $= X + XY$ $= X(1+Y)$ $= X$	18) $XY + X\bar{Y} = X$ LHS = $XY + X\bar{Y}$ $= X(Y+\bar{Y})$ $= X.1$ $= X$ $= \text{RHS}$
19) $(X+Y)(X+\bar{Y}) = X$ LHS = $(X+Y)(X+\bar{Y})$ $= XX + X\bar{Y} + XY + Y\bar{Y}$ $= X + X\bar{Y} + XY + 0$ $= X(1 + \bar{Y} + Y)$ $= X.1$ $= X$	20) $X + \bar{X}Y = X+Y$ LHS = $X + \bar{X}Y$ $= (X + \bar{X})(X+Y)$ $= 1.(X+Y)$ $= X+Y$ $= \text{RHS}$	21) $X(\bar{X}+Y) = XY$ LHS = $X(\bar{X}+Y)$ $= X.\bar{X} + X.Y$ $= 0 + XY$ $= XY$ $= \text{RHS}$

➤ DeMorgan's Theorem:

• DeMorgan's First Theorem:

- **Statement:** "When the OR sum of two variables is inverted, this is same as inverting each variable individually and then AND ing these inverted variables"
- This can be written as $\overline{X + Y} = \bar{X} \cdot \bar{Y}$
- We can prove the DeMorgan's First theorem by using Truth Table is

X	Y	\bar{X}	\bar{Y}	X+Y	$\overline{X + Y}$	$\bar{X} \cdot \bar{Y}$
0	0	1	1	0	1	1
0	1	1	0	1	0	0
1	0	0	1	1	0	0
1	1	0	0	1	0	0



- Compare the column $\overline{X + Y}$ and $\bar{X} \cdot \bar{Y}$. Both of these are identical. Hence the DeMorgan's first theorem is proved.

• DeMorgan's Second Theorem:

- **Statement:** "When the AND product of two variables is inverted, this is same as inverting each variable individually and then OR ing these inverted variables"
- This can be written as $\overline{X \cdot Y} = \bar{X} + \bar{Y}$
- We can prove the DeMorgan's Second theorem by using Truth Table is:

X	Y	\bar{X}	\bar{Y}	X.Y	$\overline{X \cdot Y}$	$\bar{X} + \bar{Y}$
0	0	1	1	0	1	1
0	1	1	0	0	1	1
1	0	0	1	0	1	1
1	1	0	0	1	0	0

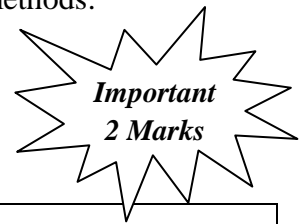
- Compare the column $\overline{X \cdot Y}$ and $\bar{X} + \bar{Y}$. Both of these are identical. Hence the DeMorgan's Second theorem is proved.

• Application of DeMorgan's Theorem:

- It is used in simplification of Boolean expression.
- DeMorgan's law commonly apply to text searching using Boolean operators AND, OR and NOT.
- It is useful in the implementation of the basic gates operations with alternative gates.

➤ **Simplification of Boolean Expression:**

- Simplification of Boolean expression can be achieved by two popular methods:
 - Algebraic Manipulation
 - Karnaugh Maps
- **Algebraic Manipulation:**



<p>1) $\bar{X}\bar{Y}Z + \bar{X}YZ + X\bar{Y}$</p> $= \bar{X}\bar{Y}Z + \bar{X}YZ + X\bar{Y}$ $= \bar{X}Z(\bar{Y} + Y) + X\bar{Y}$ $= \bar{X}Z(1) + X\bar{Y}$ $= \bar{X}Z + X\bar{Y}$	<p>2) $XYZ + XYZW + XZ$</p> $= XYZ(1 + W) + XZ$ $= XYZ \cdot 1 + XZ$ $= XZ(Y + 1)$ $= XZ$
<p>3) $Z(Y+Z)(X+Y+Z)$</p> $= (ZY + ZZ)(X+Y+Z)$ $= (ZY + Z)(X+Y+Z)$ $= Z(X+Y+Z) \text{ [Theorem 16 } X+XY=Z]$ $= ZX + ZY + ZZ$ $= ZX + ZY + Z$ $= Z(X+Y+1)$ $= Z(1)$ $= Z$	<p>4) $X + \bar{X}Y + \bar{Y} + (X + \bar{Y})\bar{X}Y$</p> $= X + \bar{X}Y + \bar{Y} + X\bar{X}Y + \bar{Y}\bar{X}Y$ $= X + \bar{X}Y + \bar{Y} + 0 + 0$ $= (X + \bar{X})(X + Y) + \bar{Y}$ $= 1(X + Y) + \bar{Y}$ $= X + Y + \bar{Y}$ $= X + 1$ $= 1$

➤ **Exercise Problems: Simplify using Algebraic Manipulation**

- 1) $(\bar{A} + B) \cdot (A + B)$
- 2) $AB + A\bar{B} + \bar{A}B$
- 3) $B(A+C) + A\bar{B} + B\bar{C} + C$

➤ **Exercise Problems: Solving using DeMorgan's Theorem**

- 1) $(\bar{A} + C) \cdot (B + D)$
- 2) $A\bar{B} + C$

➤ Minterm:

- Minterm is a product of all the literal (with or without bar) within the logic system.
- **(OR)** A single variable or the logical product of several variables. The variables may or may not be complemented.
- A variable may appear either in its normal form (X) or in its complement form (\bar{X})
- If a variable **value is 0 then its complemented** otherwise it is in its normal form.
- For example, if you have two variables X & Y, there are four possible combination can be formed with AND operation. Each of these four AND operations represents one of the Boolean expressions terms and is called a Minterm or a standard product.

X	Y	Minterm	Designation
0	0	$\bar{X}\bar{Y}$	m ₀
0	1	$\bar{X}Y$	m ₁
1	0	$X\bar{Y}$	m ₂
1	1	XY	m ₃

- A symbol for each Minterm is also shown in the table and is of the form m_j where j denotes the decimal equivalent of the binary number of the Minterm designated.
- For example, the Minterm $XY\bar{Z}$ whose combination is **1 1 0** can be written as **m₆** as decimal equivalent of **1 1 0** is **6**.
- A Boolean expression may be represented from a given truth table by forming a Minterm for each combination of the variables **which produces as 1 in the function, and then taking the OR (Logical Addition) of all those terms.**
- Assume the truth table

X	Y	Z	Output	Minterm	Designation
0	0	0	0	$\bar{X}\bar{Y}\bar{Z}$	m ₀
0	0	1	0	$\bar{X}\bar{Y}Z$	m ₁
0	1	0	1	$\bar{X}Y\bar{Z}$	m ₂
0	1	1	1	$\bar{X}YZ$	m ₃
1	0	0	0	$X\bar{Y}\bar{Z}$	m ₄
1	0	1	0	$X\bar{Y}Z$	m ₅
1	1	0	1	$XY\bar{Z}$	m ₆
1	1	1	0	XYZ	m ₇

Minterm Results 1

- The Boolean function of truth table is obtained by OR ing (Add) three Minterm i.e. 010 ($\bar{X}Y\bar{Z}$), 011 ($\bar{X}YZ$), 110 ($XY\bar{Z}$). Since each of these Minterm results is 1 (output).

$$f(X, Y, Z) = \bar{X}Y\bar{Z} + \bar{X}YZ + XY\bar{Z} = m_2 + m_3 + m_6$$

- The above Boolean function is the sum of three product terms. This type of expression is known as Sum of Product (SOP) expression.

$$f(X, Y, Z) = \sum(2, 3, 6)$$

- Where f is a Boolean function with three variables (X, Y, Z) and it can be read as function f is sum of 2nd, 3rd, and 6th Minterm.
- Sum of Product (SOP):** A Sum of product expression is a product term or several product terms logically added.

➤ **Find the Minterm designation for the following:**

1) $X\bar{Y}\bar{Z}$

Binary Equivalent = 1 0 0

Decimal Equivalent = $1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$
 $= 4 + 0 + 0$
 $= 4$

So, $X\bar{Y}\bar{Z} = m_4$

2) $A\bar{B}C\bar{D}$

Binary Equivalent = 1 0 1 0

Decimal Equivalent = $1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$
 $= 8 + 0 + 2 + 0$
 $= 10$

So, $A\bar{B}C\bar{D} = m_{10}$

- **What are the fundamental products for each of the input words; ABCD = 0010, ABCD = 110, ABCD = 1110. Write SOP expression.**

Solution: The SOP expression is: $\bar{A}\bar{B}C\bar{D} + AB\bar{C} + ABC\bar{D}$

➤ **Maxterm:**

- Maxterm is a sum of all the literal (with or without bar) within the logic system.
- (OR)** A single variable or the logical sum of several variables. The variables may or may not be complemented.
- A variable may appear either in its normal form (X) or in its complement form (\bar{X})
- If a variable **value is 1 then its complemented** otherwise it is in its normal form.
- For example, if you have two variables X & Y, there are four possible combination can be formed with OR operation. Each of these four OR operations represents one of the Boolean expressions

terms and is called a Minterm or a standard product.

X	Y	Minterm	Designation
0	0	$X + Y$	M_0
0	1	$X + \bar{Y}$	M_1
1	0	$\bar{X} + Y$	M_2
1	1	$\bar{X} + \bar{Y}$	M_3

- A symbol for each Maxterm is also shown in the table and is of the form M_j where j denotes the decimal equivalent of the binary number of the Maxterm designated.
- For example, the Maxterm $X + Y + \bar{Z}$ whose combination is **0 0 1** can be written as M_1 as decimal equivalent of **0 0 1** is **1**.
- A Boolean expression may be represented from a given truth table by forming a Maxterm for each combination of the variables **which produces as 0 in the function, and then taking the AND (Logical Multiplication) of all those terms.**
- Assume the truth table

X	Y	Z	Output	Minterm	Designation
0	0	0	0	$X + Y + Z$	M_0
0	0	1	0	$X + Y + \bar{Z}$	M_1
0	1	0	1	$X + \bar{Y} + Z$	M_2
0	1	1	1	$X + \bar{Y} + \bar{Z}$	M_3
1	0	0	0	$\bar{X} + Y + Z$	M_4
1	0	1	0	$\bar{X} + Y + \bar{Z}$	M_5
1	1	0	1	$\bar{X} + \bar{Y} + Z$	M_6
1	1	1	0	$\bar{X} + \bar{Y} + \bar{Z}$	M_7

Maxterm Results 0

- The Boolean function of truth table is obtained by AND ing (Multiply) five Maxterm i.e. 000, 001, 100, 101, 111. Since each of these Maxterm results is 0 (output).

$$f(X, Y, Z) = (X + Y + Z)(X + Y + \bar{Z})(\bar{X} + Y + Z)(\bar{X} + Y + \bar{Z})(\bar{X} + \bar{Y} + \bar{Z}) = M_0.M_1.M_4.M_5.M_7$$

- The above Boolean function is the product of three sum terms. This type of expression is known as Product of Sum (POS) expression.

$$f(X, Y, Z) = \pi(0, 1, 4, 5, 7)$$

- Where f is a Boolean function with three variables (X, Y, Z) and it can be read as function f is product of 0^{th} , 1^{st} , 4^{th} , 5^{th} and 7^{th} Maxterm.
- **Product of Sum (POS): A product of sum expression is a sum term or several sum terms logically multiplied.**

➤ Canonical Form:

- Boolean expression expressed as sum of Minterms or product of Maxterms are called canonical forms.
- For example, the following expressions are the Minterm canonical form and Maxterm canonical form of two variables X and Y.
 - Minterm Canonical = $f(X, Y) = X + \bar{X}\bar{Y} + \bar{X}Y + X\bar{Y} + XY$
 - Maxterm Canonical = $f(X, Y) = (X + \bar{Y})(\bar{X} + Y)(X + Y)$
- The Minterm canonical expression is the sum of all Minterms. Each Minterm contain all the variables.
- The maxterm canonical expression is the product of all Maxterms. Each Maxterm contain all the variables.

➤ Conversion of SOP into Canonical form:

1) Convert the Boolean function $f(X, Y) = X + X\bar{Y}$ into canonical form.

- **Solution:** The given Boolean function $f(X, Y) = X + X\bar{Y} \rightarrow (i)$
- It has two variables and sum of two Minterms. The first term X is missing one variable. So to make it of two variables it can be multiplied by $(Y + \bar{Y})$, as $(Y + \bar{Y}) = 1$.

$$\text{Therefore, } X = X(Y + \bar{Y}) = XY + X\bar{Y}$$

- Substitute the value of X in (i) we get

$$f(X, Y) = XY + X\bar{Y} + X\bar{Y}$$

- Here, the term $X\bar{Y}$ appear twice, according to theorem $X+X=X$, it is possible to remove one of them.

$$f(X, Y) = XY + X\bar{Y}$$

- Rearrange the Minterm in ascending order

$$f(X, Y) = X\bar{Y} + XY$$

$$= m_2 + m_3$$

$$f(X, Y) = \sum (2, 3)$$

2) Convert the Boolean function $f(X, Y) = X + \bar{Y}Z$ into canonical form.

➤ Conversion of POS into Canonical form:

1) Convert the Boolean function $F(X, Y, Z) = (X + Y)(Y + Z)$ into canonical form.

- **Solution:** The given Boolean function

$$F(X, Y, Z) = (X + Y)(Y + Z) \rightarrow (i)$$

- It has three variables and product of two Maxterms. Each Maxterm is missing one variable.

- The first term can be written as

$$X+Y = (X+Y+Z \cdot \bar{Z}) \quad \text{Since } Z \cdot \bar{Z} = 0$$

- Using distributive law $(X + YZ) = (X + Y) (X + Z)$, we can write

$$X+Y = (X+Y+Z) (X+Y+\bar{Z}) \text{ -----} \rightarrow \text{(ii)}$$

- The Second term can be written as

$$Y + Z = (Y+Z+X \cdot \bar{X})$$

$$Y+Z = (Y+Z+X) (Y+Z+\bar{X}) \text{ -----} \rightarrow \text{(iii)}$$

- Substitute (ii) and (iii) in (i) we get

$$F(X, Y, Z) = (X+Y+Z) (X+Y+\bar{Z}) (Y+Z+X) (Y+Z+\bar{X})$$

- The term $(X+Y+Z)$ appear twice and according to theorem $X \cdot X = X$

$$F(X, Y, Z) = (X+Y+Z) (X+Y+\bar{Z}) (Y+Z+\bar{X})$$

- Rearrange the Maxterm in ascending order.

$$F(X, Y, Z) = (X+Y+Z) (X+Y+\bar{Z}) (\bar{X} + Y + Z)$$

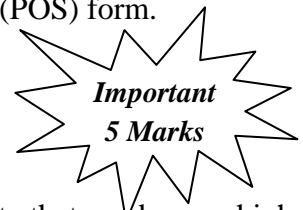
$$= M_0 \cdot M_1 \cdot M_4$$

$$F(X, Y, Z) = \pi (0, 1, 4)$$

- 2) Convert the Boolean function $F(A, B, C) = AB + \bar{A}C$ into product of sum (POS) form.

➤ Karnaugh Map:

- A graphical display of the fundamental products in a truth table.
- Fundamental Product: The logical product of variables and complements that produces a high output for a given input condition.
- The map method provides simple procedure for minimizing the Boolean function.
- The map method was first proposed by E.W. Veitch in 1952 known as “**Veitch Diagram**”.
- In 1953, Maurice Karnaugh proposed “**Karnaugh Map**” also known as “**K-Map**”.



➤ Construction of K-Map:

- The K-Map is a pictorial representation of a truth table made up of squares.
- Each square represents a Minterm or Maxterm.
- A K-Map for **n variables** is made up of **2ⁿ squares**.

➤ Single Variable K-Map:

- A one variable K-Map is shown in the following figure.
- The one variable Boolean expression is of the form $f(A)$.
- There are two Minterms (A and \bar{A}) for one variable.

- Hence the map consists of 2 squares (i.e. 2^n square, $2^1 = 2$ square)

A	\bar{A}	A		
<table border="1" style="margin: auto;"> <tr> <td style="padding: 5px;">\bar{A}</td> <td style="padding: 5px;">A</td> </tr> </table>			\bar{A}	A
\bar{A}	A			
Minterm				

A	0	1		
<table border="1" style="margin: auto;"> <tr> <td style="padding: 5px;">0</td> <td style="padding: 5px;">1</td> </tr> </table>			0	1
0	1			
Basic Labelling				

A	m_0	m_1		
<table border="1" style="margin: auto;"> <tr> <td style="padding: 5px;">m_0</td> <td style="padding: 5px;">m_1</td> </tr> </table>			m_0	m_1
m_0	m_1			

- In one variable K-map:
 - One square represents one Minterm.
 - Two adjacent squares represents a function which is always true i.e. $f(A) = 1$.

➤ **Two Variable K-Map:**

- The two variable Boolean expressions are of the form $f(A, B)$.
- There are four Minterms ($\bar{A}\bar{B}$, $\bar{A}B$, $A\bar{B}$ and AB for two variable).
- Hence the map consists of 4 squares (i.e. 2^n square, $2^2 = 4$ square)

\bar{A}	\bar{B}	B				
<table border="1" style="margin: auto;"> <tr> <td style="padding: 5px;">$\bar{A}\bar{B}$</td> <td style="padding: 5px;">$\bar{A}B$</td> </tr> <tr> <td style="padding: 5px;">$A\bar{B}$</td> <td style="padding: 5px;">AB</td> </tr> </table>			$\bar{A}\bar{B}$	$\bar{A}B$	$A\bar{B}$	AB
$\bar{A}\bar{B}$	$\bar{A}B$					
$A\bar{B}$	AB					
Minterm						

	0	1
0	00	01
1	10	11
Basic Labelling		

\bar{A}	m_0	m_1				
<table border="1" style="margin: auto;"> <tr> <td style="padding: 5px;">m_0</td> <td style="padding: 5px;">m_1</td> </tr> <tr> <td style="padding: 5px;">m_2</td> <td style="padding: 5px;">m_3</td> </tr> </table>			m_0	m_1	m_2	m_3
m_0	m_1					
m_2	m_3					
A						

➤ **Three Variable K-Map:**

- The three variable Boolean expressions are of the form $f(A, B, C)$.
- There are eight Minterms ($\bar{A}\bar{B}\bar{C}$, $\bar{A}\bar{B}C$, $\bar{A}B\bar{C}$, $\bar{A}BC$, $A\bar{B}\bar{C}$, $A\bar{B}C$, $AB\bar{C}$, and ABC) for three variable.
- Hence the map consists of 8 squares (i.e. 2^n square, $2^3 = 8$ square)

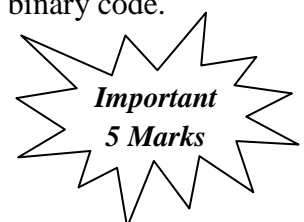
		BC			
		$\bar{B}\bar{C}$	$\bar{B}C$	$B\bar{C}$	BC
A	\bar{A}	$\bar{A}\bar{B}\bar{C}$	$\bar{A}\bar{B}C$	$\bar{A}B\bar{C}$	$\bar{A}BC$
A	A	$A\bar{B}\bar{C}$	$A\bar{B}C$	ABC	$A\bar{B}\bar{C}$
		Minterms			

		BC			
		00	01	11	10
A	0	000	001	011	010
A	1	100	101	111	110
		Basic Labelling			

- Note:** The ordering of variable i.e. 00, 01, 11 & 10 is in gray (reflected binary code) one should not use straight binary code i.e. 00, 01, 10, 11. The straight binary code was used in Veitch diagram but Mr. Karnaugh modified the veitch diagram and use reflected binary code.

➤ **Four Variable K-Map:**

- The four variable Boolean expressions are of the form $f(A, B, C, D)$.
- There are sixteen Minterms for four variables.



- Hence the map consists of 8 squares (i.e. 2^n square, $2^4 = 16$ square).
- The rows and columns are numbered in a reflected code system.

		CD			
		$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
AB	$\bar{A}\bar{B}$	$\bar{A}\bar{B}\bar{C}\bar{D}$	$\bar{A}\bar{B}\bar{C}D$	$\bar{A}\bar{B}C\bar{D}$	$\bar{A}\bar{B}CD$
	$\bar{A}B$	$\bar{A}B\bar{C}\bar{D}$	$\bar{A}B\bar{C}D$	$\bar{A}BC\bar{D}$	$\bar{A}BCD$
	AB	$AB\bar{C}\bar{D}$	$AB\bar{C}D$	$ABC\bar{D}$	$ABCD$
	$A\bar{B}$	$A\bar{B}\bar{C}\bar{D}$	$A\bar{B}\bar{C}D$	$A\bar{B}C\bar{D}$	$A\bar{B}CD$
		Minterms			

		CD			
		00	01	11	10
AB	00	0000	0001	0011	0010
	01	0100	0101	0111	0110
	11	1100	1101	1111	1110
	10	1000	1001	1011	1010
		Basic Labelling			

CHAPTER 2 – BOOLEAN ALGEBRA BLUE PRINT				
VSA (1 marks)	SA (2 marks)	LA (3 Marks)	Essay (5 Marks)	Total
-	02 Question	-	01 Question	03 Question
-	Question no 11, 12	-	Question no 27	09 Marks

Important Questions

➤ **2 Marks Question:**

1. Prove that $X + XY = X$ [March 2015, March 2017]
2. Define Minterm and Maxterm [March 2015, March 2016]
3. State and prove Involution law. [June 2015]
4. State and prove Commutative law using truth table. [June 2016]
5. What is principle of duality? Give Example [June 2015, March 2017]
6. Prove algebraically that $(X+Y)(X+Z)=X+YZ$ [March 2016]
7. Prove: $(X+Y)(X+\bar{Y}) = X$ [June 2016]
8. Prove algebraically that $X+\bar{X}Y=X+Y$
9. Draw a general K-map for four variables A, B, C and D.

➤ **5 Marks Question:**

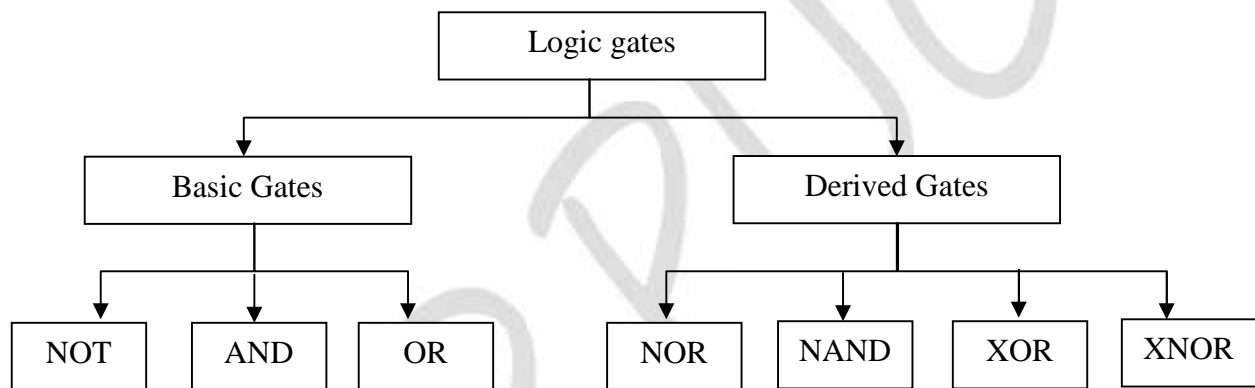
1. Give the Boolean function $F(W, X, Y, Z) = \sum (0, 4, 8, 9, 10, 11, 12, 13, 15)$. Reduce it by using K-Map. [March 2015]
2. Reduce $F(A, B, C, D) = \sum (1, 2, 3, 4, 5, 7, 9, 11, 12, 13, 14, 15)$ using K-Map [June 2015]
3. Using K-Map, Simplify the following expression in four variables
 $F(A, B, C, D) = m_1 + m_2 + m_4 + m_5 + m_9 + m_{11} + m_{12} + m_{13}$. [March 2016]

Chapter-3

LOGIC GATES

➤ Introduction:

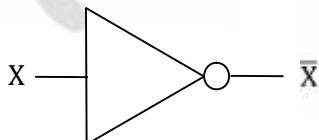
- **Gate:** A Gate is a simply an electronic circuit which operates on one or more input signals and always produces an output signal.
- Gates are digital (two state) circuits because the input and output signals are either low voltage (0) or high voltage (1).
- Gates are often called logic circuits because they can be analyzed with Boolean algebra.
- Gates are classified into two types:



➤ Basic Gates:

➤ NOT Gate:

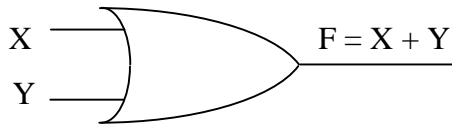
- A NOT gate has only one input and one output.
- The output state is always the opposite of the input state.
- A NOT Gate is also called as **Inverter** gate, because the output is not same as the input.
- The output is sometimes called the complement (opposite) of the input.
- The logical symbol and the truth table of NOT gate are given below.



X	\bar{X}
0	1
1	0

➤ OR Gate:

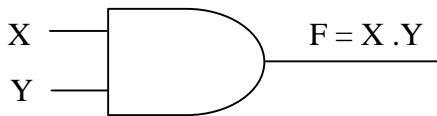
- A OR gate has two or more input signal but only one output signal.
- If any of the input signals is 1 (high), then the output is 1 (high).
- The logical symbol for two-input OR gate and the truth table is given below.



X	Y	F = X+Y
0	0	0
0	1	1
1	0	1
1	1	1

➤ **AND Gate:**

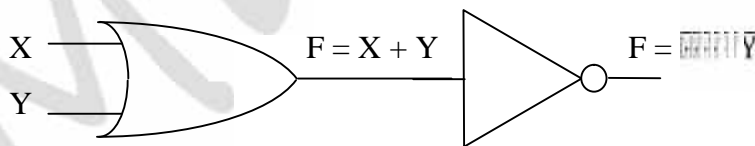
- A AND gate has two or more input signal but only one output signal.
- When all the input signals are 1 (high), the output is 1 (high), otherwise the output is 0.
- The logical symbol for two-input AND gate and the truth table is given below.



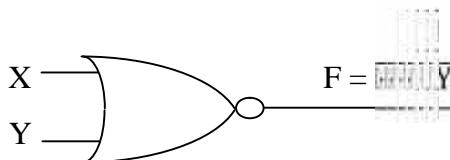
X	Y	F = X.Y
0	0	0
0	1	0
1	0	0
1	1	1

➤ **NOR Gate:**

- A NOR gate has two or more input signal but only one output signal.
- The NOR gate is a complemented of OR gate.
- The output of NOR gate will be 1 only when all inputs are 0 and output will be 0 if any input represents a 1.
- NOR is short form of NOT-OR.
- The symbol \downarrow is used to represent a NOR operation. So $\overline{X+Y}$ can be written as X NOR Y or X \downarrow Y.
- The logical structure shows an OR gate and NOT gate. For input X and Y, the output of the OR gate will be X+Y which is fed as input to the NOT gate. So the output of NOR gate is given by $\overline{X+Y}$ which is equal to $\bar{X} . \bar{Y}$



- The logical symbol for two-input NOR gate and the truth table is given below.

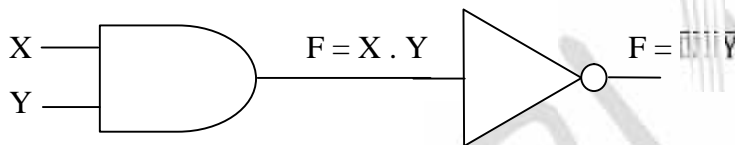


X	Y	F = $\overline{X+Y}$
0	0	1
0	1	0
1	0	0
1	1	0

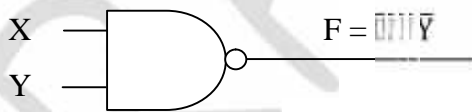
X	Y	Z	F = $\overline{X+Y}$
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

➤ **NAND Gate:**

- A NAND gate has two or more input signal but only one output signal.
- The NAND gate is a complemented of AND gate.
- The output of NAND gate will be 0 only when all inputs are 1 and output will be 0 if any input represents a 0.
- NAND is short form of NOT-AND.
- The symbol \uparrow is used to represent a NOR operation. So $\overline{X \cdot Y}$ can be written as X NAND Y or $X \uparrow Y$.
- Logical structure shows an AND gate and NOT gate. For input X and Y, the output of the OR gate will be $X \cdot Y$ which is fed as input to the NOT gate. So the output of NAND gate is given by $\overline{X \cdot Y}$ which is equal to $\bar{X} + \bar{Y}$



- The logical symbol for two-input NAND gate and the truth table is given below.



X	Y	F = $\overline{X \cdot Y}$
0	0	1
0	1	1
1	0	1
1	1	0

X	Y	Z	F = $\overline{X \cdot Y}$
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

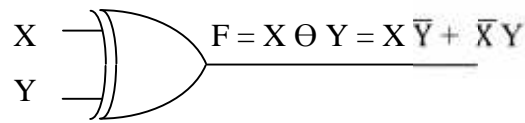
➤ **XOR (Exclusive-OR) Gate:**

- An exclusive-OR has two or more input signal but only one output signal.
- Exclusive-OR gate is different form of OR gate.
- Exclusive-OR gate produces output 1 for only those input combinations that have odd number of 1's.
- The output is 0 if there are even number of 1's in the input.
- The output is 1 if there are odd number of 1's in the input.
- In Boolean algebra, Θ sign stands for XOR operation. Thus X XOR Y can be written as $X \Theta Y$
- If the output is given by:

$$F = X \Theta Y$$

$$F = X \bar{Y} + \bar{X} Y$$

- The XOR gate has a symbol similar to OR gate, except the additional curved line of the input side.



- The following truth table illustrates XOR operation for 2 and 3 inputs.

Number Of 1's	Input		Output
	X	Y	F = X ⊕ Y
EVEN	0	0	0
ODD	0	1	1
ODD	1	0	1
EVEN	1	1	0

Number of 1's	X	Y	Z	F = X ⊕ Y ⊕ Z
EVEN	0	0	0	0
ODD	0	0	1	1
ODD	0	1	0	1
EVEN	0	1	1	0
ODD	1	0	0	1
EVEN	1	0	1	0
EVEN	1	1	0	0
ODD	1	1	1	1

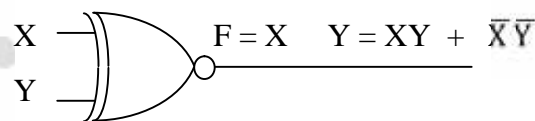
➤ **XNOR (Exclusive-NOR) Gate:**

- The XNOR gate is complement of XOR gate.
- The output of XNOR is 1 only when the logic values of both X and Y is same i.e. either both are equal to 1 or both are 0.
- Its output is 0 when its inputs are different.
- In Boolean algebra, \oplus sign stands for XNOR operation. Thus X XNOR Y can be written as $X \oplus Y$
- If the output is given by:

$$F = X \oplus Y$$

$$F = XY + \bar{X}\bar{Y}$$

- The XNOR gate has a symbol similar to NOR gate, except the additional curved line of the input side.



- The following truth table illustrates XOR operation for 2 and 3 inputs.

Number Of 1's	Input		Output
	X	Y	F = X ⊕ Y
EVEN	0	0	1
ODD	0	1	0
ODD	1	0	0
EVEN	1	1	1

Number of 1's	X	Y	Z	F = X ⊕ Y ⊕ Z
EVEN	0	0	0	1
ODD	0	0	1	0
ODD	0	1	0	0
EVEN	0	1	1	1
ODD	1	0	0	0
EVEN	1	0	1	1
EVEN	1	1	0	1
ODD	1	1	1	0

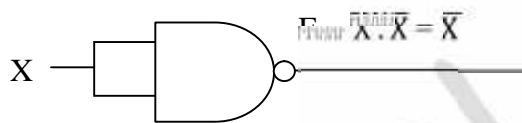
➤ **Universal Gate (NAND & NOR):**

- *Universal gate is a gate using which all the basic gates can be designed.*
- NAND and NOR gate re called as Universal Gates, because all the Boolean functions can also be implemented using these two gates.
- NAND and NOR gates are more popular as these are less expensive and easier to design.

➤ **Realization of all basic gates using NAND gate:**

➤ **NAND to NOT:**

- In the figure we have two input NAND gate whose inputs are purposely connected together so that the same input is applied to both.



- From the diagram $X \text{ NAND } X = \overline{X \cdot X}$
 $= \overline{X} + \overline{X}$ // DeMorgan's 2nd Theorem
 $= \overline{X}$ $\overline{X} + \overline{X} = \overline{X}$
 $= \text{Inverted Input} = \text{NOT gate}$

➤ **NAND to AND:**

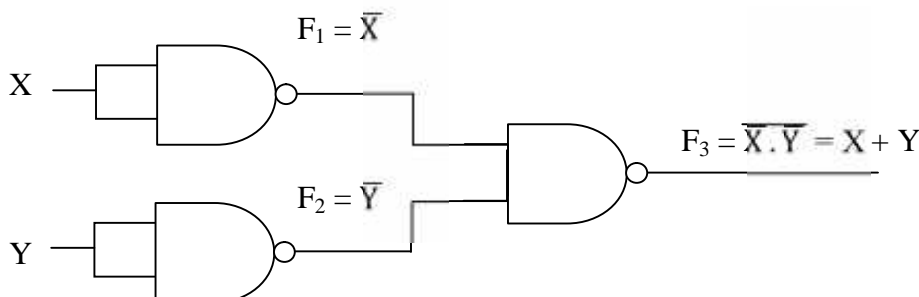
- In the figure we have two NAND gates connected so that the AND operation is performed.
- NAND gate 2 is used as a NOT gate.



- From the diagram $X \text{ NAND } Y = F1 = \overline{X \cdot Y}$
 $F2 = \overline{F1 \cdot F1}$
 $= \overline{F1} + \overline{F1}$ // DeMorgan's 2nd Theorem
 $= \overline{\overline{X \cdot Y}}$ $\overline{\overline{X \cdot Y}} = X \cdot Y$
 $F2 = X \cdot Y$
 $= \text{AND gate}$

➤ **NAND to OR:**

- The OR operation can be implemented using NAND gates connected as shown in figure.
- NAND gate 1 and NAND gate 2 are used as NOT to invert the inputs.



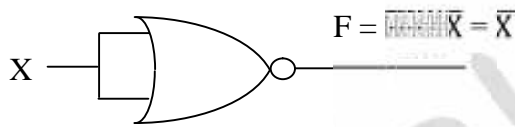
- From the diagram X NAND Y

$$\begin{aligned}
 F_1 &= \overline{X \cdot Y} \\
 F_2 &= \overline{X} + \overline{Y} \\
 F_3 &= \overline{F_1 \cdot F_2} \\
 &= \overline{\overline{X} + \overline{Y}} \quad // \text{DeMorgan's 2}^{nd} \text{ Theorem} \\
 &= \overline{\overline{X}} \cdot \overline{\overline{Y}} \quad \overline{\overline{X}} = X \text{ and } \overline{\overline{Y}} = Y \\
 F_3 &= X \cdot Y \\
 &= \text{OR gate}
 \end{aligned}$$

➤ Realization of all basic gates using NOR gate:

➤ NOR to NOT:

- Figure shows that NOR gate with its inputs connected together behaves as a NOT gate.

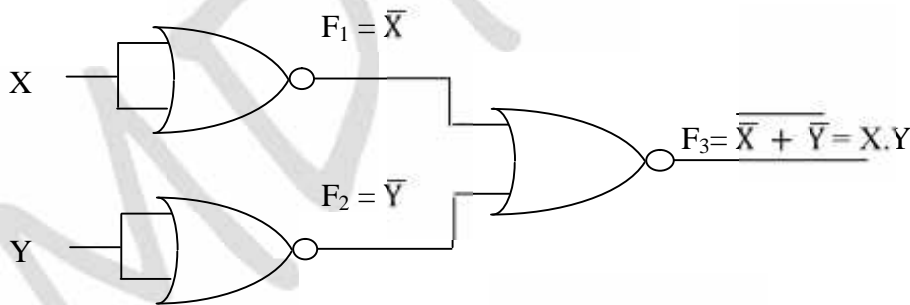


- From the diagram X NOR X

$$\begin{aligned}
 &= \overline{X + X} \\
 &= \overline{X \cdot X} \quad // \text{DeMorgan's 1}^{st} \text{ Theorem} \\
 &= \overline{X} \quad \overline{X \cdot X} = \overline{X} \\
 &= \text{Inverted Input} = \text{NOT gate}
 \end{aligned}$$

➤ NOR to AND:

- The AND operation can be implemented with NOR gate as shown in figure. Here NOR gate 1 and NOR gate 2 are used as NOT gate to invert inputs.

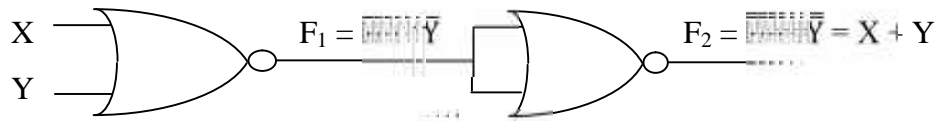


- From the diagram X NOR Y

$$\begin{aligned}
 F_1 &= \overline{X \cdot Y} \\
 F_2 &= \overline{X + Y} \\
 F_3 &= \overline{F_1 + F_2} \\
 &= \overline{\overline{X \cdot Y} + \overline{X + Y}} \quad // \text{DeMorgan's 1}^{st} \text{ Theorem} \\
 &= \overline{\overline{X} \cdot \overline{Y}} \quad \overline{\overline{X}} = X \text{ and } \overline{\overline{Y}} = Y \\
 F_3 &= X \cdot Y \\
 &= \text{AND gate}
 \end{aligned}$$

➤ **NAND to OR:**

- In the figure two NOR gates are arranged so that the OR operation is performed.
- NOR gate 2 is used as NOT gate.



- From the diagram $X \text{ NOR } Y$

$$\begin{aligned}
 F_1 &= (X+Y)' \\
 F_2 &= ((X+Y)')' \\
 F_2 &= (F_1)' \quad // \text{DeMorgan's 1}^{\text{st}} \text{ Theorem} \\
 &= \overline{\overline{X+Y}} \\
 &= X+Y \quad \overline{\overline{X}} = X \\
 F_2 &= X+Y \\
 &= \text{OR gate}
 \end{aligned}$$

➤ **Designing of Logic Circuit using all basic gates :**

➤ **Designing of Logic Circuit using NAND and NOR gates:**

CHAPTER – LOGIC GATES BLUE PRINT			
VSA (1 marks)	LA (3 Marks)	-	Total
01 Question	01 Question	-	02 Questions
Question No 1	Question No 20	-	04 Marks

Chapter-4

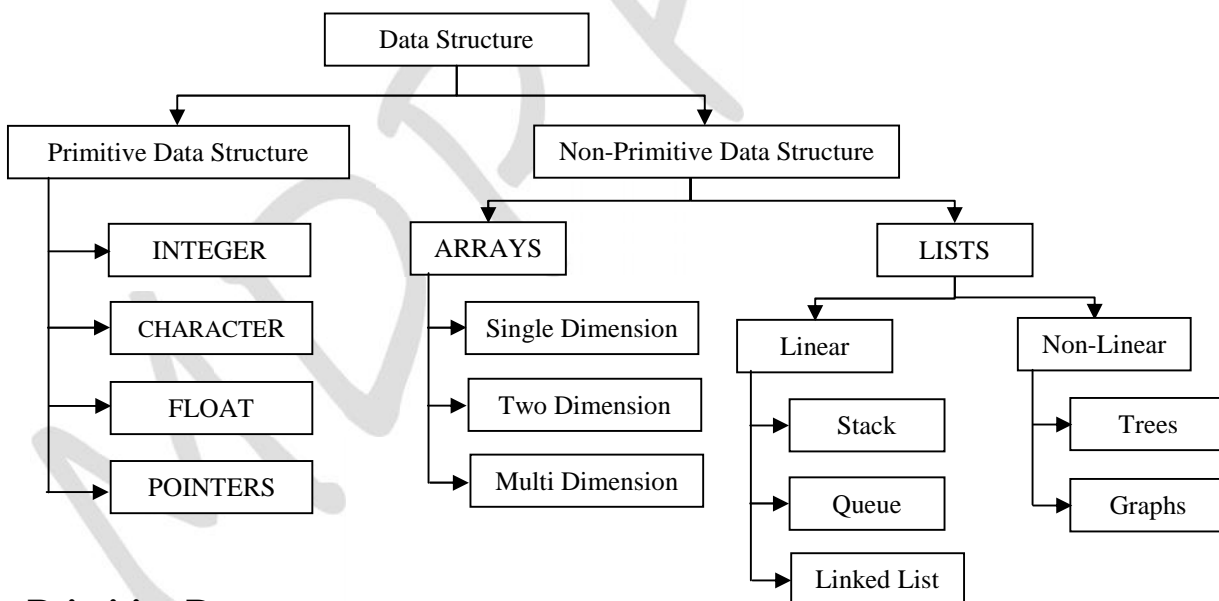
DATA STRUCTURES

➤ Introduction:

- *Data Structure is the way of collecting and organizing the data in such a way that we can perform operation on these data in an effective way.*
- For Example: We have the data student's Name "Akash" and age "16". Here "Akash" is a string type data type and 16 are of integer. We can now organize this data as a record like a Student Record. We can collect and store other **Student's Record** in a file or a database as a data structure.

➤ Data Representation:

- Computer memory is used to store the data which is required for processing. This process is known as data representation.
- Data may be of same type or different type.
- Data Structure provides efficient way of combining these data types and process data.



➤ Primitive Data structures:

- *Data structures that are directly operated upon the machine-level instructions are known as primitive data structures.*
- The integers, float, character data, pointers are primitive data structures.

➤ Operations on Primitive Data structures:

- **Create:** Create operation is used to create a new data structure. This operation reserves memory space for the program elements. It may be carried out at compile time and run time.

Example: int n=15; // memory spaced to be created for n.

- **Select:** This operation is used programmers to access the data within the data structure. This operation updates or alters data.

Example: cin>>x;

- **Update:** This operation is used to change data of data structures. An assignment operation is a good example.

Example: int n = 5; //modifies the value of n to store the new value 5 in it.

- **Destroy:** This operation is used to destroy or remove the data structure from the memory space. In C++ one can destroy data structure by using the function called *delete*.

➤ **Non-Primitive Data structures:**

- *The Data structures that are derived from the primitive data structures are called Non-primitive data structure.*
- These data structures are used to store group of values.
- Non-Primitive data structures are classified as arrays, lists and files.
- Data Structures under lists are classified as linear and non-linear data structure.

i. **Linear Data structures:**

- *Linear Data structures are kind of data structure that has homogeneous elements.*
- The data structure in which elements are in a sequence and form a liner series.
- Linear data structures are very easy to implement, since the memory of the computer is also organized in a linear fashion.
- Some commonly used linear data structures are **Stack, Queue and Linked Lists**.

ii. **Non-Linear Data structures:**

- *A Non-Linear Data structures is a data structure in which data item is connected to several other data items.*
- Non-Linear data structure may exhibit either a hierarchical relationship or parent child relationship.
- The data elements are not arranged in a sequential structure.
- The different non-linear data structures are **trees and graphs**.

➤ **Operations on Non-Primitive Data structures:**

1. **Traversing:** The processing of accessing each element exactly once to perform some operation is called *traversing*.
2. **Insertion:** The process of adding a new element into the given collection of data elements is called *insertion*.

3. **Deletion:** The process of removing an existing data element from the given collection of data elements is called *deletion*.
4. **Searching:** The process of finding the location of a data element in the given collection of data elements is called as *searching*.
5. **Sorting:** The process of arrangement of data elements in ascending or descending order is called *sorting*.
6. **Merging:** The process of combining the elements of two structures to form a single structure is called *merging*.

➤ Arrays:

- *An array is an ordered collection of elements of same data type that share common name.*
- The elements are arranged one after the other in adjacent memory location.
- These elements are accessed by numbers called subscripts or indices.
- The elements are accessed using subscripts; arrays are also called as subscripted variables.
- There are three types of arrays
 - i. One-dimensional Array
 - ii. Two-dimensional Array
 - iii. Multi-dimensional Array

➤ One dimensional array:

- *An array with only one row or column is called one-dimensional array.*
- It is finite collection of n number of elements of same type such that:
 - Elements are stored in continuous locations.
 - Elements can be referred by indexing.
- The syntax to define one-dimensional array is:

Syntax: Datatype Array_Name [Size];

- Where,
 - Datatype : Type of value it can store (Example: int, char, float)
 - Array_Name: To identify the array.
 - Size : The maximum number of elements that the array can hold.
- **Example:** int a[5];

Address	Content	Location
1000	A	A[0]
1001	B	A[1]

Lower Bound(LB)

1002	C	A[2]
1003	D	A[3]
1004	E	A[4]

Upper Bound(UB)

- **Calculating the length of the array:**

- A[n], the number of n is called the size or length of the array.
- The length of the array can be calculated by:

$$L = UB - LB + 1$$

- Here, UB is the largest Index and LB is the smallest index
- Example: If an array A has values 10, 20, 30, 40, 50, stored in location 0,1, 2, 3, 4 the UB = 4 and LB=0
- Size of the array $L = 4 - 0 + 1 = 5$

- **Basic operations on One-dimensional array:**



1. **Traversing** : Processing each element in the array.
2. **Insertion** : Inserting a new element into the array.
3. **Deletion** : Removing an element from the array.
4. **Searching** : Finding the location of an element in the array.
5. **Sorting** : Arranging the elements of the array in some order.
6. **Merging** : Combining one or more array to form a single array.

- **Traversing an array:**

- *Traversing is the process of visiting each subscript at least once from the beginning element to last element.*
- For example, to find the maximum element of the array we need to access each element of the array.

ALGORITHM: Traversal (A, LB, UB) A is an array with Lower Bound LB and Upper Bound UB.

```

Step 1:      for LOC = LB to UB do
Step 2:          PROCESS A [LOC]
                [End of for loop]
Step 3:      Exit

```

➤ Inserting an element into an array:

- *Insertion refers to inserting an element into the array.*
- Based on the requirement, new element can be added at the beginning, end or any given position of array.
- When an element is to be inserted into a particular position, all the elements from the asked position to the last element should be shifted into higher order position.
- For example: Let A[5] be an array with items 10, 20, 40, 50, 60 stored at consecutive locations. Suppose item 30 has to be inserted at position 2. The following procedure is applied.
 - Move Number 60 to the position 5.
 - Move Number 50 to the position 4.
 - Move Number 40 to the position 3.
 - Position 2 is blank. Insert 30 into the position 2 i.e. A[2]=30.

A[0]	10
A[1]	20
A[2]	40
A[3]	50
A[4]	60
A[5]	

A[0]	10
A[1]	20
A[2]	40
A[3]	50
A[4]	60
A[5]	

A[0]	10
A[1]	20
A[2]	40
A[3]	50
A[4]	
A[5]	60

A[0]	10
A[1]	20
A[2]	40
A[3]	
A[4]	50
A[5]	60

A[0]	10
A[1]	20
A[2]	30
A[3]	40
A[4]	50
A[5]	60

ALGORITHM: Insert (A, N, ITEM, Pos) A is an array with N elements. ITEM is the element to be inserted in the position Pos.

Step 1: for I = N-1 down to Pos

A[I + 1] = A[I]

[End of for loop]

Step 2: A [Pos] = ITEM

Step 3: N = N+1

Step 4: Exit

➤ Deleting an element into an array:

- *Deletion refers to deleting an element from the array.*
- Based on the requirement, element can be deleted at the beginning, end or any given position of array.
- When an element is to be deleted from a particular position, all the subsequent shifted into lower order position.

- For example: Let A[4] be an array with items 10, 20, 30, 40, 50 stored at consecutive locations. Suppose item 30 has to be deleted at position 2. The following procedure is applied.
 - Copy 30 to ITEM, i.e. Item = 30.
 - Move Number 40 to the position 2.
 - Move Number 50 to the position 3.

A[0]	10
A[1]	20
A[2]	30
A[3]	40
A[4]	50

A[0]	10
A[1]	20
A[2]	
A[3]	40
A[4]	50

A[0]	10
A[1]	20
A[2]	40
A[3]	
A[4]	50

A[0]	10
A[1]	20
A[2]	40
A[3]	50
A[4]	

ALGORITHM: Delete (A, N, ITEM, Pos) A is an array with N elements. ITEM is the element to be deleted in the position Pos and it is stored into variable Item.

Step 1: ITEM = A [Pos]
 Step 2: for I = Pos down to N-1
 A[I] = A[I+1]
 [End of for loop]
 Step 3: N = N-1
 Step 4: Exit

➤ Searching an element in an array:

- *Searching refers to finding the location of the element in a linear array.*
- There are different algorithms, but the most common methods are linear search and binary search.

➤ Linear Search:

- This is the simplest method in which the element to be searched is compared with each element of the array by one from the beginning to end of the array.
- Searching is one after the other, it is also called as *sequential search*.

ALGORITHM: Linear_Search (A, N, Element) A is an array with N elements. Element is the being searched in the array.

Step 1: LOC = -1 [Assume the element does not exist]
 Step 2: for I = 0 to N-1 do
 if (Element = A [I]) then

	LOC = I
	Goto Step 3
	[End if]
	[End of for loop]
Step 3:	if (LOC >= 0) then
	Print Element, "Found in Location", LOC
	else
	Print Element, "Not Found"
Step 4:	Exit

- **Example:** Consider the following elements stored in an array and we are searching for the element 17. The trace of the algorithm is given below.

					Index (I)	Compare 17	Location
					I = 0	17 = 12 (Does not match)	LOC = -1
A[0]	A[1]	A[2]	A[3]	A[4]	I = 1	17 = 23 (Does not match)	LOC = -1
12	23	9	17	7	I = 2	17 = 9 (Does not match)	LOC = -1
					I = 3	17 = 17 (Matches)	LOC = 3

➤ Binary Search:

- When the elements of the array are in sorted order, the best method of searching is binary search.
- This method compares the element to be searched with the middle element of the array.
- If the comparison is searched either at the right half of the array or at the left half of the array.
- Let the position of first (BEG) and last (END) elements of the array. The middle element A[MID] can be obtained by find the middle location MID by $MID = (BEG + END) / 2$.
- If $A[MID] = ELE$, then search is successful. Otherwise a new segment is found as follows:
 - If $ELE < A[MID]$, searching is continued at left half of the array. Reset $END = MID - 1$.
 - If $ELE > A[MID]$, searching is continued at right half of the array. Reset $BEG = MID + 1$.
 - If ELE not found then we get a condition $BEG > END$. This results in unsuccessful search.

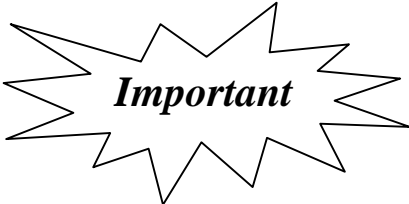
ALGORITHM: Binary_Search (BEG, END, MID ELE) A is an array with N elements. Let BEG, END, MID denote Beginning, end and middle location of the array

Step 1: Set $BEG = LB$, $END = UB$ LOC = -1

Step 2: While($BEG \leq END$)

```

MID = (BEG+END)/2
if ( ELE = A [ MID ] ) then
    LOC = MID
    Goto Step 3
else
    if( ELE < A[MID])
        END=MID-1;
    else
        BEG=MID+1;
    [End if]
[End if]
[End of While loop]
Step 3: if ( LOC >= 0 ) then
        Print Element, "Found in Location", LOC
    else
        Print Element, "Search is Unsuccessful"
Step 4: Exit
    
```



- Example:** Consider the following elements stored in an array and we are searching for the element 47. The trace of the algorithm is given below.

					BEG & END	MID = (BEG+END)/2	Compare	Location
A[0]	A[1]	A[2]	A[3]	A[4]	BEG = 0 END = 4	MID = (0+4)/2 MID = 2	47>39 (Does not match)	LOC = -1
12	23	39	47	57				
BEG MID END								
The search element i.e. 47 is greater than the element in the middle position i.e. 39 then continues the search to the right portion of the middle element.								
A[0]	A[1]	A[2]	A[3]	A[4]	BEG = 3 END = 4	MID = (3+4)/2 MID = 3	47 = 47	LOC = 3
12	23	39	47	57				
BEGMID END								
Since the element in the middle position is the same as the search element LOC gets the value 3. Since the value of LOC is greater than 0, we get the output as 47 Found in location 3								

- **Example:** Consider the following elements stored in an array and we are searching for the element 67. The trace of the algorithm is given below.

					BEG & END	MID = (BEG+END)/2	Compare	Location
A[0]	A[1]	A[2]	A[3]	A[4]	BEG = 0 END = 4	MID = (0+4)/2 MID = 2	67>39 (Does not match)	LOC = -1
12	23	39	47	57				
BEG MID END								
The search element i.e. 67 is greater than the element in the middle position i.e. 39 then continues the search to the right portion of the middle element.								
A[0]	A[1]	A[2]	A[3]	A[4]	BEG = 3 END = 4	MID = (3+4)/2 MID = 3	67>47 (Does not match)	LOC = -1
12	23	39	47	57				
BEGMID END								
The search element i.e. 67 is greater than the element in the middle position i.e. 47 then continues the search to the right portion of the middle element.								
A[0]	A[1]	A[2]	A[3]	A[4]	BEG = 4 END = 4	MID = (4+4)/2 MID = 4	67>57 (Does not match)	LOC = -1
12	23	39	47	57				
BEGMID END								
The search element i.e. 67 is greater than the element in the middle position i.e. 57 then continues the search to the right portion of the middle element.								
A[0]	A[1]	A[2]	A[3]	A[4]	BEG = 5 END = 4	Since the condition (BEG <= END) is false the comparison ends		
12	23	39	47	57				
ENDBEG								
We get the output as 67 Not Found								

➤ **Difference between Linear Search and Binary Search**

	Linear Search	Binary Search
1	This can be used in sorted and unsorted array	This can be used only in sorted array
2	Array elements are accessed sequentially	One must have direct access to the middle element in the sub list.
3	Access is very slow	Access is faster.
4	This can be used in single and multi dimensional array	Used only in single dimensional array.
5	This technique is easy and simple in implementing	Complex in operation

➤ Sorting the elements in an array:

- *Sorting is the arrangement of elements of the array in some order.*
- There are different sorting methods like Bubble Sort, Selection Sort, Shell Sort, Quick sort, Heap Sort, Insertion Sort etc.

➤ Insertion Sort:

- In Insertion sort, the first element of the array is assumed to be in the correct position next element is considered as the key element and compared with the elements before the key element and is inserted in its correct position.
- **Example:** Consider the following array contains 8 elements as follows:

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]
45	26	23	56	29	36	12	4

Pass	Location	A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]
I=1	J=0	45	26	23	56	29	36	12	4
I=2	J=0	26	45	23	56	29	36	12	4
I=3	J=3	23	26	45	56	29	36	12	4
I=4	J=2	23	26	45	56	29	36	12	4
I=5	J=3	23	26	29	45	56	36	12	4
I=6	J=0	23	26	29	36	45	56	12	4
I=7	J=0	12	23	26	29	36	45	56	4
Sorted List		4	12	23	26	29	36	45	56

ALGORITHM: Insertion_Sort (A, N) A is an array with N unsorted elements.

```

Step 1:   for I=1 to N-1
Step 2:   J = I
           While(J >= 1)
             if ( A[J] < A[J-1] ) then
               Temp = A[J];
               A[J] = A[J-1];
               A[J-1] = Temp;
             [End if]
           J = J-1
         [End of While loop]
       [End of For loop]
Step 3:   Exit
  
```


➤ Two dimensional array:

- A two dimensional array is a collection of elements and each element is identified by a pair of subscripts. (A[5] [5])
- The elements are stored in continuous memory locations.
- The elements of two-dimensional array as rows and columns.
- The number of rows and columns in a matrix is called as the order of the matrix and denoted as mxn.
- The number of elements can be obtained by multiplying number of rows and number of columns.

	[0]	[1]	[2]
A[0]	1	2	3
A[1]	7	8	9
A[2]	4	5	6

➤ Representation of Two Dimensional Array:

- A is the array of order m x n. To store m*n number of elements, we need m*n memory locations. The elements should be in contiguous memory locations.
- There are two methods:
 - Row-major method
 - Column-major method
- **Row-Major Method:** All the first-row elements are stored in sequential memory locations and then all the second-row elements are stored and so on.

1001	1	A[0][0]
1002	7	A[0][1]
1003	4	A[0][2]
1004	2	A[1][0]
1005	8	A[1][1]
1006	5	A[1][2]
1007	4	A[2][0]
1008	5	A[2][1]
1009	6	A[2][2]

Row-Major Method

1001	1	A[0][0]
1002	2	A[1][0]
1003	3	A[2][0]
1004	7	A[0][1]
1005	8	A[1][1]
1006	9	A[2][1]
1007	3	A[0][2]
1008	9	A[1][2]
1009	6	A[2][2]

Column Major Method

- **Column-Major Method:** All the first column elements are stored in sequential memory locations and then all the second-column elements are stored and so on.

➤ Advantages of Array:

- It is used to represent multiple data items of same type by using single name.
- It can be used to implement other data structures like linked lists, stacks, queues, tree, graphs etc.
- Two-dimensional arrays are used to represent matrices.
- Many databases include one-dimensional arrays whose elements are records.

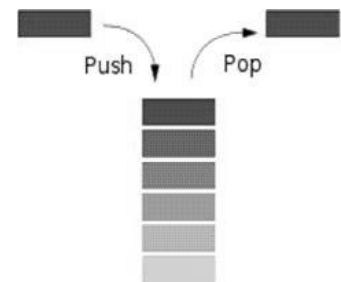
➤ Disadvantages of Array:

- We must know in advance the how many elements are to be stored in array.
- Array is static structure. It means that array is of fixed size. The memory which is allocated to array cannot be increased or decreased.
- Array is fixed size; if we allocate more memory than requirement then the memory space will be wasted.
- The elements of array are stored in consecutive memory locations. So insertion and deletion are very difficult and time consuming.

➤ STACKS:

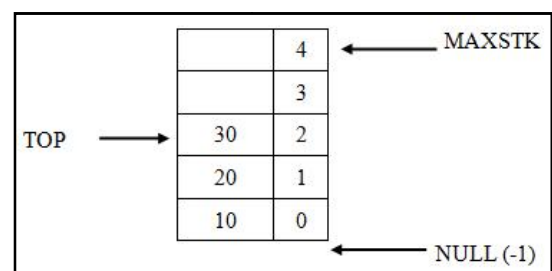
- *A stack is an ordered collection of items in which an element may be inserted or deleted only at same end.*

- This end is called as the **top** of the stack.
- The end opposite to top is known as the **base**.
- Stacks are sometimes known as **LIFO** (Last In First Out).



➤ Representation of Stacks using Array:

- Stack can be represented using a one-dimensional array.
- The items into the stack are stored in a sequential order from the first location of the memory block.
- A pointer TOP contains the location of the top element of the stack.
- A variable MAXSTK contains the maximum number of element that can be stored in the stack.
- If we attempt to add new element beyond the maximum size, we will encounter a stack **overflow condition**. Similarly, you cannot remove elements beyond the base of the stack. If such is the case, we will reach a stack **underflow condition**.
- The condition **TOP = MAXSTX** indicates that the stack is full and **TOP = NULL** indicates that stack is empty.



➤ Operation on Stacks:

- **Stack()**: It creates a new stack that is empty. It needs no parameter and returns an empty stack.
- **push(item)**: It adds a new item to the top of the stack.
- **pop()**: It removes the top item from the stack.
- **peek()**: It returns the top item from the stack but does not remove it.
- **isEmpty()**: It tests whether the stack is empty.
- **size()**: It returns the number of items on the stack.



➤ PUSH Operation:

- *The process of adding one element or item to the stack is represented by an operation called as the PUSH operation.*
- The new element is added at the topmost position of the stack.

ALGORITHM: PUSH (STACK, TOP, ITEM) STACK is the array with N elements. TOP is the pointer to the top of the element of the array. ITEM to be inserted.

```

Step 1:      if TOP = N-1 then                [Check Overflow]
              PRINT " STACK is Full or Overflow"
              Exit
              [End if]
Step 2:      TOP = TOP + 1                    [Increment the TOP]
Step 3:      STACK[TOP] = ITEM               [Insert the ITEM]
Step 4:      Return
  
```

➤ POP Operation:

- *The process of deleting one element or item from the stack is represented by an operation called as the POP operation.*

ALGORITHM: POP (STACK, TOP, ITEM) STACK is the array with N elements. TOP is the pointer to the top of the element of the array. ITEM to be inserted.

```

Step 1:      if TOP = NULL then              [Check Underflow]
              PRINT " STACK is Empty or Underflow"
              Exit
              [End if]
Step 2:      ITEM = STACK[TOP]              [copy the TOP Element]
Step 3:      TOP = TOP - 1                  [Decrement the TOP]
Step 4:      Return
  
```

➤ Application of Stacks:

- It is used to reverse a word. You push a given word to stack – letter by letter and then pop letter from the stack.
- “Undo” mechanism in text editor.
- Backtracking: This is a process when you need to access the most recent data element in a series of elements. Once you reach a dead end, you must backtrack.
- Language Processing: Compiler’ syntax check for matching braces is implemented by using stack.
- Conversion of decimal number to binary.
- Conversion of infix expression into prefix and postfix.
- Quick sort
- Runtime memory management.

➤ Arithmetic Expression:

- An expression is a combination of operands and operators that after evaluation results in a single value.
- Operand consists of constants and variables.
- Operators consists of { , +, -, *, /,),] etc.
- Expression can be
 - Infix Expression
 - Postfix Expression
 - Prefix Expression
- **Infix Expression:** If an operator is in between two operands, it is called infix expression.
Example: $a + b$, where a and b are operands and $+$ is an operator.
- **Postfix Expression:** If an operator follows the two operands, it is called postfix expression.
Example: $ab +$
- **Prefix Expression:** an operator precedes the two operands, it is called prefix expression.
Example: $+ab$

➤ Algorithm for Infix to Postfix:

1. Examine the next element in the input.
2. If it is operand, output it.
3. If it is opening parenthesis, push it on stack.
4. If it is an operator, then
 - If stack is empty, push operator on stack.

- If the top of the stack is opening parenthesis, push operator on stack.
 - If it has higher priority than the top of stack, push operator on stack.
 - Else pop the operator from the stack and output it, repeat step 4.
5. If it is a closing parenthesis, pop operators from the stack and output them until an opening parenthesis is encountered. Pop and discard the opening parenthesis.
 6. If there is more input go to step 1.
 7. If there is no more input, pop the remaining operators to output.
- Example: Suppose we want to convert $2*3/(2-1)+5*3$ into postfix form.

Expression	Stack	Output
2	Empty	2
*	*	2
3	*	23
/	/	23*
(/(23*
2	/(23*2
-	/(-	23*2
1	/(-	23*21
)	/	23*21-
+	+	23*21-/+
5	+	23*21-/+5
*	+*	23*21-/+5*
3	+*	23*21-/+5*3
	Empty	23*21-/+5*3+

- Consider the following examples of converting from infix to postfix

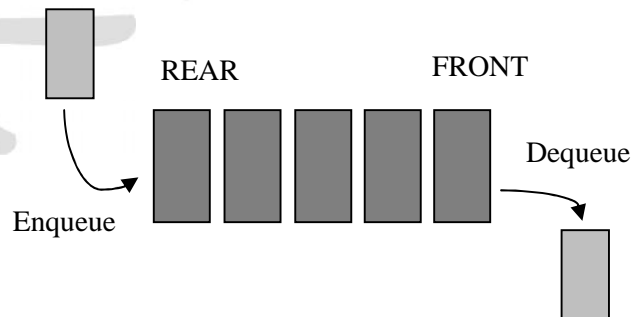
Sl No	Infix		Postfix
1	$(A + B) * C$	$= [AB+] * C$	$AB+C*$
2	$A + B + C$	$= [AB+] + C$	$AB+C+$
3	$(A + B) / (X - Y)$	$= [AB+] / [XY-]$	$AB+XY-/+$
4	$A ^ B * C - D$	$= [AB^] * C - D$ $= AB^C * - D$	$AB^C * D -$
5	$((A + B) * C - (D - E)) ^ (X + Y)$	$= ([AB+] * C - [DE-]) ^ [XY+]$ $= ([AB+C*] - [DE-]) ^ [XY+]$ $= (AB+C*DE--)^ [XY+]$	$AB+C*DE--XY+^$

- Consider the following examples of converting from infix to prefix.

Sl No	Infix		Prefix
1	$(A + B) * C$	$= [+AB] * C$	$*+ABC$
2	$A + B + C$	$= [+AB]+C$	$++ABC$
3	$(A + B) / (X - Y)$	$= [+AB]/[-XY]$	$/+AB-XY$
4	$A ^ B * C - D$	$= [^AB] *C-D$ $= [*^ABC]-D$	$-*^ABCD$
5	$((A + B)*C-(D-E))^(X+Y)$	$= ([+AB]*C-[-DE])^[+XY]$ $= ([*+ABC]-[-DE])^[+XY]$ $= (-*+ABC-DE)^[+XY]$	$^-*+ABC-DE^[+XY]$

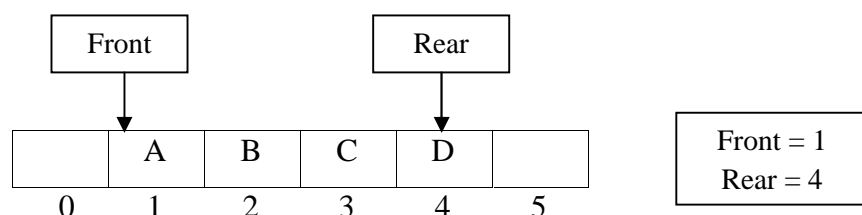
➤ **QUEUES:**

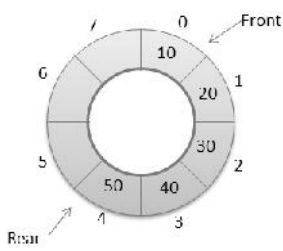
- A queue is an ordered collection of items where an item is inserted at one end called the “rear” and an existing item is removed at the other end, called the “front”.
- Queue is also called as **FIFO** list i.e. **First-In First-Out**.
- In the queue only two operations are allowed enqueue and dequeue.
- Enqueue means to insert an item into back of the queue.
- Dequeue means removing the front item.



➤ **Types of Queues:**

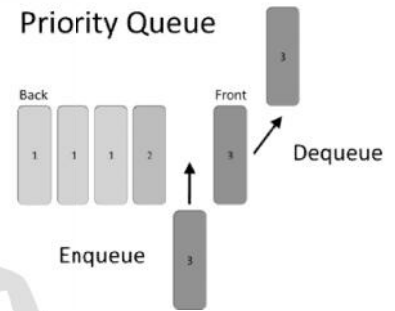
- Queue can be of four types:
 - Simple Queue
 - Circular Queue
 - Priority Queue
 - Dequeue (Double Ended Queue)
- Simple Queue:** In simple queue insertion occurs at the rear end of the list and deletion occurs at the front end of the list.



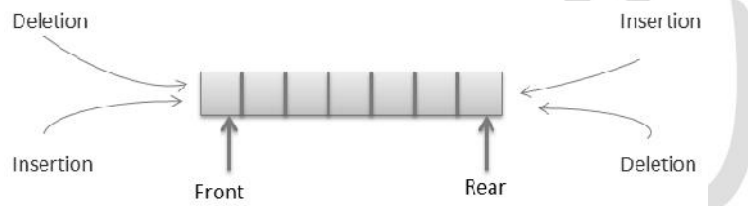


- **Circular Queue:** A circular queue is a queue in which all nodes are treated as circular such that the last node follows the first node.

- **Priority Queue:** A priority queue is a queue that contains items that have some present priority. An element can be inserted or removed from any position depending upon some priority.



- **Dequeue:** It is a queue in which insertion and deletion takes place at the both ends.

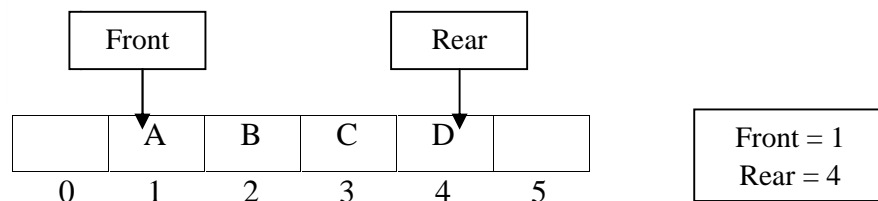


➤ **Operation on Queues:**

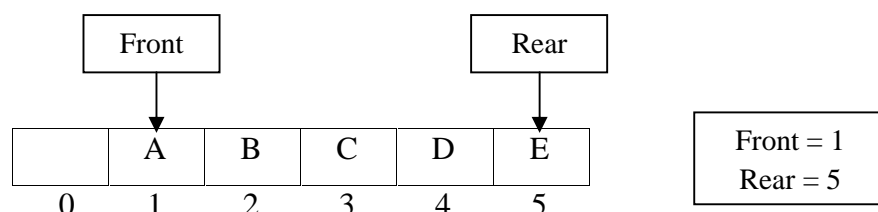
- **Queue():** It creates a new queue that is empty.
- **enqueue(item):** It adds a new item to the rear of the queue.
- **dequeue():** It removes the front item from the queue.
- **isEmpty():** It tests to see whether the queue is empty.
- **size():** It returns the number of items in the queue.

➤ **Memory Representation of a queue using array:**

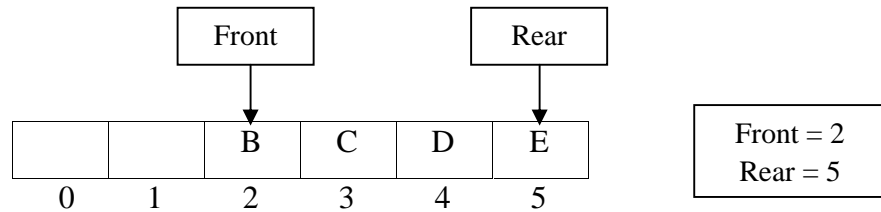
- Queue is represented in memory using linear array.
- Let QUEUE is a array, two pointer variables called FRONT and REAR are maintained.
- The pointer variable FRONT contains the location of the element to be removed or deleted.
- The pointer variable REAR contains location of the last element inserted.
- The condition FRONT = NULL indicates that queue is empty.
- The condition REAR = N-1 indicates that queue is full.



REAR = REAR + 1, QUEUE [REAR] = 'E'



$$\text{ITEM} = \text{QUEUE}[\text{FRONT}], \text{FRONT} = \text{FRONT} + 1$$



➤ **Queue Insertion Operation (ENQUEUE):**

- Consider a queue Q with 5 elements.

Q[0]	Q[1]	Q[2]	Q[3]	Q[4]

Initially Queue is Empty FRONT = -1, REAR = -1

- If we want to add one element i.e. 8 in the queue, then FRONT and REAR Indicator are set to 0 and the element 8 would be stored at the position pointed by the REAR.

REAR = 0, FRONT = 0, Q[REAR]=Q[0]=8

Q[0]	Q[1]	Q[2]	Q[3]	Q[4]
8				

- If we want to add one more element i.e. 12 in the queue, then REAR Indicator would be incremented by 1 and the element 12 would be stored at the position pointed by the REAR.

REAR = 1, FRONT = 0, Q[REAR]=Q[1]=12

Q[0]	Q[1]	Q[2]	Q[3]	Q[4]
8	12			

- Repeat this procedure three more times to insert 4, 18 and 34 elements.

REAR = 4, FRONT = 0, Queue is full.

Q[0]	Q[1]	Q[2]	Q[3]	Q[4]
8	12	4	18	34

➤ **Algorithm for Insertion:**

ALGORITHM: ENQUEUE (QUEUE, REAR, FRONT, ITEM) QUEUE is the array with N elements. FRONT is the pointer that contains the location of the element to be deleted and REAR contains the location of the inserted element. ITEM is the element to be inserted.

```

Step 1:      if REAR = N-1 then                [Check Overflow]
              PRINT "QUEUE is Full or Overflow"
              Exit
              [End if]
Step 2:      if FRONT = NULL then             [Check Whether Queue is empty]
              FRONT = 0
              REAR = 0
              else
                REAR = REAR + 1               [Increment REAR Pointer]
Step 3:      QUEUE[REAR] = ITEM              [Copy ITEM to REAR position]
Step 4:      Return
    
```


➤ **Queue Deletion Operation (DEQUEUE):**

- Consider a queue Q with 3 elements.

Q[0]	Q[1]	Q[2]	Q[3]	Q[4]
8	12	4		

FRONT = 0, REAR = 2, Delete 8

- If we want to delete an element i.e. 8 from the queue, then the value of FRONT will be incremented by 1. Deletions are done from this end of the queue.

REAR = 2, FRONT = 1,

Q[0]	Q[1]	Q[2]	Q[3]	Q[4]
	12	4		

- If we want to delete an element i.e. 12 from the queue, then the value of FRONT will be incremented by 1 i.e. 2.

REAR = 2, FRONT = 2

Q[0]	Q[1]	Q[2]	Q[3]	Q[4]
		4		

- Observe that queue has only one element. When FRONT = REAR condition is true, then the queue Q has only one element. If we want to this element also, then the queue Q becomes empty and set the FRONT and REAR pointers to -1.

REAR = -1, FRONT = -1, Queue is Empty.

Q[0]	Q[1]	Q[2]	Q[3]	Q[4]

➤ **Algorithm for Deletion:**

ALGORITHM: DEQUEUE (QUEUE, REAR, FRONT, ITEM) QUEUE is the array with N elements. FRONT is the pointer that contains the location of the element to be deleted and REAR contains the location of the inserted element. ITEM is the element to be deleted.

Step 1: if FRONT = NULL then [Check Whether Queue is empty]

PRINT "QUEUE is Empty or Underflow"

Exit

[End if]

Step 2: ITEM = QUEUE[FRONT]

Step 3: if FRONT = REAR then [if QUEUE has only one element]

FRONT = NULL

REAR = NULL

else

FRONT = FRONT + 1 [Increment FRONT pointer]

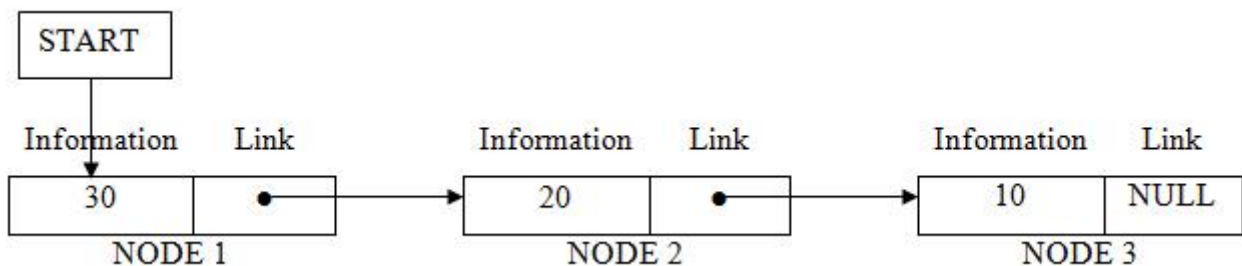
Step 4: Return

➤ Application of Queue:

- Simulation
- Various features of Operating system
- Multi-programming platform systems.
- Different types of scheduling algorithms
- Round robin technique algorithms
- Printer server routines
- Various application software's is also based on queue data structure.

➤ LINKED LIST:

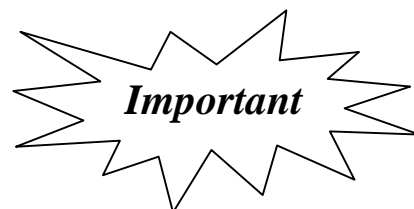
- *A linked list is a linear collection of data elements called nodes.*
- Each nodes is divided into two parts:
 - The first part contains the **information** of the element.
 - The second part contains the memory **address** of the next node in the list. Also called Link part.



- In the above figure shows an representation if Linked List with three nodes, each node contains two parts.
- The left part of the node represents Information part of the node.
- The right part represents the next pointer field. That contains the address of the next node.
- A pointer START or HEAD gives the location of the first node.
- The link field of the last node contains NULL.

➤ Types of Linked List:

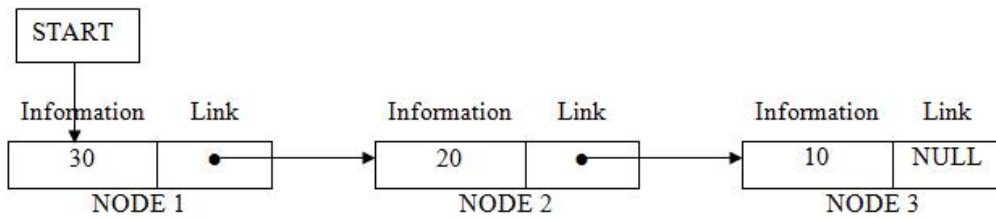
- There are three types of linked lists:
 - Singly Linked List
 - Doubly Linked List
 - Circular Linked List



✓ Singly Linked List:

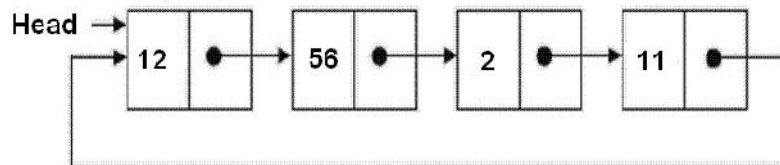
- *A singly linked list contains two fields in each node - an information field and the linked field.*

- The *information* field contains the data of that node.
- The *link* field contains the memory address of the next node.
- There is only one link field in each node, the linked list is called singly linked list.



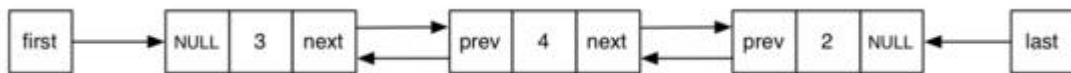
✓ Circular Linked List

- *The link field of the last node contains the memory address of the first node, such a linked list is called circular linked list.*
- In a circular linked list every node is accessible from a given node.



✓ Doubly Linked List

- *It is a linked list in which each node is points both to the next node and also to the previous node.*



- In doubly linked list each node contains three parts:
 - FORW : It is a pointer field that contains the address of the next node
 - BACK: It is a pointer field that contains the address of the previous node.
 - INFO: It contains the actual data.
- In the first node, if BACK contains NULL, it indicated that it is the first node in the list.
- The node in which FORW contains, NULL indicates that the node is the last node.

➤ Operation on Linked List:

- The operation that are performed on linked lists are:
 - Creating a linked list
 - Traversing a linked list
 - Inserting an item into a linked list.
 - Deleting an item from the linked list.
 - Searching an item in the linked list
 - Merging two or more linked lists.

✓ Creating a linked list:

- The nodes of a linked list can be created by the following structure declaration.

```
struct Node
{
    int info;
    struct Node *link;
}*node1, node2;
```

- Here info is the information field and link is the link field.
- The link field contains a pointer variable that refers the same node structure. Such a reference is called as *Self addressing pointer*.

✓ Operator new and delete:

- Operators new allocate memory space.
- Operators new [] allocates memory space for array.
- Operators delete deallocate memory space.
- Operators delete [] deallocate memory space for array.

✓ Traversing a linked list:

- Traversing is the process of accessing each node of the linked list exactly once to perform some operation.

ALGORITHM: TRAVERS (START, P) START contains the address of the first node. Another pointer p is temporarily used to visit all the nodes from the beginning to the end of the linked list.

Step 1:	P = START	
Step 2:	while P != NULL	
Step 3:	PROCESS data (P)	[Fetch the data]
Step 4:	P = link(P)	[Advance P to next node]
Step 5:	End of while	
Step 6:	Return	

✓ AVAIL List

- The operating system of the computer maintains a special list called AVAIL list that contains only the unused deleted nodes.
- Whenever node is deleted from the linked list, its memory is added to the AVAIL list.
- AVAIL list is also called as the free-storage list or free-pool.

✓ **Inserting a node into the linked list:**

- Inserting a node at the beginning of the linked list
- Inserting a node at the given position.
- Inserting a node at the end of the linked list.

• **Inserting a node at the beginning of the linked list**

1. Create a node.
2. Fill data into the data field of the new node.
3. Mark its pointer field as NULL
4. Attach this newly created node to START
5. Make the new node as the STARTing node.

ALGORITHM: INS_BEG (START, P) START contains the address of the first node.

Step 1: P \leftarrow new Node;
 Step 2: data(P) \leftarrow num;
 Step 3: link(P) \leftarrow START
 Step 4: START \leftarrow P
 Step 5: Return

✓ **Garbage Collection:**

- The operating system of the computer periodically collects all the deleted space into the free-storage list. This technique of collecting deleted space into free-storage list is called as *garbage collection*.

✓ **Deleting an item from the linked list:**

- Deletion of the first node
- Deletion of the last node
- Deletion of the node at the give position

• **Deletion of the last node**

ALGORITHM: DEL_END (P1, P2, START) This used two pointers P1 and P2. Pointer P2 is used to traverse the linked list and pointer P1 keeps the location of the previous node of P2.

Step 1: START
 Step 2: P2 \leftarrow START;
 Step 3: while (link(P2) \neq NULL)
 P1 \leftarrow P2

P2 ← link(P2)

While end

Step 4: PRINT data(p2)

Step 5: link(P1) ← NULL

Free(P2)

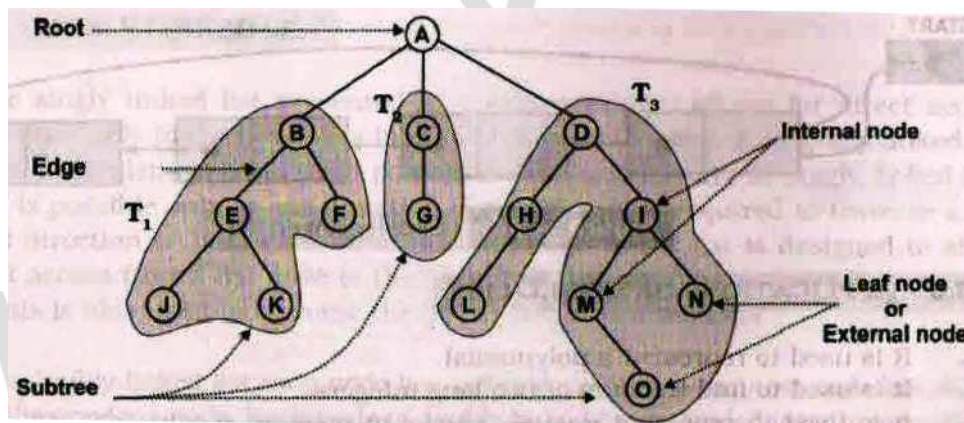
Step 6: STOP

➤ Non-Linear Data structures:

- A Non-Linear Data structures is a data structure in which data item is connected to several other data items.
- The data items in non-linear data structure represent hierarchical relationship.
- Each data item is called node.
- The different non-linear data structures are trees and graphs.

➤ Trees:

- A tree is a data structure consisting of nodes organized as a hierarchy.



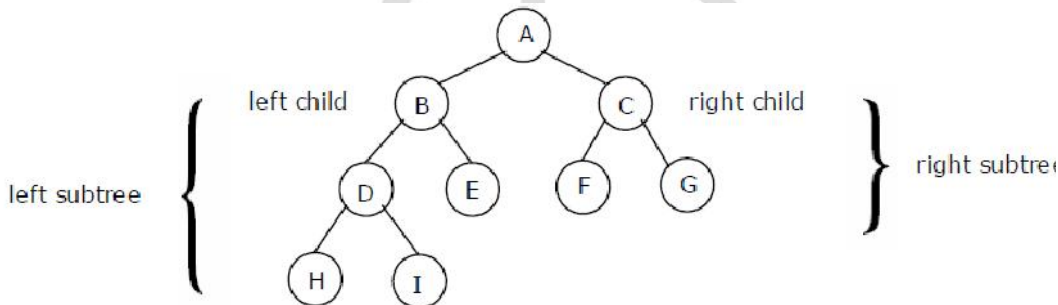
➤ Terminology of a tree:

- **Node:** A node is a fundamental part of tree. Each element of a tree is called a node of the tree.
- **Root:** The topmost node of the tree.
- **Edge:** It connects two nodes to show that tree is a relationship between them.
- **Parent:** It is an immediate predecessor a node. (A is parent of B, C, D)
- **Child:** A successor of parent node is called the child node. (E, F is a child of B)
- **Siblings:** Nodes that are children of the same parent are called siblings. (B,C,D) (M,N)
- **Leaf or terminal node:** Nodes that do not have any children are called leaf node. (J, K, L)
- **Internal Node:** A node has one or more children is called an internal node. (B, C, D, H)

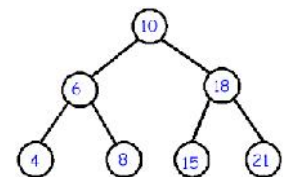
- **Path:** A path is an ordered list of nodes that are connected by edges. In figure path A to O is A, D, I, M, O.
- **Height:** the height or depth of a tree is defined to be the maximum number of nodes in a branch of tree. (The height of tree is 5)
- **Ancestor:** A node reachable by repeated proceeding from child to parent. (Ancestor of M is I, D, and A)
- **Descendant:** A node reachable by repeated proceedings from parent to child. (Descendent of I are (M,O) and N)
- **Subtree:** A Subtree is a set of nodes and edges comprised of parent and all the descendants of that parent. In figure T1, T2 and T3 are subtrees.

➤ **Binary tree:**

- *A binary tree is an ordered tree in which each internal node can have maximum of two child nodes connected to it.*
- A binary tree consists of:
 - A node (called the root node)
 - Left and right sub trees.

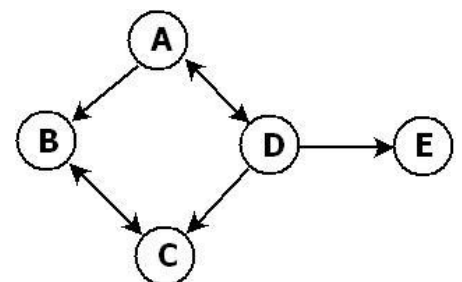


- **A Complete binary tree is a binary tree in which each leaf is at the same distance from the root i.e. all the nodes have maximum two subtrees.**



➤ **GRAPH**

- *A graph is a set of vertices and edges which connect them.*
- A graph is a collection of nodes called vertices and the connection between them called edges.
- **Directed graph:** When the edges in a graph have a direction, the graph is called directed graph or digraph, and the edges are called directed edges or arcs.



CHAPTER – DATA STRUCTURES BLUE PRINT			
VSA (1 Marks)	LA (3 Marks)	Essay (5 Marks)	Total
01 Question	01 Question	02 Question	04 Questions
Question No 3	Question No 23	Question No 28 & 29	14 Marks

Important Questions

1 Mark Questions:

1. **Definition of Data Structure, Linear Data Structures, Array, Stack, Queue, Linked List, Non Linear Data Structures.**
2. Definition on Tree, Tree Terminologies, Binary Tree, Complete Binary Tree, Graph.

3 Marks Questions:

1. Types of Data Structure, Array, Queue, Linked list.
2. Memory Representation of Array, Stack.
3. **Algorithms – Traverse, Insertion, Deletion, Linear Search, PUSH, POP**
4. Conversions: Infix to Postfix & Prefix.
5. Advantages/Applications of arrays, stacks, queues

5 Marks Questions:

- **Operations on Non-Primitive Data Structure, Array, Stack, Queue, Linked list.**
- **Algorithms: Binary Search, Insertion Sort, Enqueue, Dequeue.**
- Tracing Algorithm: Binary Search, Insertion Sort.

Chapter 5

Review of C++

Introduction:

OOP characteristics:

- Abstraction
- Data encapsulation
- modularity
- Inheritance
- Polymorphism
- Dynamic binding
- Message passing

Chapter 5

Review of C++

OOP characteristics:

- **Abstraction:** abstraction represents the essential features of an entity without including background details about it.
- **Data encapsulation:** the wrapping up of data and functions into a single unit called as the class is known as data encapsulation.

the process of insulating the data from direct access by the program is called data hiding.

Chapter 5

Review of C++

OOP characteristics:

- **Modularity:** the lengthy program is divided into small logical components, each component performs a specific task. Each component is called module.
- **Inheritance:** the process in which object of one class acquires the properties of objects of another class is called inheritance.

Chapter 5

Review of C++

OOP characteristics:

- **Polymorphism:** it is an ability of the message to process in more than one form. Poly means many morph means forms. Example, function overloading and operator overloading.
- **Dynamic binding:** binding means linking of a function call to the code to be executed when the function is called.
dynamic binding means the code associated with a function call is not known until the time of the call at run-time.

Chapter 5

Review of C++

OOP characteristics:

- **Message passing:** the objects of a class communicate with each other. For example, `obj.getdata();`

Chapter 5

Review of C++

Fundamentals of C++

C++ was developed by **Bjarne Stroustrup** developed at AT&T bell laboratories new jersey.

C++ character set:

Alphabets: a to z or A to Z

Digits: 0 to 9

Special characters: (,),*,&,!,?,#,@ etc.

Chapter 5

Review of C++

Tokens: a token is a smallest individual unit in a program or lexical unit.

C++ has following tokens:

Identifiers

Keywords

Variables

Constants

Punctuators

Operators

Chapter 5

Review of C++

Identifiers: an identifier is a name given to the programming elements such as variables, functions, arrays etc.

Example: `_abc`, `my_name`, `employee123`, etc.

Keywords: keywords are predefined words that has fixed meaning to the compiler that cannot be changed by the programmer.

Example: `break`, `if`, `else`, `continue`, `case`, `for`, `while` etc.

Chapter 5

Review of C++

Constants or literals: the quantity or value that does not change during execution of the program is called constant.

Integer constant: numbers that doesn't have fractional part is called integer constants.

Integer constants may be decimal, octal, hexadecimal and unsigned form.

Chapter 5

Review of C++

Decimal constants: numbers ranges from 0 to 9 are called decimal constants. Example, +125, -450, 0 etc.

Octal constants: numbers ranges from 0 to 7 are called octal constants. Example, +0725, -0235, 0412 etc.

Hexadecimal constants: numbers ranges from 0 to 9 and A through F. are called hexadecimal constants. Example, 0xfffab, -0X456, etc.

Chapter 5

Review of C++

Unsigned constants: the numbers that are positive are called unsigned constants. These numbers always suffix with u or U. to specify the long type, the numbers suffix with ul or UL.

Example: 132u, 0x452785ul, 0472UL etc.

Chapter 5

Review of C++

Floating point constants: these are also called as real numbers. The numbers that has decimal point are called floating-point numbers.

- Fractional form
- Mantissa exponent notation

Example: 25.25, 78.10, -45.23 etc.

1.58e10, -2.3E-10 etc.

Chapter 5

Review of C++

Character constants: a single character enclosed within a single pair of quotation marks is called character constant. Each character represent ASCII code.

Example: 'H', '6', '?', etc.

Escape sequences: these are another set of character constants. These are non-printable characters commonly used with output statements to organize the output in a proper order.

Example: \', \0, \", \a, \b, \t, \f etc.

Chapter 5

Review of C++

String constants: a string is a sequence of characters enclosed within a pair of double quotation marks. Strings are always terminated by the special character called NULL character ('\0').

Example: "Programming"

Punctuators: these are the symbols that have syntactic and semantic meaning to the compiler. But do not by themselves specify an operation that yields a value.

Example: !, %, ^, &, *, (,), ~

Chapter 5

Review of C++

Operators: an operator is a symbol that tells the compiler to perform some specific mathematical or logical manipulation.

Operators are classified as,

- **Unary operator**
- **Binary operator**
- **Ternary operator**

Chapter 5

Review of C++

- **Unary operator:** operators operates only on one operand is called unary operator.

Example, !, &, ~, *, ++, --, +, -

Binary operator: operators operates on two operands is called binary operator.

Binary operators are classified as,

Arithmetic operators

Relational operators

Logical operators

Bitwise operators

Shorthand operators

Assignment operators

Special operators

Chapter 5

Review of C++

- **Arithmetic operators:** +, -, *, /, %
- **Relational operators:** ==, !=, <, <=, >, >=
- **Logical operators:** &&, ||, !
- **Bitwise operators:** &, |, ^, ~, <<, >>
- **Shorthand operators:** +=, -=, /=, %=, &=, |=, ^=
- **Assignment operator:**

Syntax:

variable=value or expression;

Example, n=5;

- **Special operators:** sizeof(), comma(,), dot(.), pointer(*)

Chapter 5

Review of C++

- **Ternary operators:** operators operate on three or more operands. It is also called as conditional operator.

? :

Precedence of operators or hierarchy of operators

A=2+3*4

BODMAS rule

Type conversion: the process of converting one type of data into another is called type conversion or type casting.

There are two types,

- Implicit type casting
- Explicit type casting

Chapter 5

Review of C++

- **Structure of C++ program**

Include files

class declarations

Member function declaration

main() function

Library function: these are built-in functions present in specific header file

Chapter 5

Review of C++

- **Character functions:** ctype.h

isalpha(), isdigit(), isalnum(), isupper(), isspace(), islower(), tolower(), toupper(), toascii(), ispunct().

String functions: string.h

strlen(), strcat(), strcmp(), strcmpi(), strcpy(), strrev(), strlwr(),strupr()

Console I/O functions: getch(), putchar(), gets(), puts().

Chapter 5

Review of C++

- **Variables:** the quantity that changes during the execution of a program is called variable.

- **Declaration: syntax:** data_type variable_name;

Example, int n;

- **Initializing a variable: syntax:** data_type variable_name=value;

Example, int n=10;

Chapter 5

Review of C++

Data types: it is a type of data that the variable can hold.

There are three categories of data types,

- **Fundamental data types**

int, float, char, double, void

- **Derived data types**

Arrays, functions, pointer and references

- **User defined data types**

enumerated, structure, union and class

Chapter 5

Review of C++

Input and output operators:

Input operator “>>”

Example,

```
int age;
```

```
cin>>age;
```

Output operator “<<”

Example,

```
cout<<“Let us learn C++”;
```

Chapter 5

Review of C++

Cascading of I/O operators:

C++ allows multiple input and output operators in the same instruction is called input output cascading.

Example,

```
cout<<"Enter the value of a";
```

```
cin>>a;
```

```
cout<<"enter the value of b";
```

```
cin>>b;
```

Instead of writing multiple lines of cin and cout statements, we can write it in a single line of instruction as,

```
cout<<"Enter the value of a and b";
```

```
cin>>a>>b;
```


Chapter 5

Review of C++

Control statements: the statements that alter the flow of a sequence of instructions.

Two types of control statements are,

- Selection statements
 - simple if statement
 - if-else statement
 - if-else-if ladder statement
 - nested if statement
- Iterative statements
 - while loop
 - do-while loop
 - for loop
- Jump statements
 - break, exit(), continue and goto.

Chapter 5

Review of C++

Arrays: array is a collection of homogeneous type of data under the same name. each element can be accessed with the help of its index number assigned as 0, 1, 2, ..., n-1.

Types of arrays

- one-dimensional array
- Two-dimensional array
- Multi-dimensional array

Chapter 5

Review of C++

One-dimensional array: elements of the array can be accessed with only one subscript is called one-dimensional array.

Declaration:

Syntax:

```
data_type array_name[size];
```

Example, `int a[10];`

Initialization:

Example, `int a[5]={10,20,30,40,50};`

Chapter 5

Review of C++

Two-dimensional array: elements of the array can be accessed with two subscripts is called two-dimensional array.

Declaration:

Syntax:

```
data_type array_name[row][column];
```

Example, `int a[10][10];`

Initialization:

Example, `int a[2][2]={10,20,30,40};`

Multi-dimensional array: these are array of n-dimensions. It is also called as array of arrays.

Chapter 5

Review of C++

Functions: the complex programs are divided in to smaller logical units, each unit consists of certain set of instructions that performs a specific task called function. It is also called routine or sub-program.

Types of functions:

- Built-in functions
 - `stdio.h`, `string.h`, `iomanip.h`, `iostream.h`, `math.h`, `conio.h` etc.
- User-defined functions

Chapter 5

Review of C++

Inputting single character: get() function is used to input single character.

Example,

```
char ch;          OR          char ch;  
cin=get(ch);     ch=cin.get();
```

Outputting single character: put() function is used to output single character.

Example, cout.put(ch);

Chapter 5

Review of C++

String functions: a string is a sequence of characters enclosed within a pair of double quotation marks. To use string functions we have to include string.h header file.

Syntax:

```
char string_name[size];
```

Example,

```
char str[20];
```

Initializing a string:

```
char str[20]="India";
```

Chapter 5

Review of C++

Inputting a string:

Syntax: `cin.getline(string,size);`

Example, `cin.getline(str,20);`

Outputting a string:

Syntax: `cin.write(string.size);`

Example, `cin.write(str,20);`

Chapter 5

Review of C++

strlen() function: this function is used to return length of the string and terminated by a null ('\0') character.

Syntax: variable=strlen(string);

Example, int l=strlen("Program"); returns 7

strcat() function: this function is used to combine two strings together is called string concatenation.

strcpy() function: a string cannot be copied to another string by using assignment operator. The function strcpy() function is used to copy from one string to another.

Chapter 5

Review of C++

strcmp() function: this function is used to compare character by character. This function is case sensitive. i.e., both uppercase and lowercase letters are distinct.

strcmpi() function: this function is also used to compare character by character. But this function is not case sensitive. i.e. both uppercase and lowercase letters are same.

strrev() function: this function is used to reverse the string.

Chapter 5

Review of C++

User-defined functions: the functions defined by the user to solve his/her problems.

Function definition or structure of user-defined function:

```
return_type function_name(argument_list_with_declaration)
{
Local variable declarations;
Executable statement-1;
Executable statement-2;
...
Executable statement-n;
return(expression);
```

Chapter 5

Review of C++

Calling a function: variable=function_name(argument_list);

OR

Variable=function_name();

Example, big=biggest(a,b,c);

main() function:

```
int main()
```

```
{
```

```
Executable_statements;
```

```
return 0;
```

```
}
```

```
void main()
```

```
{
```

```
OR executable_statements;
```

```
}
```

Returning a value: return(expression) or return 0;

Chapter 5

Review of C++

Function prototypes:

Return_type function_name(type1 arg1, type2 arg2, ...);

OR

Return_type function_name(type1, type2, ...);

Example:

float cube(int a);

OR

float cube(int);

Chapter 5

Review of C++

Types of arguments:

- **Actual arguments**

In the function call `big=biggest(a,b,c);`

- **Formal arguments**

In the function header `int biggest(int a, int b, int c);`

Local variables:

Global variables:

Chapter 5

Review of C++

Type of functions or categories of functions:

- Function with no arguments and no return values.
- Function with arguments and no return values.
- Function with no arguments and with return values.
- Function with arguments and with return values.
- Recursive functions.

Chapter 5

Review of C++

Type of functions or categories of functions:

- Function with no arguments and no return values.

Example: void natural()

```
{  
  for(int i=0;i<n;i++)  
    cout<<i;  
}
```


Chapter 5

Review of C++

Type of functions or categories of functions:

- Function with arguments and no return values.

Example: void average(int x, int y, int z)

```
{  
float sum,average;  
sum=a+b+c;  
average=sum/3.0;  
cout<<"Average="<<average;  
}
```

Chapter 5

Review of C++

Type of functions or categories of functions:

- Function with no arguments and with return values.

Example: int greatest()

```
{  
    if(a>b)  
        return(a);  
    else  
        return(b);  
}
```

Chapter 5

Review of C++

Type of functions or categories of functions:

- Function with arguments and with return values.

Example: float interest(float p, float t, float r)

```
{  
    si=p*t*r/100.0;  
    return(si);  
}
```

- **Recursive functions:** The function that calls by itself is called recursive function.

Recursion function must have one or more terminating conditions to terminate recursion. Otherwise, recursion will become infinite.

Chapter 5

Review of C++

Passing default arguments to functions:

Example: consider the prototype,

```
float interest(float p, int t, float r=0.10);
```

0.10 is a default value assigned to the argument rate

The function call statement must be,

```
si=interest(1000.0,2);
```

Passing constant arguments:

Example: `int strlen(const char *p);`

```
int total(const int x, const int y);
```

Chapter 5

Review of C++

Pass by value or call by value:

Pass by reference or call by reference:

Passing arrays to functions:

Passing structures to functions:

Chapter 5

Review of C++

Structures:

Defining a structure:

```
struct structure_name
{
Data_type1 member_name1;
Data_type2 member_name2;
...
Data_type-n member_name-n;
};
```

Chapter 5

Review of C++

Example:

```
struct employee
{
int empid;
char name[10];
char designation[10];
float salary;
};
```

Chapter-6

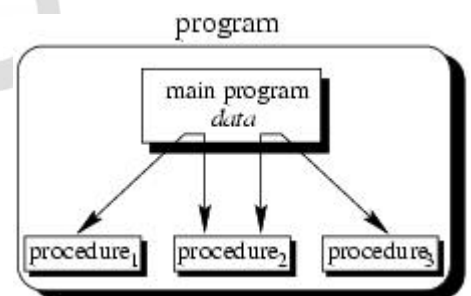
BASIC CONCEPT OF OOP

➤ Introduction:

- Object oriented programming is the principle of design and development of programs using modular approach.
- Object oriented programming approach provides advantages in creation and development of software for real life application.
- The basic element of object oriented programming is the data.
- The programs are built by combining data and functions that operate on the data.
- Some of the OOP's languages are C++, Java, C #, Smalltalk, Perl, and Python.

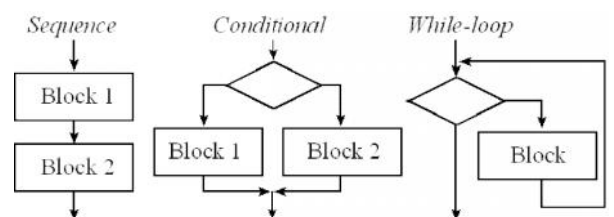
➤ Procedural programming:

- The procedural programming focuses on **processing** of instructions in order to perform a desired computation. Therefore it emphasizes more on doing things like algorithms.
- This programming is lengthy, increases the complexity of program, difficult to understand and modify the program.
- This technique is used in a conventional programming language such as C and Pascal.



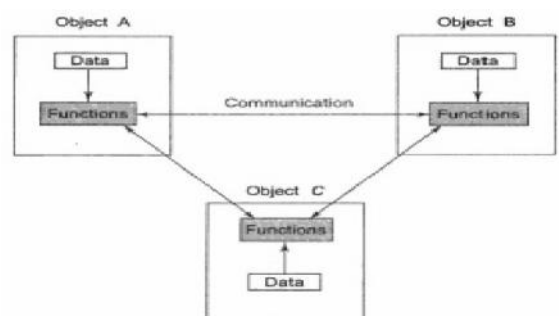
➤ Structured programming:

- An organized approach to programming involving the use of three basic control structures – Sequence, Conditional and loop.
- The top-down concepts to decompose main functions into lower level components for modular coding purpose.
- The major drawback is that it is very difficult to model the real world scenario using this model.



➤ Object oriented programming:

- Object oriented programming (OOP) is a concept that combines both the data and the functions that operate on that data into a single unit called the object.
- An object is a collection of set of data known as



member data and the functions that operate on these data known as member function.

- OOP follows bottom-up design technique.
- Class is the major concept that plays important role in this approach. Class is a template that represents a group of objects which share common properties and relationships.

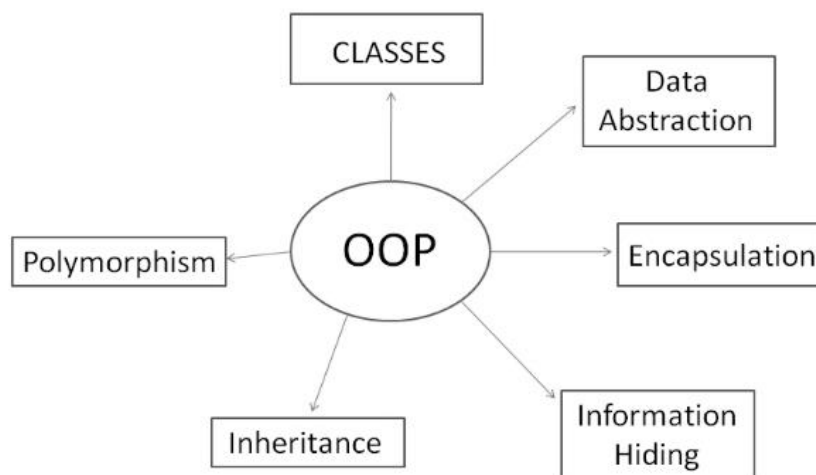
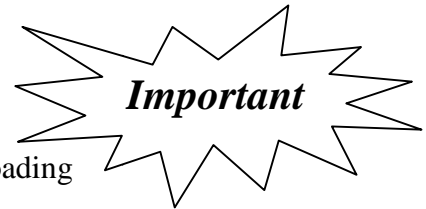
➤ **Difference between Procedural Programming & Object Oriented programming:**

Procedural Programming	Object Oriented Programming
Large programs are divided into smaller programs known as functions	Programs are divided into objects
Data is not hidden and can be accessed by external functions	Data is hidden and cannot be accessed by external functions
Follow top down approach in the program design	Follows bottom-up approach in the program design
Data may communicate with each other through functions	Objects may communicate with each other through functions.
Emphasize is on procedure rather than data	Emphasize is on data rather than procedure

➤ **Basic Concepts of OOP's:**

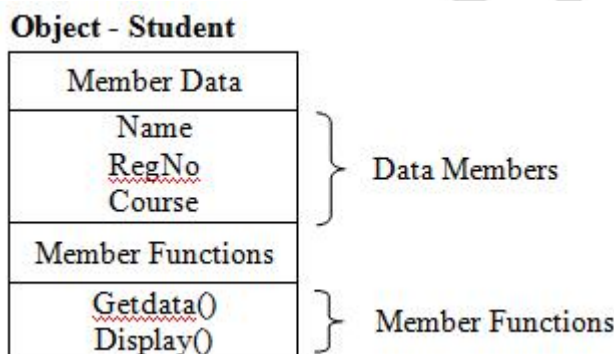
The following are the major characteristics of OOP's:

- Objects
- Class
- Data abstraction
- Data encapsulation
- Inheritance
- Overloading
- Polymorphism
- Dynamic Binding
- Message Passing



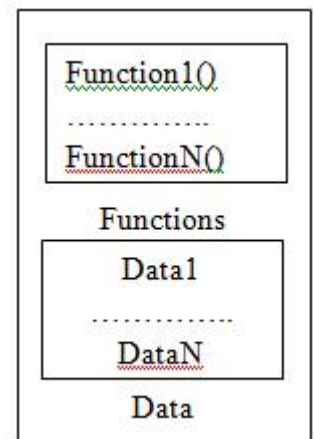
➤ **Objects**

- Objects are basic building blocks for designing programs.
- *An object is a collection of data members and associated member functions.*
- An object may represent a person, place or a table of data.
- Each object is identified by a unique name. Each object must be a member of a particular class.
- Example: Apple, orange, mango are the objects of class fruit.
- Objects take up space in memory and have address associated with them.
- At the time of execution of a program, the objects interact by sending the messages to one another.
- The objects can interact with one another without having to know details of data or functions within an object.



➤ **Classes:**

- The objects can be made user defined data types with the help of a class.
- *A class is a collection of objects that have identical properties, common behavior and shared relationship.*
- Once class is defined, any number of objects of that class is created.
- Classes are user defined data types. A class can hold both data and functions.
- For example: Planets, sun, moon are member of class solar system.

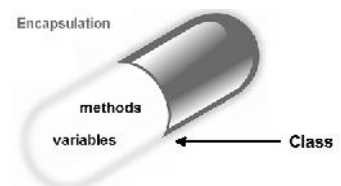


➤ **Data Abstraction:**

- *Data Abstraction refers to the process of representing essential features without including background details or explanations.*

➤ **Data Encapsulation:**

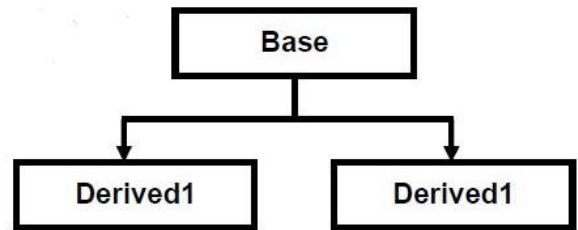
- *The wrapping of data and functions into a single unit (class) is called data encapsulation.*
- Data encapsulation enables data hiding and information hiding.



- **Data hiding** is a method used in object oriented programming to hide information within computer code.

➤ Inheritance:

- **Inheritance is the process by which one object can acquire and use the properties of another object.**
- The existing class is known as **base class** or super class.
- The new class is known as **derived class** or sub class.
- The derived class shares some of the properties of the base class. Therefore a code from a base class can be reused by a derived class.

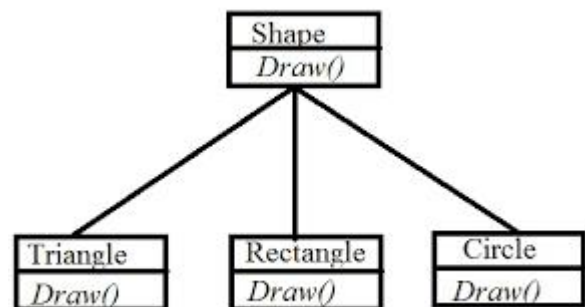


➤ Overloading:

- **Overloading allows objects to have different meaning depending upon context.**
- There are two types of overloading viz.
 - Operator Overloading
 - Function Overloading
- When an existing operator operates on new data type is called **operator overloading**.
- **Function overloading** means two or more function have same, but differ in the number of arguments or data type of arguments.

➤ Polymorphism:

- **The ability of an operator and function to take multiple forms is known as Polymorphism.**
- The different types of polymorphism are operator overloading and function overloading.



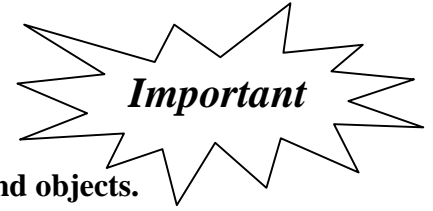
➤ Dynamic binding:

- Binding is the process of connecting one program to another.
- Dynamic binding is the process of linking the procedure call to a specific sequence of code or function at run time or during the execution of the program.

➤ Message Passing:

- In OOP's, processing is done by sending message to objects.
- A message for an object is request for execution of procedure.

- *Message passing involves specifying the name of the object, the name of the function (message) and the information to be sent.*



➤ Advantage of OOP's

- The programs are modularized based on the **principles of classes and objects**.
- Linking code & object allows related objects to share common code. This reduces **code duplication and code reusability**.
- Creation and implementation of OOP code is **easy and reduces software development time**.
- The concept of data abstraction separates **object specification and object implementation**.
- Data encapsulated along with functions. Therefore external non-member function cannot access or modify data, thus **proving data security**.
- Easier to develop complex software, because **complexity can be minimized through inheritance**.
- **OOP can communicate through message passing** which makes interface description with outside system very simple.

➤ Disadvantage of OOP's

- **Larger program size:** OOP's typically involves more lines of code than procedural programs.
- **Slower Programs:** OOP's typically slower than procedure based programs, as they typically require more instructions to be executed.
- Not suitable for all types of programs.
- To convert a real world problem into an object oriented model is difficult.
- OOP's software development, debugging and testing tools are not standardized.
- Polymorphism and dynamic binding also requires processing time, due to overload of function calls during run time.

➤ Application of OOP's

- Computer graphics applications.
- .CAD/CAM software
- Object-oriented database.
- User-Interface design such as windows
- Real-time systems.
- Simulation and Modeling
- Artificial intelligence and expert systems.
- Client-Server Systems.

CHAPTER 6 – Basic Concept of OOP's BLUE PRINT				
VSA (1 marks)	SA (2 marks)	LA (3 Marks)	Essay (5 Marks)	Total
-	01 Question	-	01 Question	02 Question
-	Question No 13		Question No 30	07 Marks

Important Questions

2 Marks Question:

1. Explain: Classes, Objects, Data Abstraction, Data Encapsulation, Inheritance, and Polymorphism.
2. What is Base class and derived class?

5 Marks Question:

1. Distinguish between procedural and object oriented programming.
2. Explain the characteristics of OOP's.
3. Briefly explain the basic concepts of OOP's.
4. Explain the advantages of OOP's.
5. Mention disadvantages of OOP's.
6. Write the applications of OOP's.

Chapter-7

CLASSES AND OBJECTS

➤ Classes:

- *A class is a collection of objects that have identical properties, common behavior and shared relationship.*
- A class binds the data and its related functions together.

➤ Definition and Declaration of Classes:

- A class definition is a process of naming a class and data variables, and interface operation of the class.
- The variables declared inside a class are known as *data members*.
- The functions declared inside a class are known as *member functions*.
- A class declaration specifies the representation of objects of the class and set of operations that can be applied to such objects.
- The general syntax of the class declaration is:

```
class User_Defined_Name
{
    private :
        Data Member;
        Member functions;
    public :
        Data Member;
        Member functions;
    protected :
        Data Member ;
        Member functions;
};
```



- Key word **class** is used to declare a class. User_Defined_Name is the name of the class.
- Class body is enclosed in a pair of flower brackets. Class body contains the declaration of its members (data and functions).
- There are generally three types of members namely private, public and protected.
- Example: Let us declare a class for representation of bank account.

```
class account
{
    private:
```

```

        int accno;
        char name[20];
        char acctype[4];
        int bal_amt;
    public:
        void get_data( );
        void display_data( );
};

```

➤ Access Specifiers:

- Every data member of a class is specified by three levels of access protection for hiding data and function members internal to the class.
- They help in controlling the access of the data members.
- Different access specifiers such as private, public, and protected.

✓ **private:**

- *private* access means a member data can only be accessed by the class member function or friend function.
- The data members or member functions declared *private* cannot be accessed from outside the class.
- The objects of the class can access the private members only through the public member functions of the class. This property is also called *information hiding*.
- By default data members in a class are private.
- Example:

```

private:
    int x;
    float y;

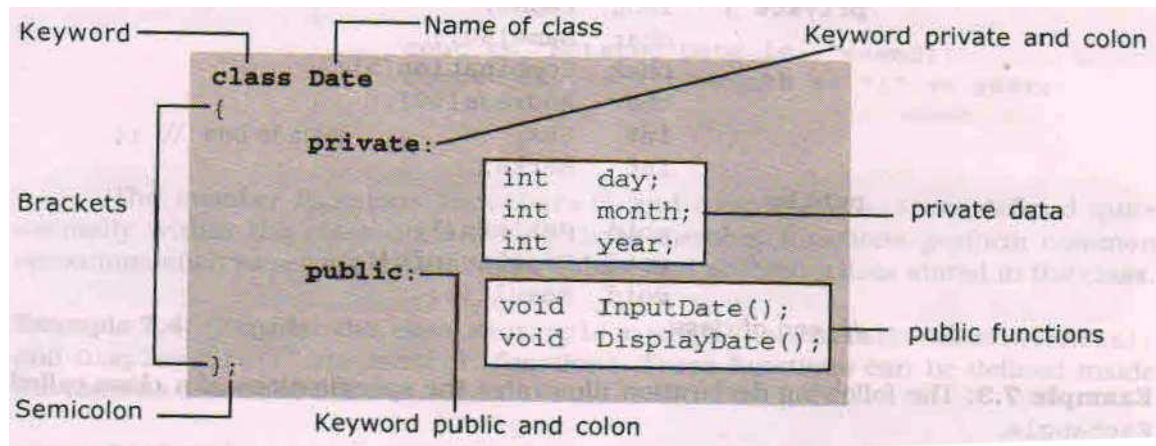
```

✓ **protected:**

- The members which are declared using *protected* can be accessed only by the member functions, friend of the class and also the member functions derived from this class.
- The members cannot be accessed from outside the class.
- The *protected* access specifier is similar to private access specifiers.

✓ **public:**

- *public* access means that member can be accessed any function inside or outside the class.
- Some of the *public* functions of a class provide interface for accessing the private and protected members of the class.



➤ Member Function:

- Member functions are functions that are included within a class (Member functions are also called Methods).
- Member functions can be defined in two places.
 - Inside class definition
 - Outside class definition

**Important
5 Marks**

✓ Inside class definition:

- To define member function inside a class the function declaration within the class is replaced by actual function definition inside the class.
- A function defined in a class is treated as inline function.
- Only small functions are defined inside class definition.
- Example:

```

class rectangle
{
    int length, breadth, area;
public:
    void get_data( )
    {
        cout<< " Enter the values for Length and Breadth";
        cin>>length>>breadth;
    }
    void compute( )
    {
        area = length * breadth;
    }
    void display( )
    {
        cout<<" The area of rectangle is"<<area;
    }
};

```


✓ **Outside class definition:**

- To define member function outside the class declaration, you must link the class name of the class with the name of member function.
- We can do this by preceding the function name with the class name followed by two colons (::).
- The two colons (::) are called *scope resolution operator*.
- Scope resolution operator (::) is used to define the member function outside the class.
- The general form of a member function defined outside the class is:

```

return_type class_name :: member_function_name( arg1, arg2, ....argN)
{
    function body;
}

```

- Example:

```

class operation
{
    private:
        int a, b;
    public:
        int sum( );
        int product( );
};
int operation :: sum( )
{
    return (a+b);
}
int operation :: product( )
{
    return (a * b);
}

```

- ❖ **Program: To use classes using member functions inside and outside class definition.**

```

#include<iostream.h>
class item
{
    private:
        int numbers;
        float cost;
    public:
        void getdata(int a, float b);
        void putdata( )
        {
            cout<<"Number: "<<number<<endl;
            cout<<"Cost:"<<cost<<endl;
        }
};

```

```

void item : : getdata(int a, float b)
{
    number = a;
    cost = b;
}
int main()
{
    item x;
    x. getdata( 250, 10.5);
    x.putdata( );
    return 0;
}

```

OUTPUT:**Number: 250****Cost: 10.5**➤ **Defining object of a class:**

- An object is a real world element which is identifiable entity with some characteristics (attributes) and behavior (functions).
- An object is an instance of a class. Objects are sometimes called as instance variables.
- An object is normally defined in the main () function.
- The syntax for defining objects of a class as follows:

```

class Class_Name
{
    private :           //Members
    public :           //Members
};
class Class_Name Object_name1, Object_name2,.....;

```

where class keyword is optional.

- **Example 1:** The following program segment shows how to declare and create objects.

```

class Student
{
    private:
        int rollno;
        char name[20];
        char gender;
        int age;
    public:
        void get_data( );
        void display( );
};
Student S1, S2, S3;           //creation of objects

```

- Here, creates object S1, S2, and S3 for the class Student.
- When an object is created space is set aside for it in memory.

- **Example 2:**

```
class num
{
    private :
        int x, y;
    public :
        int sum(int p, int q)
        int diff(int p, int q)
};
void main( )
{
    num s1, s2;
    s1.sum ( 200,300);
    s2.diff (600, 500);
}
```

➤ **Accessing member of the class:**

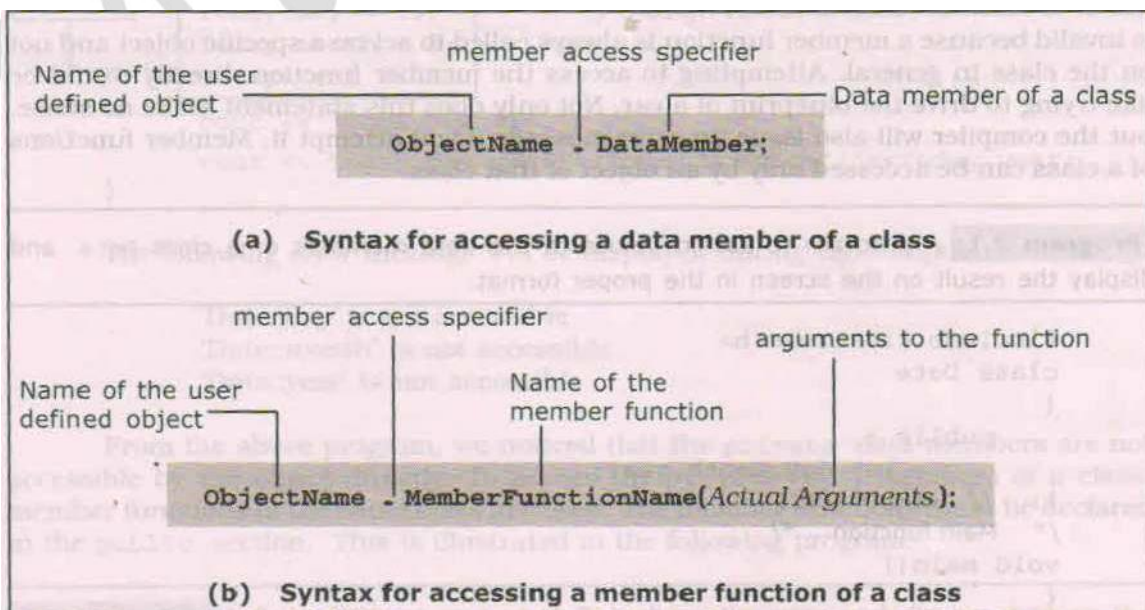
- The member of the class can be data or functions.
- Private and protected members of the class can be accessed only through the member functions of the class.
- No functions outside a class can include statements to access data directly.
- The public data members of objects of a class can be accessed using *direct member access operator* (.).
- The syntax of accessing member (data and functions) of a class is:

a) Syntax for accessing a data member of the class:

Object_Name . data_member;

b) Syntax for accessing a member function of the class:

Object_Name . member_function(arguments)



- **Example:**

```

class rectangle
{
    int length, breadth, area;
public:
    void get_data( )
    {
        cout<<"Enter the length and breadth"<<end;
        cin>>length>>breadth;
    }
    void compute( )
    {
        area = length * breadth;
    }
    void display( )
    {
        cout<<"The area of rectangle is "<<area;
    }
};
void main( )
{
    rectangle r1;
    clrscr( );
    r1.get_data( );
    r1.compute( );
    r1.display( );
    getch( );
}

```

OUTPUT:

```

Enter the length and breadth
30    10
The area of rectangle is 300

```

- **Array as member of classes:**

- Array can be used as a data member of classes.
- An array can be used as private or public member of a class.
- **This is illustrated in the following program.**

```

#include<iostream.h>
#include<conio.h>
class array
{
    private:
        int a[100], m;
    public:
        void setnoofelements( int n);
        {
            m = n;
        }
}

```

```

        void readarray( );
        void displayarray( );
};
void array :: readarray( )
{
    cout<<"Enter "<<m<<"Array elements"<<endl;
    for(int i=0; i<m; i++)
        cin>> a[i];
}
void array :: displayarray( )
{
    cout<<"Array elements are:"<<endl;
    for(int i=0; i<m i++)
        cout<< a[i]<<"\t";
}
void main( )
{
    int n;
    array a;
    clrscr( );
    cout<<"Input number of elements:"<<endl;
    cin>>n;
    a.setnoofelements(n);
    a.readarray( );
    a.dispalyarray( );
    getch( );
}

```

OUTPUT:

Input number of elements: 5

Enter 5 Array elements

10 20 30 40 50

Array elements are:

10 20 30 40 50

➤ Classes, Objects and Memory:

- The class declaration does not allocate memory to the class data member.
- When a object is declared, memory is reserved for only data members and not for member functions.
- **The following program illustrates as follows:**

```

#include<iostream.h>
#include<conio.h>
class student
{
    private:
        long regno;           // 4 bytes of memory
        char name[20];        // 20 bytes of memory
        char comb[4];         // 4 bytes of memory
        int marks;           // 2 bytes of memory
}

```

```

        char address[30];           // 30 bytes of memory
    public:
        void readdata( );
        void display( );
};
void main( )
{
    student s1, s2;
    cout<<"Size of the object s1 is = "<<<sizeof(s1)<<endl;
    cout<<"Size of the object s2 is = "<<<sizeof(s2)<<endl;
    cout<<"Size of the class is = "<<<sizeof(student)<<endl;
}

```

OUTPUT:

Size of the object s1 is = 60

Size of the object s2 is = 60

Size of the class is = 60

➤ **Array of Objects:**

- An array having class type elements is known as array of objects.
- An array of objects is declared after definition and is defined in the same way as any other array.
- Example:

```

class employee
{
    private:
        char name[10];
        int age;
    public:
        void readdata( );
        void displaydata( );
};
employee supervisor[3];
employee sales_executive[5];
employee team_leader[10];

```



**Important
5 Marks**

- In the above example, the array supervisor contains 3 objects namely supervisor[0], supervisor[1], supervisor[2].
- The storage of data items in an object array is:

	Name	Age
supervisor[0]		
supervisor[1]		
supervisor[2]		

- **Program to show the use of array of objects:**

```

#include<iostream.h>
#include<conio.h>
class data
{
    private:

```

```

        int regno, maths, computer;
    public:
        void readdata( );
        void average( );
        void display( );
};
void data :: readdata( )
{
    cout<<"Enter Register No:";
    cin>>regno;
    cout<<"Enter Maths marks:";
    cin>>maths;
    cout<<"Enter Computer marks:";
    cin>>computer;
    dipalay( );
}
void data :: average( );
{
    int avg;
    avg = (maths+computer)/2;
}
void data :: display( )
{
    cout<<"Average = "<<average( )<<endl;
}
void main( )
{
    data stude[3];
    clrscr( );
    for(i=0; i<3; i++)
        stud[i]. readdata( );
    getch( );
}

```

OUTPUT:

```

Enter Register No: 20
Enter Maths marks: 56
Enter Computer marks: 78
Average = 67
Enter Register No: 22
Enter Maths marks: 56
Enter Computer marks: 77
Average = 66
Enter Register No: 10
Enter Maths marks: 44
Enter Computer marks: 89
Average = 66

```

➤ Objects as function arguments:

- A function can receive an object as a function argument.
- This is similar to any other data being sent as function argument.
- An object can be passed to a function in two ways:
 - Copy of entire object is passed to function (Pass by value)
 - Only address of the object is transferred to the function (Pass by reference)
- In **pass by value**, copy of object is passed to the function.
- The function creates its own copy of the object and uses it.
- Therefore changes made to the object inside the function do not affect the original object.

```

#include<iostream.h>
#include<conio.h>
class exam
{

```

```

private:
    float phy, che, mat ;
public:
    void readdata( )
    {
        cout<<"Input Physics, Chemistry, Maths marks : " ;
        cin>>phy>>che>>mat;
    }
    void total(exam PU , exam CT)
    {
        phy = PU.phy + CT.phy;
        che = PU.che + CT.che;
        mat = PU.mat + CT.mat;
    }
    void display( )
    {
        cout<< "Physics : " <<phy<<endl;
        cout<< "Chemistry : " <<che<<endl;
        cout<< "Maths : " <<mat<<endl;
    }
};
void main( );
{
    Exam PUC, CET, Puc_plus_Cet;
    clrscr( );
    cout<<"Enter PUC Marks"<<endl;
    PUC.readdata( );
    cout<<"Enter CET Marks"<<endl;
    CET.readdata( );
    Puc_plus_Cet.total(PUC, CET);
    cout<<"Total marks of PUC and CET is:" <<endl;
    Puc_plus_Cet.display( );
}

```

OUTPUT:

```

Enter PUC Marks
Input Physics, Chemistry, Maths marks :
67    89    80
Enter CET Marks
Input Physics, Chemistry, Maths marks :
60    76    91
Total marks of PUC and CET is:
Physics: 127
Chemistry: 165
Maths: 171

```

- In **pass by reference**, when an address of an object is passed to the function, the function directly works on the original object used in function call.
- This means changes made to the object inside the function will reflect in the original object, because the function is making changes in the original object itself.
- Pass by reference is more efficient, since it requires only passing the address of the object and not the entire object.

➤ Difference between Structure and Classes:

Structure	Classes
A structure is defined with the struct keyword	A class is defined with the class keyword
All the member of a structure are public by default	All the members of a class are private by default

Structure cannot be inherit	Class can be inherit
A structure contains only data member	A class contain both data member and member functions
There is no data hiding features	Classes having data hiding features by using access specifiers(public, private, protected)

CHAPTER 7 – Classes and Objects BLUE PRINT

VSA (1 marks)	SA (2 marks)	LA (3 Marks)	Essay (5 Marks)	Total
01 Question	-	-	01 Question	06 Marks
Question No 04	-	-	Question No 31	-

IMPORTANT QUESTIONS:

1 Mark questions:

1. What is a Class, Objects, Data Member, Member Functions, Scope Resolution Operator, and Array of objects?
2. Mention the access specifiers used with a class?

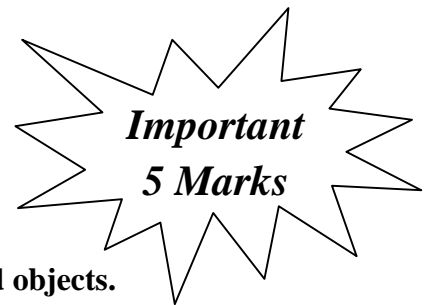
5 Mark questions:

1. Explain class definitions and class declaration with syntax and example.
2. Explain Member function.
 - a. Inside class definition
 - b. Outside class definition
3. Explain the array of objects.

Exercise programs

1. Write a C++ program to find the simple interest using class and objects.

```
#include<iostream.h>
#include<conio.h>
class SI
{
    private:
        float p, t, r, si;
    public:
        void readdata( )
        {
            cout<<"Enter the Principal Amount, Time & Rate"<<endl;
```



```

        cin>>p>>t>>r;
    }
    void compute( )
    {
        si = (p * t * r)/100;
    }
    void display( )
    {
        cout<<"Simple Interest = "<<si;
    }
};
void main( )
{
    SI s;
    clrscr( );
    s.readdata( );
    s.compute( );
    s.display( );
    getch( );
}

```

2. Let product list be a linear array of size N where each element of the array contains following field Itemcode, Price and Quantity. Declare a class Product list with three data members and member functions to perform the following

a. Add values to the product list

b. Printing that total stock values

```

#include<iostream.h>
#include<conio.h>
#include<iomainp.h>
class product
{
    private:
        char itemcode[6];
        float price, quantity;
    public:
        void Addproduct( )
        {
            cout<<"Enter the Item Code"<<endl;
            cin>>itemcode;
            cout<<"Enter the Price"<<endl;
            cin>>price;
            cout<<"Enter the Quantity"<<endl;
            cin>>quantity;
        }
}

```

```

        void display( )
        {
            cout<<itemcode<<"\t"<<price<<"\t"<<quantity<<endl;
        }
};
void main( )
{
    int N=0;
    char ans;
    product list[100];
    clrscr( );
    while(1)
    {
        cout<<"Item Code, Price and Quantity"<<endl;
        List[N].Addproduct( );
        cout<<"Do you want to add next item (Y/N)?"<<endl;
        cin>>ans;
        if(toupper(ans) == 'N')
            break;
        N++;
    }
    cout<<"Item Code \t Price \t Quantity"<<endl;
    for(i=0; i<N; i++)
        List[i].display( );
    getch( );
}

```

3. A class cock has following member hours and minutes. Create member function

- a. To initialize the data members**
- b. Display the time**
- c. To convert hours and minutes to minutes.**

```

#include<iostream.h>
#include<conio.h>
class clock
{
    private:
        int hh, mm;
    public:
        void initialize( int h, int m)
        {
            hh = h;
            mm = m;
        }
        void display( )
        {

```

```

        cout<<"Hours = "<<hh;
        cout<<"Minutes = "<<mm;
    }
    void convert( )
    {
        mm = hh * 60 + mm;
        cout<<"Total Minutes = "<<mm;
    }
};
void main( )
{
    int h, m;
    clock c;
    clrscr( );
    cout<<"Enter the Hour and Minutes"<<endl;
    cin>>h>>m;
    c.intialize( );
    c.display( );
    c.convert( )
    getch( );
}

```

4. Write a C++ program that receives arrival time and departure time and speed of an automobile in kilometers/hours as input to a class. Compute the distance travelled in meters/second and display the result using member functions.

```

#include<iostream.h>
#include<conio.h>
class Distance
{
    private:
        int Ahh, Amm, Dhh, Dmm;
        float speed;
    public:
        void inputtime( )
        {
            cout<<"Enter the Arrival Time:"<<endl;
            cout<<"Enter the Hour and Minutes"<<endl;
            cin>>Ahh>>Amm;
            cout<<"Enter the Departure Time:"<<endl;
            cout<<"Enter the Hour and Minutes"<<endl;
            cin>>Dhh>>Dmm;
            cout<<"Enter the speed in Kmph"<<endl;
            cin>>speed;
        }
        void computedistance( )

```

```
        {
            float dist;
            dist = ( Ahh * 60 + Amm) – (Dhh * 60 + Dmm) ) * speed/60;
            dist = (dist * 1000 / (60 * 60));
            cout<<"Distance Travelled = "<<dist<<"Meter/Second";
        }
};
void main( )
{
    Distance d;
    clrscr( );
    d.inputtime( );
    d.computedistance( );
    getch( );
}
```

Chapter-8

FUNCTION OVERLOADING AND MEMBER FUNCTION

➤ Introduction:

- User defined function is a function defined by the user to solve his/her problem. Such a function can be called from anywhere and any number of times in the program.
- C++ implements polymorphism through function overloading and operator overloading.
- Function overloading allows the user to create new abstract data type.

➤ Function Overloading:

- Function Overloading means two or more functions have same name, but differ in the number of arguments or data types of arguments.
- Function overloading is the process of defining same function name to carry out similar types of activities with various data items.

➤ Definition and Declaration of overloaded functions:

- The main factor in function overloading is a functions argument list.
- If there are two functions having same name and different types of arguments or different number of arguments, then function overloading is invoked automatically by the compiler.
- Function Overloading is also known as *Compile time polymorphism*.
- Example:

```
int sum (int a, int b)
float sum ( float p, float q)
```

- The function sum() that takes two integer arguments is different from the function sum() that takes two float arguments. This is function overloading.
- To overload a function, each overloaded function must be declared and defined separately.
- Example:

```
int product ( int p, int q, int r);
float product ( float x, float y);
int product ( int p, int q, int r)
{
    cout<<"Product = "<<p * q * r << endl;
}
float product ( float x, float y, float z);
{
    cout<< "Product = " << x * y <<endl;
}
```



➤ Calling Overloaded Functions:

- The following program shows how overloaded functions can be called.

Program: To compute area of rectangle, circle, triangle using overloaded functions.

```
#include<iostream.h>
#include<conio.h>
class funoverloaded
{
    public:
        int area ( int l, int b)           // area of rectangle
        {
            return (l * b);
        }
        float area ( float r)             // area of circle
        {
            return (3.14 * r * r);
        }
        float area (float b, float h)
        {
            return (0.5 * b * a);        // area of triangle
        }
};
void main()
{
    funoverloaded f;
    clrscr();
    cout<<"Area of Rectangle:"<<f.area(4,6)<<endl;
    cout<<"Area of Circle:"<<f.area(10)<<endl;
    cout<<"Area of Triangle:"<<f.area(3.0, 7.0)<<endl;
    getch();
}
```

OUTPUT:

```
Area of Rectangle: 24
Area of Circle: 314.2857
Area of Triangle: 10.5
```

➤ Need for function overloading:

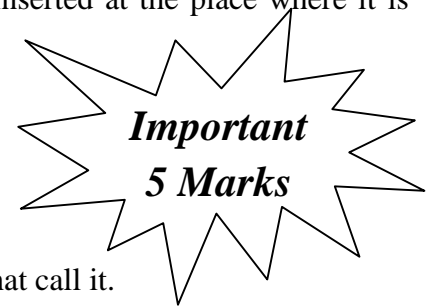
- The advantage of function overloading are:
 - Code is executed faster.
 - It is easier to understand the flow of information and debug.
 - Code Maintenance is easy.
 - Easier interface between programs and real world objects.

➤ Restrictions on Overloaded Functions:

- Each function in a set of overloaded functions must have different argument list.
- If typedef is used for naming functions, then the function is not considered as different type.

➤ **Inline Function:**

- An **inline** function is a special type of function whose body is inserted at the place where it is called, instead of transferring the control to the function.
- The keyword **inline** is used to define inline function.
- Rules:
 - Inline function definition starts with keyword **inline**.
 - The inline function should be defined before all function that call it.
 - The compiler replaces the function call statement with the function code itself (expansion) and then compiles the entire code.



- The general format for the inline function declaration is given below:

```
inline Returntype Fun_Name ( [Argument] )
{
    ..... ;
    return expression ;
}
```

- **Program to find the cube of a number using inline function:**

```
#include<iostream.h>
#include<conio.h>
inline int cube ( int a )
{
    return a * a * a;
}
void main( )
{
    int n ;
    clrscr( );
    cout<<"Enter the input number"<<endl;
    cin>>n;
    cout<<"Cube of " <<n<<" = "<<cube(n);
    getch();
}
```

OUTPUT:

```
Enter the input number
4
Cube of 4 = 64
```

✓ **Advantage of inline function:**

- The size of the object code is considerably reduced.
- The speed of execution of a program increases.
- Very efficient code can be generated.
- There is no burden on the system for function calling.
- It also saves the overhead of return call from a function.
- The readability of the program increases.

✓ **Disadvantage of inline function:**

- May increase the size of the executable file
- More memory is needed.
- If used in header file, it will make your header file size large and may also make it unreadable.

✓ **Note: The inline function may not work some times for one of the following reasons:**

- The inline function definition is too long or too complicated.
- The inline function is recursive.
- The inline function has looping constructs.
- The inline function has a switch or goto.

➤ **Friend Function:**

- A friend function is a non-member function of a class has the access permission to the private member of the class.
- The friend function is declared within a class with the prefix friend.
- But it should be defined outside the class like a normal function without the prefix friend.
- The general format for the friend function is given below:

```
class class_name
{
    public:
        friend return_type function_name ( [arguments] );
}
```



✓ **The friend functions have the following properties:**

- Friend function is not a member function of the class, has full access permission to private and protected members of the class.
- It can be declared either in public or private part of a class.
- A friend function cannot be called using the object of that class. It can be invoked like any normal function.
- The function is declared with keyword friend. But while defining friend function it does not use either keyword friend or :: operator.
- They are normal external functions that are given special access privileges.
- It cannot access the data member variables directly and has to use an object name.membername.
- Use of friend function is rare, since it violates the rule of encapsulation and data hiding.

- ✓ **Program to check a number is even or odd using a friend function.**

```
#include<iostream.h>
#include<conio.h>
class number
{
    private:
        int a;
    public:
        void readdata( )
        {
            cout<<"Enter the Number"<<endl;
            cin>>a;
        }
        friend int even(number);
};
int even(number n)
{
    if(n.a % 2 == 0)           //friend function can access the private data a of the object n
        return 1;
    else
        return 0;
}
void main( )
{
    number num1;
    clrscr( );
    num1.readadadata( );
    if( even(num1) )           //friend function call
        cout<<"Number is Even";
    else
        cout<<'Number is Odd';
    getch();
}
```

OUTPUT:

```
Enter the Number
10
Number is Even
Enter the Number
11
Number is Odd
```

CHAPTER 8 – Function Overloading and Member Function BLUE PRINT				
VSA (1 marks)	SA (2 marks)	LA (3 Marks)	Essay (5 Marks)	Total
-	-	-	01 Question	01 Question
-	-		Question No 32	05 Marks

Important Questions

5 Marks Question:

1. What is function overloading? Explain the need for function overloading.
2. Discuss overloaded functions with syntax and example.
3. What is inline function? Write a simple program for it.
4. Mention the advantage and disadvantage of inline function.
5. Explain friend function and their characteristics.
6. Program to check whether a number is prime or not using inline function:

```
#include<iostream.h>
#include<conio.h>
inline int prime ( int n )
{
    for(int i=2; i<n/2; i++)
        if ( n % i == 0)
            return 0;
    return 1;
}
void main()
{
    int num ;
    clrscr();
    cout<<"Enter the input number"<<endl;
    cin>>num;
    if ( prime(num)) //inline function call
        cout<<"Number is Prime " ;
    else
        cout<<"Number is not a Prime " ;
    getch();
}
```

OUTPUT:

```
Enter the input number
5
Number is Prime
```

Chapter-9

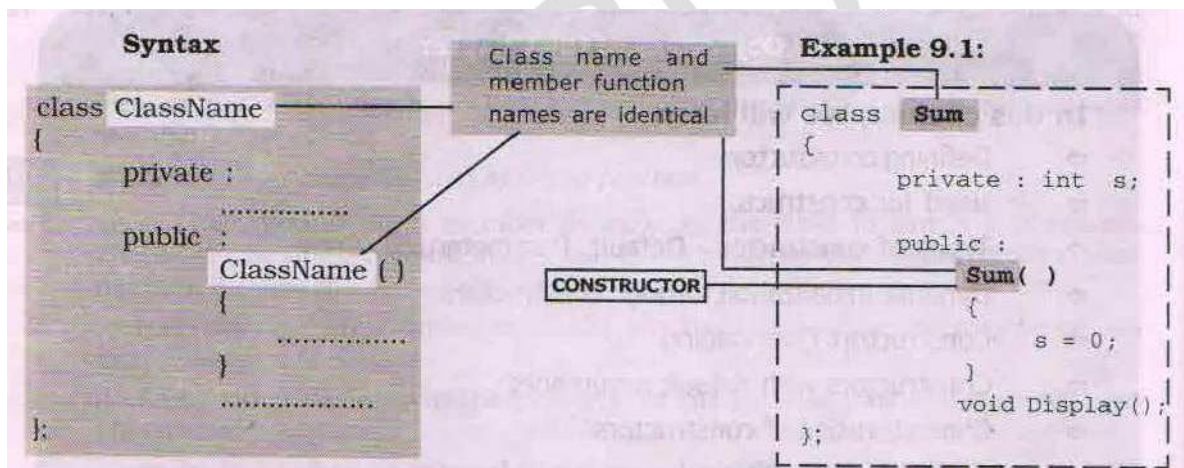
CONSTRUCTORS AND DESTRUCTORS

➤ Introduction:

- It is sometimes convenient if an object can initialize itself when it is first created, without the need to make a separate call to member functions.
- Automatic initialization is carried out using special member functions called constructors.

➤ Constructors:

- *A Constructor is a special member function that is called automatically when an object is created.*
- The purpose of a constructor is to mainly initialize the member variables of a class.
- The general syntax of a the constructor in C++ is:



In the above example class declaration, class **Sum** has a member function **Sum()** with the same name of the class and which provides initial value to its data member **s**.

➤ Characteristics of Constructor:

- The name of the constructor is the same as the name of the class.
Example: In the above class, name of the class is **Sum** and the function name is **Sum**.
- A Constructor, even though it is a function, has no return type, i.e. it is neither a value-returning function nor a void function.
Example: **Sum ()** function has no return type and not even void.
- The constructor should be declared in the **public** section.
- Constructors are executed automatically i.e. they are never invoked. They are executed when a class object is created.

Important
5 Marks

- A class can have more than one constructor. However all constructor of a class should have the same name.
- It is not possible to refer to the address of the constructors.
- The constructors make implicit calls to the operator new and delete when memory allocation is required.
- **Example: Program to demonstrate how constructor is automatically executed at the time of object creation.**

```
#include<iostream.h>
#include<conio.h>
class Student
{
    public:
        Student( )
        {
            cout<<"Constructor called automatically";
            cout<<"at the time of object creation"<<endl;
        }
};
void main( )
{
    Student S1;
    Student S2;
    Student S3;
}
```

OUTPUT:

```
Constructor called automatically at the time of object creation
Constructor called automatically at the time of object creation
Constructor called automatically at the time of object creation
```

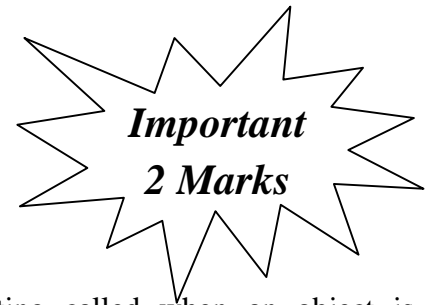
- **Example: Program to demonstrate how a constructor is use to initialize data member of an object.**

```
#include<iostream.h>
#include<conio.h>
class Number
{
    private:
        int a;
    public:
        Number ( )
        {
            cout<<"I am in the Constructor";
            a = 100;
        }
        void display( )
        {
            cout<<"Value of a is ="<<a;
        }
}
```

```
};
void main()
{
    Number N;
    N.display;
}
```

OUTPUT:

```
I am in the Constructor
Value of a is = 100
```



➤ **Need for a Constructor:**

- Constructors are named as constructors because they are getting called when an object is constructed.
- The use of a constructor can be cleverly done especially in those problems where it is necessary to initialize certain data members compulsorily.
- Instead of having separate member functions for initializing we can perform those operations inside the constructor itself.
- **Example: A Program to find the sum of N natural numbers using a class constructor.**

```
#include<iostream.h>
#include<conio.h>
class Sum
{
    private:
        int n, s;
    public:
        Sum ()
        {
            s = 0;
        }
        void readdata()
        {
            cout<<"Enter the input limit"<<endl;
            cin>>n;
        }
        void display()
        {
            for(int i =1; i<=n; i++)
                s = s + i;
            cout<<"Sum of Natural numbers ="<<s;
        }
};
void main()
{
    Sum S1;
    S1.readdata();
    S1.display();
}
```

OUTPUT:

```
Enter the input limit
10
Sum of Natural numbers = 55
```

➤ Types of constructor:

- Constructors are normally classified as follows:
 - Default Constructors.
 - Parameterized Constructors
 - Copy Constructors.



➤ Default Constructors:

- *A default constructor is a special member function which is invoked by the C++ compiler without any argument for initializing the object of a class.*
- It is also called as zero argument constructors.
- Some of the features of the default constructors are:
 - A default constructor function initializes the data member with no argument.
 - It can be explicitly written in the public section of the class.
 - In case, default constructor is not defined in a program, the C++ compiler automatically generates it in a program.
 - The purpose of the default constructor is to construct a default object of the class type.
- The general format of default constructor is as follows:

Syntax	Example
<pre>class Class_Name { public: Class_Name() { } };</pre>	<pre>class Number { public: Number() { n = 0; } };</pre>

- **Example: A program to display N natural numbers using a class default constructor.**

```
#include<iostream.h>
#include<conio.h>
class Number
{
    private:
        int n;
    public:
        Number ( )                //Default Constructor with no arguments
        {
            n = 0;
        }
        void readdata( )
        {
```

```

        cout<<"Enter the input limit"<<endl;
        cin>>n;
    }
    void display()
    {
        for( i =1; i<=n; i++)
            cout<<"Natural numbers ="<< i<<"\t";
    }
};
void main( )
{
    Sum S1;
    S1.readdata( );
    S1.display( );
}

```

OUTPUT:

Enter the input limit

10

Natural numbers = 1 2 3 4 5 6 7 8 9 10

- **Some disadvantages of default constructors are:**

- When many objects of the same class are created, all objects are initialized to same set of values by default constructors.
- It is not possible to initialize different objects with different initial values using default constructors.

- **Parameterized Constructors:**

- *A constructor that takes one or more arguments is called parameterized constructor.*
- Using this constructor, it is possible to initialize different objects with different values.
- Parameterized constructors are also invoked automatically, whenever objects with arguments are created. The parameters are used to initialize the objects.
- The keyword *inline* is used to define inline function.
- The general format of parameterized constructor is as follows:

Syntax	Example
<pre> class Class_Name { public: Class_Name(argu1, argu2....) { } }; </pre>	<pre> class MAX { public: MAX(int a, int b) { if (a > b) big = a; else big = b; } }; </pre>

- **Some of the features of the parameterized constructors are:**

- The parameterized constructors can be overloaded.
- For an object created with one argument, constructor with only one argument is invoked and executed.
- The parameterized constructor can have default arguments and default values.

- ✓ **Invoking Constructors:**

- A Constructor is automatically invoked by C++ compiler with an object declaration. The constructor can be invoked through the following methods.

- Implicit Call
- Explicit Call
- Initialization at the time of declaration with “ = “ operator.

**Important
5 Marks**

- ✓ **Implicit Call:**

- An Implicit call means the declaration of the object is followed by argument list enclosed in parenthesis.

- **Example: Program to initialize the data members using implicit declaration**

```
#include<iostream.h>
#include<conio.h>
class num
{
    private:
        int a, b;
    public:
        num ( int m, int n)                //Parameterized Constructor
        {
            a = m;
            b = n;
        }
        void display( )
        {
            cout<<" a = "<< a <<" b = "<< b;
        }
};
void main( )
{
    num obj1(10, 20);                //Implicit Call
    num obj2(40, 50);                //Implicit Call
    obj1.display( );
    obj2.display( );
}
```

OUTPUT:

a = 10 b = 20
a = 40 b = 50

✓ **Explicit Call:**

- In explicit call, declaration of an object is followed by assignment operator, constructor name and argument list enclosed in parenthesis.
- **Example: Program to initialize the data members using explicit declaration**

```
#include<iostream.h>
#include<conio.h>
class num
{
    private:
        int a, b;
    public:
        num ( int m, int n)                //Parameterized Constructor
        {
            a = m;
            b = n;
        }
        void display( )
        {
            cout<<" a = " << a <<" b = " << b;
        }
};
void main( )
{
    num obj1 = num(10, 20);                //Explicit Call
    num obj2 = num(40, 50);                //Explicit Call
    obj1.display( );
    obj2.display( );
}
```

OUTPUT:

a = 10 b = 20

a = 40 b = 50

✓ **Initialization of object during declaration with assignment operator “ = “:**

- This method is used for the constructor with exactly one argument. In this method declaration is followed by assignment operator and value to be initialized.
- **Example: Program to initialize objects using assignment operator.**

```
#include<iostream.h>
#include<conio.h>
class num
{
    private:
        int a;
    public:
        num ( int m)                        //Parameterized Constructor
        {
            a = m;
```

```

    }
    void display( )
    {
        cout<< a << endl ;
    }
};
void main( )
{
    num obj1 = 100
    num obj2 = 200
    cout<<"Object1 = " ;
    obj1.display( );
    cout<<"Object2 = " ;
    obj2.display( );
}

```

OUTPUT:

```

Object1 = 100
Object2 = 200

```

➤ **Copy Constructors:**

- *Copy constructor is a parameterized constructor using one object can be copied to another object.*
- Copy Constructors are used in the following situations:
 - To initialize an object with the values of already existing objects.
 - When objects must be returned as function values.
 - To state objects as by value parameters of a function.
- Copy Constructor can accept a single argument of reference to same class type. The argument must be passed as a constant reference type.
- The general format of copy constructor is as follows:

Syntax	Example
<pre> class Class_Name { public: Class_Name(Class_Name &ptr) { } }; </pre>	<pre> class Number { public: Number(int n) { a = n; } Number(Number & X) { a = X.a; cout<<"Copy Constructor invoked"; } }; </pre>

- **Note that:**
 - Copy constructor is not invoked explicitly.
 - Copy constructors are invoked automatically when a new object is created and equated to an already existing object in the declaration statement itself.

```
Example:      x      a1;           //Default Constructor
              x      a2 = a1;      //Copy Constructor
```

- When a new object is declared and existing object is passed as a parameter to it in the declaration, then also copy constructor is invoked.

```
Example:      x      a1(100, 200); //Parameterized Constructor
              x      a2(a1);       //Copy Constructor invoked
```

- When object is passed to a function using pass by value, copy constructor is automatically called.
- Copy constructor is invoked when an object returns a value.

- **Example: Program to find factorial of a number using copy constructor.**

```
#include<iostream.h>
#include<conio.h>
class copy
{
    private:
        int var;
    public:
        copy ( int temp)
        {
            var = temp;
        }
        int calculate( )
        {
            int fact, i;
            fact = 1;
            for( I =1; i<=var; i++)
                fact = fact * I;
            return fact;
        }
};
void main( )
{
    int n;
    cout<<"Enter the Number:";
    cin>>n;
    copy obj(n);
    copy cpy = obj;
    cout<<"Before Copying : "<< n<<"! =" <<obj.calculate( )<<endl;
    cout<<"After Copying : "<< n<<"! =" <<cpy.calculate( )<<endl;
}
```

OUTPUT:

```
Enter the Number: 5
Before Copying: 5! = 120
After Copying: 5! = 120
```

➤ Constructor Overloading:

- *A class has two or more constructor functions with the same name but different signatures are called Constructors Overloading.*
- Depending upon the type of argument, the constructors will be invoked automatically by the compiler to initialize the objects.
- **Example: Program to find simple interest using constructor overloading.**

```
#include<iostream.h>
#include<conio.h>
class simpleinterest
{
    private:
        float p, r, t, si;
    public:
        simpleinterest( )                //Default constructor
        {
        }
        simpleinterest(float x, float y, float z) //Parameterized Constructor
        {
            p = x;
            r = y;
            t = z;
        }
        void compute ( )
        {
            si = (p * t * r)/100;
            cout<<"Simple Interest is = "<< si;
        }
};
void main( )
{
    simpleinterest S1, S2(10000.0, 12.0, 2.0);
    S2.compute( );
}
```

OUTPUT:

Simple Interest is = 2400

➤ Destructors:

- *A destructor is special member function that is executed when an object of that class is destroyed.*
- Destroying an object means, de-allocating all the resources such as memory that was allocated for the object by the constructor.
- It will have like constructor, the name same as that of the class but preceded by a tilde (~).
- The general format of destructor is as follows:

Syntax	Example
<pre>class Class_Name { public: Class_Name(); ~ Class_Name(); };</pre>	<pre>class Counter { public: Counter() //Constructor { n = 0; } ~Counter () //Destructor { } };</pre>

- **Some of the characteristics of destructor are:**

- The destructor name must have the same name as the class preceded by a tilde (~).
- The destructor cannot take arguments therefore cannot be overloaded.
- The destructor has no return type.
- There can be only one destructor in each class.
- It should have public access in the class declaration.
- The destructor cannot be inherited.

**Important
5 Marks**

- **Example: Program to illustrate the use of destructors in C++.**

```
#include<iostream.h>
#include<conio.h>
class num
{
    private:
        int x;
    public:
        num(); //Default constructor
        void display();
        ~ num();
};
num :: num()
{
    cout<<"In Constructor: \n";
    x = 100;
}
num :: ~ num()
{
    cout<<"In Destructor";
}
void num :: display()
{
    cout <<"Value of X " << x <<endl;
```

```

}
void main( )
{
    num a;
    a.display( );
}

```

OUTPUT:

In Constructor:

Value of X = 100

In Destructor

CHAPTER 9 – Constructors and Destructors BLUE PRINT

VSA (1 marks)	SA (2 marks)	LA (3 Marks)	Essay (5 Marks)	Total
-	01 Question	-	01 Question	02 Question
-	Question No 14	-	Question No 33	07 Marks

Important Questions**5 Marks Question:**

1. What is Constructor? Give the rules for writing a constructor function.
2. What is default constructor? Write a program to illustrate it.
3. Explain parameterized constructor with syntax and example.
4. Mention the different methods by which constructor are invoked. Explain anyone with an illustrative example.
5. Explain the features of copy constructor.
6. Explain destructor with syntax and example.
7. Write a C++ program to find the sum of the series $1+X+X^2+\dots+X^n$ using constructor:

```

#include<iostream.h>
#include<conio.h>
class copy
{
    private:
        int x, n;
    public:
        int compute;
        copy(int xx, int nn)
        {
            x = xx;
            n = nn;
        }
};

```

```
int copy :: compute( )
{
    int nextterm;
    int sum =1;
    nextterm = x;
    for (int i=1; i<=n; i++)
    {
        sum = sum + nextterm;
        nextterm = nextterm * x;
    }
    return sum;
}
void main( )
{
    int n, x ;
    clrscr();
    cout<<"Enter the X and N values"<<endl;
    cin>>x >> n;
    copy obj ( x, n);
    copy cpy = obj;
    cout<<"Object 1: Sum of the series:" <<obj.compute( )<<endl;
    cout<<"Object 2: Sum of the series:" <<cpy.compute( )<<endl;
    getch();
}
*****
```


Chapter-10

INHERITANCE

➤ Introduction:

- Inheritance is another important aspect of object oriented programming.
- C++ allows the user to create a new class (derived class) from an existing class (base class).

➤ Inheritance:

- *Inheritance is the capability of one class to inherit properties from another class.*
- **Base Class:** It is the class whose properties are inherited by another class. It is also called Super class.
- **Derived Class:** It is the class that inherits the properties from base class. It is also called Sub class.

➤ Need of Inheritance:

- Suppose X is a class already defined and we need to redefine another class Y has same properties of X and in addition its own.
- Suppose if we use direct option without using inheritance, it has following problems.
 - Code written in X is repeated again in Y which leads to unnecessary wastage of memory.
 - Testing to be done separately for both class X and class Y leads to waste of time.
- The above problem can be solved by using the concept of inheritance.
- If we use the code of X even in Y without rewriting it. The class Y inherits all the properties of X.
- The class X is called base class and the class Y is called derived class.
- **The main advantages of Inheritance are:**
 - Reusing existing code
 - Faster development time
 - Easy to maintain
 - Easy to extend
 - Memory Utilization
- **The main disadvantage of Inheritance are:**
 - Inappropriate use of inheritance makes programs more complicated.
 - Calling member functions using objects creates more compiler overheads.

➤ Defining Derived Classes:

- A derived class is a composite class – it inherits members from the base class and adds member of its own.
- The general form of the derived class is given below.

IMAGE

- Here,
 - class → Keyword
 - derived_class_name → Name of the derived class
 - : → Shows the derivation from the base class.
 - Visibility Mode → Specifies the type of derivation
 - base_class_name → Name of the base class.
 - The use of a constructor can be cleverly done especially in those problems where it is necessary to initialize certain data members compulsorily.

• Example:

Public Derived Class	Private Derived Class	Protected Derived Class
<pre>class father //Base class { private: char name[10]; public: char caste[10]; int age; void readdata(); }; class son : public father { private: char gender[5]; public: void display(); };</pre>	<pre>class father //Base class { private: char name[10]; public: char caste[10]; int age; void readdata(); }; class son : private father { private: char gender[5]; public: void display(); };</pre>	<pre>class father //Base class { private: char name[10]; public: char caste[10]; int age; void readdata(); }; class son : protected father { private: char gender[5]; public: void display(); };</pre>

➤ Visibility mode:

- Visibility mode can be public, private or protected. The private data of base class cannot be inherited.
 - **public:** If inheritance is done in public mode, public members of the base class become the public member of derived class and protected member of base class become the protected member of derived class..

- **private:** If inheritance is done in a private mode, public and protected members of base class become the private members of derived class.
- **protected:** If inheritance is done in a protected mode, public and protected members of base class become the protected members of the base class.

		Derived Class		
		public	private	protected
Base class	public	public	private	protected
	private	Not inherited	Not inherited	Not inherited
	protected	protected	private	protected

➤ Public Inheritance:

- When a base class is inherited as public, all public members of the base class become public members of derived class.
- The private members of the base class remain private to that class, and are not accessible by members of the derived class.
- Example: A program illustrates public inheritance.

```
#include<iostream.h>
#include<conio.h>
class shape //Base Class
{
    public:
        int side1, side2;
};
class rectangle : public shape //Derived Class
{
    public:
        int area;
        void compute( )
        {
            area = side1 * side2;
        }
};
void main( )
{
    rectangle R; // R is the object of derived class
    R.side1 = 5; // Data directly accessed by object
    R.side2 = 6;
    R.compute( );
    cout<< "Area of the rectangle = " <<R.area;
}
```



OUTPUT:

Area of the rectangle = 30

➤ Private Inheritance:

- When a base class is inherited as private, then all public and protected members of the base class become private members of derived class.
- This means that they are still accessible by member of the derived class, but cannot be accessed by other parts of the program.
- Example: A program illustrates private inheritance.

```

#include<iostream.h>
#include<conio.h>
class shape //Base Class
{
    protected:
        int side1, side2;
    public:
        int area;
        void compute( )
        {
            area = side1 * side2;
        }
};
class rectangle : private shape //Derived Class
{
    public:
        void readdata( )
        {
            cout << " Enter the input first side :";
            cin>>side1;
            cout << " Enter the input second side :";
            cin>>side2;
        }
        void display( )
        {
            compute(); // Calling base class
            cout<< "Area of the rectangle = " <<area;
        }
};
void main( )
{
    rectangle R;
    R.readdata( );
    R.display( );
}

```

OUTPUT:

```

Enter the input first side : 9
Enter the input second side : 5
Area of the rectangle = 45

```

➤ Protected Inheritance:

- When a base class is inherited as protected, then all public and protected members of the base class become protected members of derived class.
- The private data members of base class are not visible to derived class.
- They can only be accessed through public and protected member functions of base class.
- Example: A program illustrates protected inheritance with the base class having protected and public members.

```

#include<iostream.h>
#include<conio.h>
class shape //Base Class
{
    protected:
        int side1, side2;
    public:
        int compute()
        {
            return(side1 * side2);
        }
};
class rectangle : protected shape //Derived Class
{
    public:
        void readdata()
        {
            cout << " Enter the input first side :";
            cin>>side1;
            cout << " Enter the input second side :";
            cin>>side2;
        }
        void display()
        {
            cout<< "Area of the rectangle = " <<compute();
        }
};
void main()
{
    rectangle R;
    R.readdata();
    R.display();
}

```

OUTPUT:

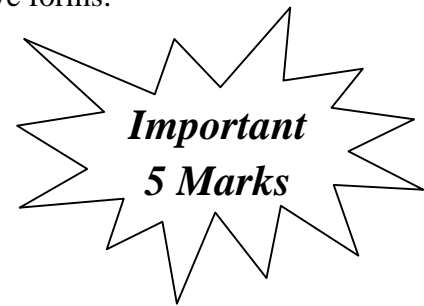
```

Enter the input first side : 7
Enter the input second side : 8
Area of the rectangle = 56

```

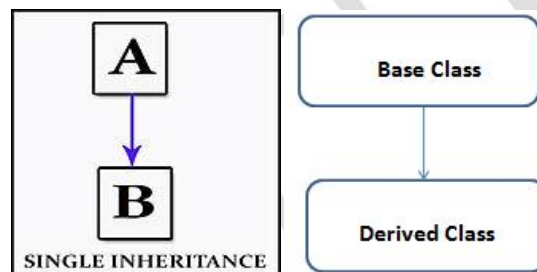
➤ Types of Inheritance:

- Based on the relationship, inheritance can be classified into five forms:
 - Single Inheritance
 - Multilevel Inheritance
 - Multiple Inheritance
 - Hierarchical Inheritance
 - Hybrid Inheritance



➤ Single Inheritance:

- *Single Inheritance is the process of creating a new class from existing class base class.*
- It is very common in inheritance that a class is derived from the base class.
- The data members and memberfunction of the base class are data member and member function of the derived class.



- A derived class with single inheritance is declared as follows:

```
class Base_Class
{
    .....
};
class Derived_class : public Base_calss
{
    .....
};
```

- **Example: Program to illustrate single level inheritance.**

```
#include<iostream.h>
#include<conio.h>
class base
{
    private:
        int rollno;
        char name[10];
    public:
        void read( )
```

```

    {
        cout << " Enter Roll Number and Name " << endl;
        cin >> rollno >> name;
    }
void display()
{
    cout << " Roll No : " << rollno << endl;
    cout << " Name : " << name << endl;
}
};
class derived : public base
{
    private:
        int m1, m2, t;
    public:
        void read1()
        {
            cout << " Enter Maths and Computer marks " << endl;
            cin >> m1 >> m2;
            t = m1 + m2;
        }
        void display1()
        {
            cout << " Maths : " << m1 << endl;
            cout << " Computer : " << m2 << endl;
            cout << "Total Marks : " << t << endl;
        }
};

void main()
{
    derived obj;
    clrscr();
    obj.read();
    obj.read1();
    obj.display();
    obj.display1();
    getch();
}

```

OUTPUT:

```

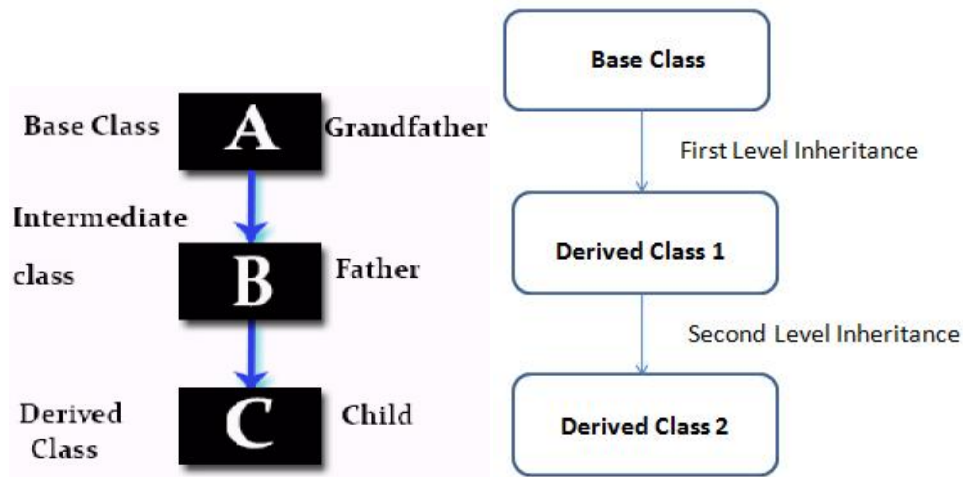
Enter Roll Number and Name
1234 RAM
Enter Maths and Computer marks
80    90
Roll No : 1234
Name : RAM
Maths : 80
Computer : 90
Total Marks : 170

```

➤ Multilevel Inheritance:

- *Derivation of a class from another derived class is called multilevel inheritance.*
- In the figure class A is the base class for class AB and class AB is the base class for class ABC.
- The class AB provides a link for the inheritance between A and ABC, and is known as

intermediate base class.



- A derived class with multilevel inheritance is declared as follows:

```

class A
{
    .....
};
class AB : public A
{
    .....
};
class ABC : public AB
{
    .....
};
  
```

- **Example: Program to illustrate multilevel inheritance.**

```

#include<iostream.h>
#include<conio.h>
class A
{
    public:
        void displayA()
        {
            cout << " Base class A"<<endl;
        }
};
class AB : public A
{
    public:
  
```



```

void displayAB()
{
    cout << " Intermediate Base class AB"<<endl;
    cout << " Derived from A" << endl;
}
};
class ABC : public AB
{
    public:
    void displayABC()
    {
        cout << " Derived Class ABC"<<endl;
        cout << " Derived from AB" << endl;
    }
    void output()
    {
        displayA();
        displayAB();
        displayABC();
    }
};
void main()
{
    ABC obj;
    clrscr();
    obj.output();
    getch();
}

```

OUTPUT:

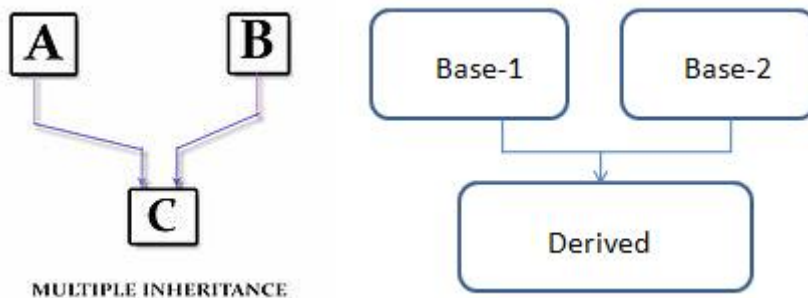
```

Base Class A
Intermediate Base Class AB
Derived from A
Derived Class ABC
Derived from AB

```

➤ Multiple Inheritance:

- A class can be derived from more than one base class is known as multiple inheritance.



- A derived class with multiple inheritance is declared as follows:

```

class A //Base Class A
{
    .....
};

```

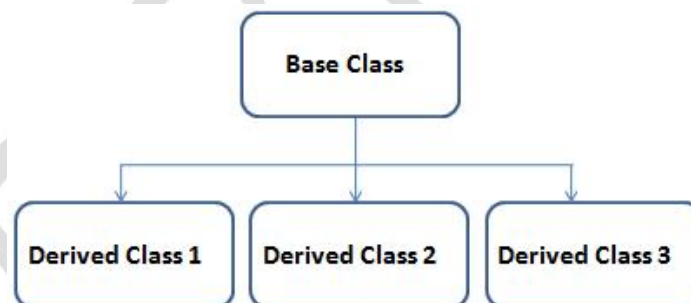
```

class B                //Base Class B
{
    .....
};
class C                //Base Class C
{
    .....
};
class Derived_Class : public A, private B, protected C
{
    .....                //Members of derived class
};

```

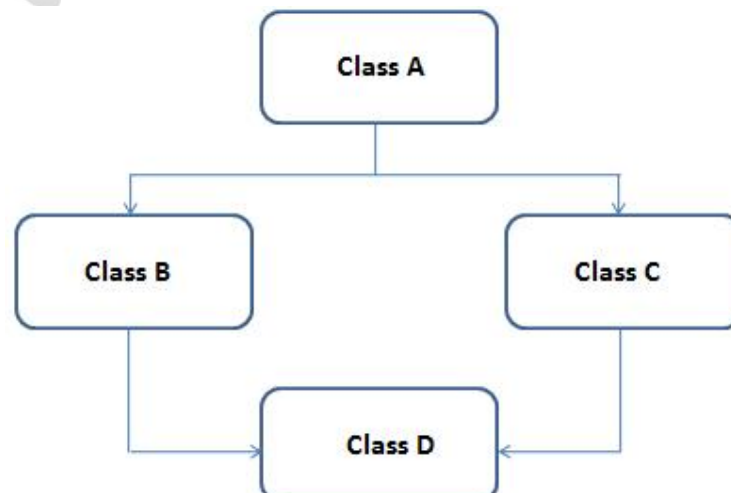
➤ Hierarchical Inheritance:

- *If a number of classes are derived from a single base class, it is called as hierarchical inheritance.*
- Hierarchical model exhibits top down approach by breaking up a complex class into simpler class.



➤ Hybrid Inheritance:

- *Hybrid Inheritance is combination of Hierarchical and multilevel inheritance.*



➤ Virtual Base Classes:

- When two or more objects are derived from a common base class, we can prevent multiple copies of the base class being present in an object derived from those objects by declaring the base class as virtual when it is being inherited.
- Such a base class is known as *virtual base class*.
- This can be achieved by preceding the base class name with the word virtual.
- Example:

Class A

```
{
    -----;
    -----;
};
```

class B : virtual public A

```
{
    -----;
    -----;
};
```

class C : virtual public A

```
{
    -----;
    -----;
};
```

class D : public B, public C

```
{
    -----;
    -----;
};
```

➤ Abstract Class:

- An abstract class is one that is not used to create objects.
- An abstract class is designed only to act as a base class (to be inherited by other classes).

➤ Constructor and Destructors in derived classes:

- *A destructor is special member function that is executed when an object of that class is destroyed.*

- Destroying an object means, de-allocating all the resources such as memory that was allocated for the object by the constructor.
- It will have like constructor, the name same as that of the class but preceded by a tilde (~).
- **Example: Program to illustrate the use of destructors in C++.**

```

class Base
{
    public:
        Base( );                //Default constructor
        {
            cout<<"Inside Base Constructor"<<endl;
        }
        ~ Base( );             //Destructor
        {
            cout<<"Inside Base Destructor"<<endl;
        }
};
class Derived : public Base
{
    public:
        Derived( );           //Default constructor
        {
            cout<<"Inside Derived Constructor"<<endl;
        }
        ~ Derived( );         //Destructor
        {
            cout<<"Inside Derived Destructor"<<endl;
        }
};
void main( )
{
    Derived x;
    x.display( );
}

```

OUTPUT:

```

Inside Base Constructor
Inside Derived Constructor
Inside Derived Destructor
Inside Base Destructor

```

CHAPTER 10 – Inheritance BLUE PRINT

VSA (1 marks)	SA (2 marks)	LA (3 Marks)	Essay (5 Marks)	Total
-	-	-	01 Question	01 Question
-	-	-	Question No 34	05 Marks

Important Questions

5 Marks Question:

1. What is Inheritance? Explain any two types of Inheritance.
2. What are visibility modes? Explain.
3. What is the difference between public, private and protected access specifiers?
4. Explain single inheritance with a suitable C++ program.
5. What is Multilevel Inheritance? Explain with a suitable program example.
6. What is virtual base class? Give example.
7. What are the advantages of Inheritance? (Any five)

Chapter-11

POINTERS

➤ Introduction:

- Pointers are a powerful concept in C++ and have the following advantages.
 - i. It is possible to write efficient programs.
 - ii. Memory is utilized properly.
 - iii. Dynamically allocate and de-allocate memory.

➤ Memory Utilization of Pointer:

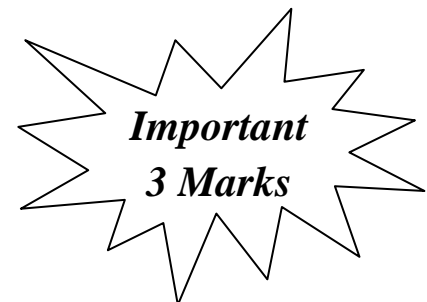
- Memory is organized as an array of bytes. A byte is basic storage and accessible unit in memory.
- Each byte is identifiable by a unique number called address.
- We know that variables are declared before they are used in a program. Declaration of a variable tells the compiler to perform the following.
 - Allocate a location in memory. The number of location depends on data type.
 - Establish relation between address of the location and the name of the variable.
- Consider the declaration, `int num;`
- This declaration tells the compiler to reserve a location in memory. We know that size of int type is two bytes. So the location would be two bytes wide.

Address	Num
100	15
101	

- In the figure, num is the variable that stores the value 15 and address of num is 100. The address of a variable is also an unsigned integer number. It can be retrieved and stored in another variable.

➤ Pointer:

- *A pointer is a variable that holds a memory address of another variable.*
- The pointer has the following advantages.
 - Pointers save memory space.
 - Dynamically allocate and de-allocate memory.
 - Easy to deal with hardware components.
 - Establishes communication between program and data.
 - Pointers are used for file handling.
 - Pointers are used to create complex data structures such as linked list, stacks, queues trees



and graphs.

➤ **Pointer Declaration:**

- Pointers are also variables and hence, they must be defined in a program like any other variable.
- The general syntax of pointer declaration is given below.

Syntax: **Data_Type *Ptr_Variablename;**

- Where,
 - Data_Type is any valid data type supported by C++ or any user defined type.
 - Ptr_Variablename is the name of the pointer variable. The presence of ‘*’ indicates that it is a **pointer variable**.
- Defining pointer variables:
 - int *iptr; iptr is declared to be a pointer variable of int type.
 - float *fptr; fptr is declared to be a pointer variable of float type.
 - char *cptr; cptr is declared to be a pointer variable of character type.

➤ **Pointer Initialization:**

- Once we declare a pointer variable, we must make it to point to something.
- We can do this by assigning or initializing to the pointer the address of the variable you want to point to as in: **iptr = #**
- The ‘&’ is the **address operator** and it represents address of the variable.
- Example: A program to display the content of num and the address of the variable num using a pointer variable.

```
#include<iostream.h>
void main( )
{
    int num;                        // normal integer variable
    int *iptr;                     // Pointer declaration, pointing to integer data

    num = 574;                     // assign value to num
    iptr = &num                   // assign address of num to int pointer

    cout<<" Value of num is :"<<num<<endl;
    cout<<" Address of num is :"<<iptr<<endl;
}
```

➤ **The address of operator (&):**

- ‘&’ is a unary operator that returns *the memory address of its operand*.
- For example, if var is an integer variable, then &var is its address.
- We should read ‘&’ operator as “the address-of” which means &var will be read as “the address of

var”.

- Example:

```
int num = 25;
int *iptr;
iptr = &num;           //The address of operator &
```

➤ Pointer Operator or Indirection Operator (*):

- The second operator is indirection operator ‘*’, and it is the complement of ‘&’.
- It is a unary operator that returns *the value of the variable located* at the address specified by its operand.
- Example:

```
int num = 25;
int *iptr;           //Pointer Operator (Indirection Operator *)
iptr = &num;
```

- Example: A program executes the two operations.

```
#include<iostream.h>
#include<conio.h>
void main( )
{
    int num;
    int *iptr;
    int val;

    num = 300;
    iptr = & num;
    val = *iptr;

    cout<<" Value of num is :"<<num<<endl;
    cout<<" Value of pointer :"<<iptr<<endl;
    cout<<" Value of val :"<<val<<endl;
}
```

OUTPUT:

```
Value of num is : 300
Value of pointer : 0xbff64494
Value of val : 300
```

➤ Pointer Arithmetic:

- We can perform arithmetic operations on a pointer just as you can a numeric value.
- There are four arithmetic operators that can be used on pointers:
 - Increment ++
 - Decrement --
 - Addition +

- Subtraction -

- Example:

```
int num, *iptr;
num = 9;
iptr = &num;
iptr++;
cout<<iptr;
```

iptr →	1200	9
	1201	
iptr++ →	1202	
	1203	

- The following operation can be performed on pointers.
 - We can add integer value to a pointer.
 - We can subtract an integer value from a pointer.
 - We can compare two pointers, if they point the elements of the same array.
 - We can subtract one pointer from another pointer if both point to the same array.
 - We can assign one pointer to another pointer provided both are of same type.
- The following operations cannot be performed on pointers.
 - Addition of two pointers.
 - Subtraction of one pointer from another pointer when they do not point to the same array.
 - Multiplication of two pointers.
 - Division of two pointers.
- A program to illustrate the pointer expression and pointer arithmetic.

```
#include<iostream.h>
#include<conio.h>
void main( )
{
    int a, b, x, y;
    int *ptr1, *ptr2;
    a = 30;
    b = 6;
    ptr1 = &a
    ptr2 = &b;
    x = *ptr1 + *ptr2 - 6;
    y = 6 - *ptr1 / *ptr2 + 30;

    cout<<"Address of a = "<<ptr1<<endl;
    cout<<"Address of b = "<<ptr2<<endl;
    cout<<"a = "<<a<<"b = "<<b<<endl;
    cout<<"x = "<<x<<"y = "<<y<<endl;

    *ptr1 = *ptr1 + 70;
    *ptr2 = *ptr2 * 2;
```

OUTPUT:

```
Address of a = 65524
Address of b = 65522
a = 30          b = 6
x = 30          y = 6
a = 100         b = 12
```

```

    cout<<"a = "<<a<<"b = "<<b<<endl;
}

```

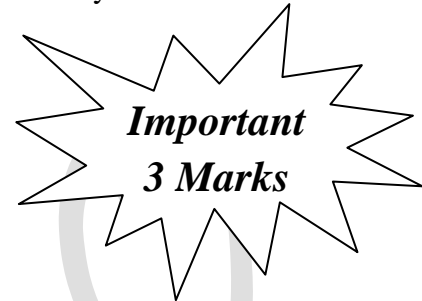
➤ Pointers and Arrays:

- There is a close relationship between array and pointers in C++.
- Consider the following program which prints the elements of an array A.

```

#include<iostream.h>
void main( )
{
    int A[5] = { 15, 25, 67, 83, 12};
    for (int i = 0; i<5; i++)
        cout<<A[i]<<"\t";
}

```



- Output of the above program is: 15 25 67 83 12
- When we declare an array, its name is treated as a constant pointer to the first element of the array.
- This is also known as the **base address of the array**.
- In other words base address is the address of the first element in the array or the address of a[0].
- If we use constant pointer to print array elements.

```

#include<iostream.h>
void main( )
{
    int A[5] = { 15, 25, 67, 83, 12};
    cout<< *(A)        <<"\t";
    cout<< *(A+1)     <<"\t";
    cout<< *(A+2)     <<"\t";
    cout<< *(A+3)     <<"\t";
    cout<< *(A+4)     <<"\t";
}

```

Constant Pointer →

A	17500	15	A[0]
	17501		
A+1	17502	25	A[1]
	17503		
A+2	17504	67	A[2]
	17505		
A+3	17506	83	A[3]
	17507		
A+4	17508	12	A[4]
	17509		

- Output of the above program is: 15 25
67 83 12
- Here the expression *(A+3) has exactly same effect as A[3] in the program.
- The difference between constant pointer and the pointer variable is that the constant pointer cannot be incremented or changed while the pointer to an array which carries the address of the first element of the array may be incremented.
- The following example shows the relationship between pointer and one dimensional array.

```

#include<iostream.h>
void main( )
{

```

```

int a[10], i, n;
cout<<"Enter the input for array";
cin>>n;
cout<<"Enter array elements:";
for(i=0; i<n; i++)
    cin>>*(a+i);
cout<<"The given array elements are .:";
for(i=0; i<n; i++)
    cout<<"\t"<<*(a+i);
getch( );
}

```

OUTPUT:

```

Enter the input for array 5
Enter array elements
1    2    3    4    5
The given array elements are :
1    2    3    4    5

```

➤ **Array of Pointers:**

- There is an array of integers; array of float, similarly there can be an array of pointers.
- “An array of pointer means that it is a collection of address”.
- The general form of array of pointers declaration is:

int *pint[5];

- The above statement declares an array of 5 pointers where each of the pointer to integer variable.
- Example: Program to illustrate the array of pointers of isolated variables.

```

#include<iostream.h>
#include<conio.h>
void main( )
{
    int *pint[5];
    int a = 10, b = 20, c = 30, d=40, e=50;
    pint[0] = &a;
    pint[1] = &b;
    pint[2] = &c;
    pint[3] = &d;
    pint[4] = &e;

    for( int i=0; i<=4; i++)
        cout<<"Value " << *pint[i]<<" stored at "<<pint[i]<<endl;
}

```

OUTPUT:

```

Value 10 stored at 17500
Value 20 stored at 17502
Value 30 stored at 17504
Value 40 stored at 17506
Value 50 stored at 17508

```

➤ **Pointers and strings:**

- String is sequence of characters ends with null ('\0') character.
- C++ provides two methods of declaring and initializing a string.
- **Method 1:**

char str1[] = "HELLO";

- When a string is declared using an array, the compiler reserves one element longer than the

number of characters in the string to accommodate NULL character.

- The string `str1[]` is 6 bytes long to initialize its first 5 characters `HELLO\0`.

- **Method 2:**

```
char *str2 = "HELLO";
```

- C++ treats string constants like other array and interrupts a string constant as a pointer to the first character of the string.
- This means that we can assign a string constant to a pointer that point to a char.
- Example: A program to illustrate the difference between strings as arrays and pointers.

```
#include<iostream.h>
#include<conio.h>
void main( )
{
    char str1[ ] = "HELLO";
    char *str2 = "HELLO";
    cout<<str1<<endl;
    cout<<str2<<endl;
    str2++;
    cout<<str2<<endl;
}
```

OUTPUT:

```
HELLO
HELLO
ELLO
```

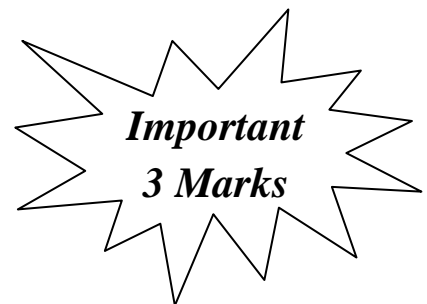
➤ **Pointers as Function Parameters:**

- A pointer can be a parameter. It works like a reference parameter to allow change to argument from within the function.

```
void swap(int *m, int *n)
{
    int temp;
    temp = *m;
    *m = *n;
    *n = temp;
}
void swap(&num1, & num2);
```

➤ **Pointers and Functions:**

- A function may be invoked in one of two ways :
 - Call by value
 - Call by reference
- The second method call by reference can be used in two ways :
 - By passing the references
 - By passing the pointers
- Reference is an alias name for a variable.



- For Example:


```
int m = 23;
int &n = m;
int *p;
p = &m;
```
- Then the value of m i.e. 23 is printed in the following ways :


```
cout <<m; // using variable name
cout << n; // using reference name
cout << *p; // using the pointer
```

✓ Invoking Function by Passing the References :

- When parameters are passed to the functions by reference, then the formal parameters become references (or aliases) to the actual parameters to the calling function.
- That means the called function does not create its own copy of original values, rather, it refers to the original values by different names i.e. their references.
- For example the program of swapping two variables with reference method:

```
#include<iostream.h>
void main()
{
    void swap(int &, int &);
    int a = 5, b = 6;
    cout << "\n Value of a : " << a << " and b : " << b;
    swap(a, b);
    cout << "\n After swapping value of a : " << a << "and b : " << b;
}
void swap(int &m, int &n)
{
    int temp;
    temp = m;
    m = n;
    n = temp;
}
```

OUTPUT:

```
Value of a : 5 and b : 6
After swapping value of a : 6 and b : 5
```

✓ Invoking Function by Passing the Pointers:

- When the pointers are passed to the function, the addresses of actual arguments in the calling function are copied into formal arguments of the called function.
- That means using the formal arguments (the addresses of original values) in the called function; we can make changing the actual arguments of the calling function.
- For example the program of swapping two variables with Pointers :


```
#include<iostream.h>
```

```

void main()
{
    void swap(int *m, int *n);
    int a = 5, b = 6;
    cout << "\n Value of a : " << a << " and b : " << b;
    swap(&a, &b);
    cout << "\n After swapping value of a : " << a << "and b : " << b;
}
void swap(int *m, int *n)
{
    int temp;
    temp = *m;
    *m = *n;
    *n = temp;
}

```

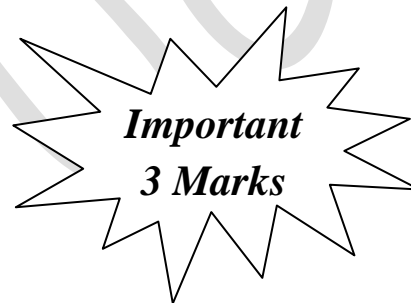
OUTPUT:

Value of a : 5 and b : 6

After swapping value of a : 6 and b : 5

➤ **Memory Allocation of pointers:**

- Memory Allocation is done in two ways:
 - Static Allocation of memory
 - Dynamic allocation of memory.

✓ **Static Allocation of Memory:**

- Static memory allocation refers to the process of allocating memory during the compilation of the program i.e. before the program is executed.
- Example:

```
int a; // Allocates 2 bytes of memory space during the compilation.
```

✓ **Dynamic Allocation of Memory:**

- Dynamic memory allocation refers to the process of allocating memory during the execution of the program or at run time.
- Memory space allocated with this method is not fixed.
- C++ supports dynamic allocation and de-allocation of objects using **new** and **delete** operators.
- These operators allocate memory for objects from a pool called the **free store**.
- The new operator calls the special function operator new and delete operators call the special function operator delete.

❖ **new operator:**

- We can allocate storage for a variable while program is running by using new operator.
- It is used to allocate memory without having to define variables and then make pointers point to them.

- The following code demonstrates how to allocate memory for different variables.
- To allocate memory type integer

```
int *pnumber;
pnumber = new int;
```
- The first line declares the pointer, pnumber. The second line then allocates memory for an integer and then makes pnumber point to this new memory.
- To allocate memory for array, double *dptr = new double[25];
- To allocates dynamic structure variables or objects, student sp = new student;

❖ delete Operator:

- The delete operator is used to destroy the variables space which has been created by using the new operator dynamically.
- Use delete operator to free dynamic memory as : delete iptr;
- To free dynamic array memory: delete [] dptr;
- To free dynamic structure, delete structure;

✓ Difference between Static Memory Allocation and Dynamic Memory Allocation:

Sl no	Static Memory Allocation	Dynamic Memory Allocation
1	Memory space is allocated before the execution of program.	Memory space is allocated during the execution of program.
2	Memory space allocated is fixed	Memory space allocated is not fixed
3	More memory space is required	Less memory space is required.
4	Memory allocation from stack area	Memory space form heap area.

✓ Free store (Heap memory):

- Free store is a pool of memory available to **allocated and de-allocated storage** for the objects during the **execution of the memory**.

✓ Memory Leak:

- If the objects, that are allocated memory dynamically, are not deleted using delete, the memory block remains occupied even at the end of the program.
- Such memory blocks are known as orphaned memory blocks.
- These orphaned memory blocks when increases in number, bring adverse effect on the system. This situation is called *memory leak*.

➤ Pointers and Structures:

- We can create pointers to structures variable.

```
struct student
{
    int roll_no;
    float fee;
};
student s;
student * sp = &s;
(*sp).roll_no = 14;
```

- The above statement can be written using the operator as →

```
sp->roll_no = 14;
```

➤ Pointers and Objects:

- The Pointers pointing to objects are referred to as object pointers.

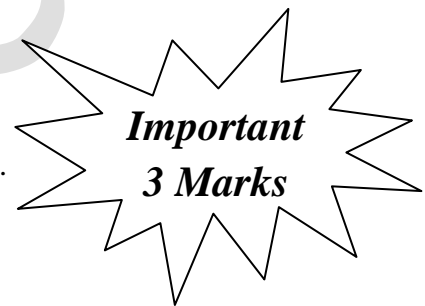
- **Declaration of pointers to object**

```
class_name * object-pointer;
```

- Here class_name is the name of the class, object-pointer is the pointer to an object of this class type.

- Example: employee * eptr

```
#include<iostream.h>
#include<conio.h>
class employee
{
    private:
        int empno;
        char name[20];
        float salary;
    public:
        void read( )
        {
            cout<<"Enter the Employee Number, Name and Salaray"<<endl;
            cin>>empno>>name>>salary;
        }
        void display( )
        {
```




```

        cout<<"Employee Number:"<<empno;
        cout<<"Employee Name:"<<name;
        cout<<"Employee Salary:"<<salary;
    }
};
void main( )
{
    employee e, * ep;
    ep = &e;
    clrscr( );
    ep → read( );
    ep → display ( );
    getch( );
}

```

- Here, employee is an already defined class. When accessing members of the class an object pointer, the arrow operator (→) is used instead of dot (.) operator.

➤ this pointers:

- Every object in C++ has access to its own address through an important pointer called this pointer.
- The “this pointer” is an implicit parameter to all member functions.
- Therefore, inside a member function, this may be used to refer to the invoking object.

➤ **Program 12:** Create a class containing the following data members Register_No, Name and Fees. Also create a member function to read and display the data using the concept of pointers to objects.

```

#include<iostream.h>
#include<conio.h>
class Student
{
    private:
        long regno;
        char name[20];
        float fees;
    public:
        void readdata( );
        void display( );
};
void Student::readdata( )
{
    cout<<"Enter the Register Number:"<<endl;
    cin>>regno;
}

```

```

    cout<<"Enter the Student Name:"<<endl;
    cin>>name;
    cout<<"Enter the Fees:"<<endl;
    cin>>fees;
}
void Student::display( )
{
    cout<<"Register Number      : "<<regno<<endl;
    cout<<"Student Name   : "<<name<<endl;
    cout<<"Fees           : "<<fees<<endl;
}
void main( )
{
    Student *S;                // Create a pointer to point Student object
    clrscr( );
    S->readdata( );            // Access Student data member using a pointer
    S->display( );             // Display data using a pointer
    getch( );
}

```

OUTPUT 1:

```

Enter the Register Number:
243850
Enter the Student Name:
Keerthi
Enter the Fees:
14050
Register Number : 243850
Student Name    : Keerthi
Fees           : 14050

```

OUTPUT 2:

```

Enter the Register Number:
12345
Enter the Student Name:
Akash
Enter the Fees:
25000
Register Number : 12345
Student Name    : Akash
Fees           : 25000

```

CHAPTER 11 – POINTERS BLUE PRINT

VSA (1 marks)	SA (2 marks)	LA (3 Marks)	Essay (5 Marks)	Total
01 Question	-	01 Question	-	02 Question
Question No 5	-	Question No 22	-	04 Marks

Important Questions**1 Marks Question:**

1. Define pointer. [June 2016]
2. Write the declaration syntax for a pointer. [March 2015]
3. How do we initialize a pointer? [March 2016]
4. Write any one advantage of pointers. [June 2015, March 2017]
5. What is the purpose of new operator in C++? [June 2017]
6. What is a pointer variable?

7. Which is the address operator?
8. What is pointer operator?
9. What is static memory & dynamic memory?
10. What is free store?

3 Marks Question:

1. What are the advantages of pointers? [June 2016]
2. What are the operations performed on pointers? [March 2015, June 2017]
3. What is array of pointer? Give example. [June 2015]
4. Explain the new and delete operator in pointers. [March 2016]
5. Define: [March 2017]
 - a. Pointer.
 - b. Static memory allocation.
 - c. Dynamic memory allocation
6. What is the relationship between pointers and arrays?
7. Explain with example call by reference.
8. Distinguish between static and dynamic memory allocation.
9. What is the relationship between pointers and objects? Give an example
10. Explain the use of "this pointer".

Chapter-12

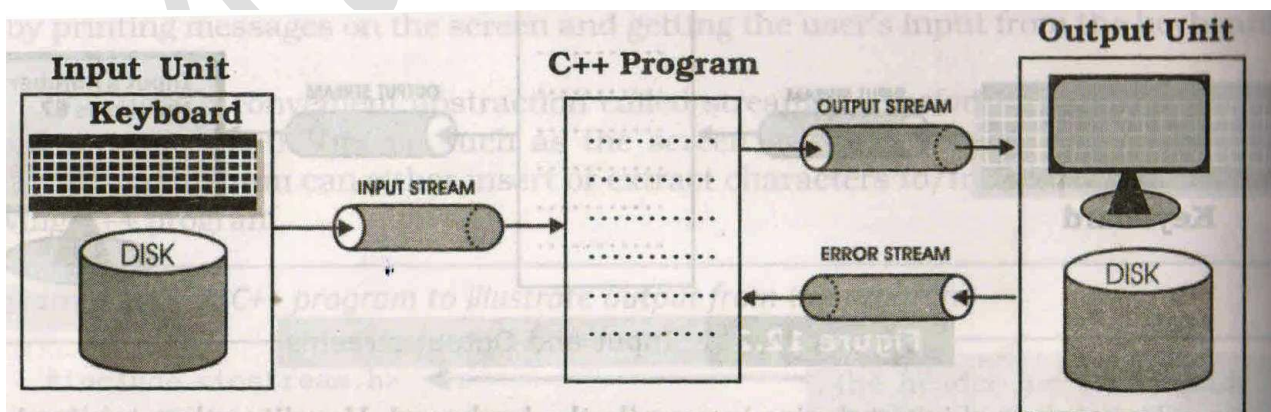
DATA FILE HANDLING

➤ Introduction:

- A file is a collection of related data stored in a particular area on the disk.
- Programs can be designed to perform the read and write operations on these files.
- In general a file is a sequence of bits, bytes, lines or records whose meaning is defined by its user.
- C++ I/O occurs in streams, which are sequence of bytes.
- If bytes flows from device like a keyboard, a disk drive, etc to main memory, this is called **input operation**.
- If bytes flow from main memory to devices like a display screen, a printer etc. this is called **output operation**.
- In C++, file input/output facilities are implemented through a header file **fstream.h**.

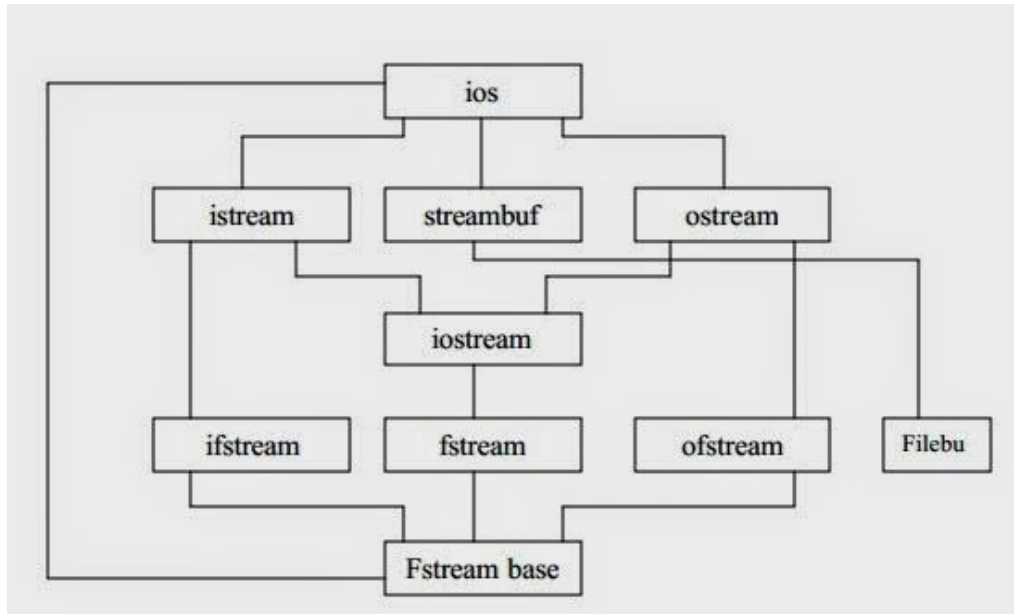
➤ Stream in C++:

- A stream is sequence of bytes. In C++, a stream is a general name given to flow of data.
- Different streams are used to represent different kinds of data flow.
- The three streams in C++ are as follows.
 - **Input Stream:** The stream that supplies data to the program is known as input stream.
 - **Output Stream:** The stream that receives data from the program is known as output stream.
 - **Error Stream:** Error streams basically an output stream used by the programs to the file or on the monitor to report error messages.



➤ **fstream.h header file:**

- The I / O system of C++ contains a set of classes that define the file handling methods.
- These include ifstream, ofstream and fstream.
- These classes are derived from fstream base and from the corresponding iostream.h.
- These classes, designed to manage the disk files, are declared in fstream.h and therefore we must include this file in any program that uses files.



➤ **Classes for file stream operation:**

Class	Meanings
filebuf	It sets the file buffer to read and write
fstreambase	It supports operations common to the file streams. It serves as a base class for the derived classes ifstream, ofstream and fstream and contains open() and close() as member functions
ifstream	It supports input operations. It contains open() with default input mode and inherits get(), getline(), read(), seekg() and tellg() functions from istream.
ofstream	It supports output operations. It contains open() with default output mode and inherits put(), seekp(), tellp() and write() functions from ostream
fstream	It supports simultaneous input and output operations. It contains open() with default input mode and inherits all the functions from istream and ostream classes through iostream

➤ **Types of data Files:**

- Generally there are two types of files in C++:

✓ **Text Files:**

- A text file is a file that stores the information in ASCII characters.
- Each line of text is terminated by a special character, known as End of Line (EOL) or delimiter.

✓ **Binary Files:**

- A binary file 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.

➤ **Opening and Closing of Files:**

- A file must first be opened before data can be read from it or written to it.
- In C++ there are two ways to open a file with the file stream object.
 - Opening file using constructor.
 - Opening file using open () member function.
- The first method is preferred when a single file is used with a stream. However for managing multiple files with the same stream, the second method is preferred.

➤ **Opening files using Constructors:**

- In order to access a file, it has to be opened either in read, write or append mode.
- In all the three file stream classes, a file can be opened by passing a filename as the first parameter in the constructor itself.
- The syntax for opening a file using constructor is

```
streamclass_name file_objectname ("filename")
```

- The syntax of opening a file for output purpose only using an object ofstream class and the constructor is as follows:

```
ofstream ofstream_object("file name");
```

- **Example:** ofstream fout ("results.dat");
- The syntax of opening a file for input purpose only using an object ifstream class and the constructor is as follows:

```
ifstream ifstream_object("file name");
```

- **Example:** ifstream fin ("results.dat");

➤ **Opening files using open():**

- open() can be used to open multiple files that use the same stream object.
- The syntax for opening a file using open () member function is as follows:

```
file_stream_class stream_object;  
stream_object.open ("file_name");
```

- The syntax of opening a file for output purpose only using an object ofstream class and open() member function is as follows:

ofstream_object.open("file name");

- **Example:** ofstream outfile;
outfile.open ("data");
outfile.open ("text.dat");
- The syntax of opening a file for input purpose only using an object ifstream class and open() member function is as follows:

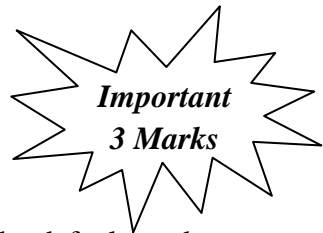
ifstream_object.open("file name");

- Example: ifstream ifile;
ifile.open ("data");
- To open a file for both input and output, we declare objects of fstream class. We know that the class fstream is derived from both ifstream and ofstream,
- The syntax for opening a file an object of type fstream class and the constructor is as follows:

fstream fstream-object("file name', mode);

- The syntax for opening a file an object of type fstream class and the open() member function is as follows:

fstream-object.open("file name', mode);



➤ File Modes:

- While using constructors or open(), the files were created or opened in the default mode.
- There was only one argument passed, i.e. the filename.
- C++ provides a mechanism of opening a file in different modes in which case the second parameter must be explicitly passed.
- **Syntax:** **stream_object.open("filename", mode);**
- Example: fout.open("data", ios::app) // This opens the file data in the append mode.
- The lists of file modes are:

Mode method	Meaning	Stream Type
ios::app	append to end of the file at opening time	ofstream
ios::in	open file for reading	ifstream
ios::out	open file for writing	ofstream
ios::ate	Open file for updating and move the file pointer to the end of file	ifstream

ios::trunc	On opening, delete the contents of file	ofstream
ios::nocreate	Turn down opening if the file does not exists	ofstream
ios::noreplace	Turn down opening if the file already exists	ofstream
ios::binary	Opening a binary file.	ifstream

Example:

```

ofstream fout ("text", ios::out);      // open text in output mode
ifstream fin ("text", ios::in);       // open text in input mode
fout.open ("data", ios::app)          // This opens the file data in the append mode
fout.open ("data", ios::app | ios::nocreate)
// This opens the file in the append mode but fails to open if it does not exist

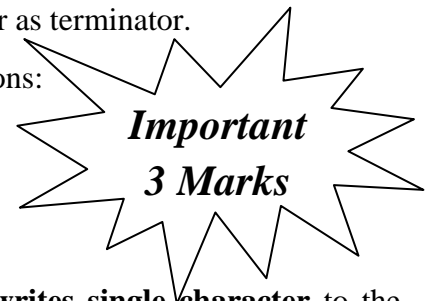
```

➤ Closing File:

- The member function close() on its execution removes the linkage between the file and the stream object.
- Syntax: **stream_object.close();**
- Example: ofstream.close();
ifstream.close();

➤ Input and output operation in text file:

- The data in text files are organized into lines with new line character as terminator.
- Text file need following types of character input and output operations:
 - put() function
 - get() function

**✓ put ():**

- The put() member function belongs to the class **ofstream** and **writes single character** to the associated stream.
- Syntax: **ofstream_object.put(ch);** // where ch is the character variable.
- Example: char ch='A';
ofstream fout ("text.txt");
fout.put (ch);
- fout is the object of ofstream. Text is the name of the file. Value at ch is written to text.

✓ get():

- The get() member function belong to the class **ifstream** and **reads a single character** from the associated stream.

- Syntax: **ifstream_object.get (ch);** // where ch is the character variable.
- Example:

```
char ch;
ifstream fin("text.txt");
fin.get (ch);
```
- fin is the object of ifstream. Text is the name of the file. Reads a character into the variable ch.

✓ **getline():**

- It is used to read a whole line of text. It belongs to the class ifstream.
- Syntax: **fin.getline(buffer, SIZE)**
- It reads SIZE characters from the file represented by the object fin or till the new line character is encountered, whichever comes first into the buffer.
- **Example:**

```
char book[SIZE];
ifstream fin;
fin.getline (book, SIZE);
```

➤ **Input and output operation in binary files:**

- Binary files are very much use when we have to deal with database consisting of records.
- The binary format is more accurate for storing the numbers as they are stored in the exact internal representation.
- There is no conversion while saving the data and hence it is faster.
- Functions used to handle data in binary form are:
 - write () member function.
 - read () member function



✓ **write ():**

- The write () member function belongs to the class ofstream and which is used to write binary data to a file.
- Syntax: **ofstream_object.write((char *) & variable, sizeof(variable));**
- These functions take 2 arguments. The first is the address of the variable and second the size of the variable in bytes. The address of the variable must be type casted to pointer to character.
- Example:

```
student s;
ofstream fout("std.dat", ios::binary);
fout.write((char *) &s, sizeof(s));
```

✓ read ():

- The read () member function belongs to the class ifstream and which is used to read binary data from a file.
- Syntax: **ifstream_object.read((char *) & variable, sizeof(variable));**
- These functions take 2 arguments. The first is the address of the variable and second the size of the variable in bytes. The address of the variable must be type casted to pointer to character.
- Example:

```
student s;
ifstream fin("std.dat", ios::binary)
fin.write((char *) &s, sizeof(s));
```

➤ Detecting End of file:

- Detecting end of file is necessary for preventing any further attempt to read data from the file.
- eof() is a member function of ios class.
- It returns a non-zero (true) value if the end of file condition is encountered while reading; otherwise returns a zero (false).
- Example:

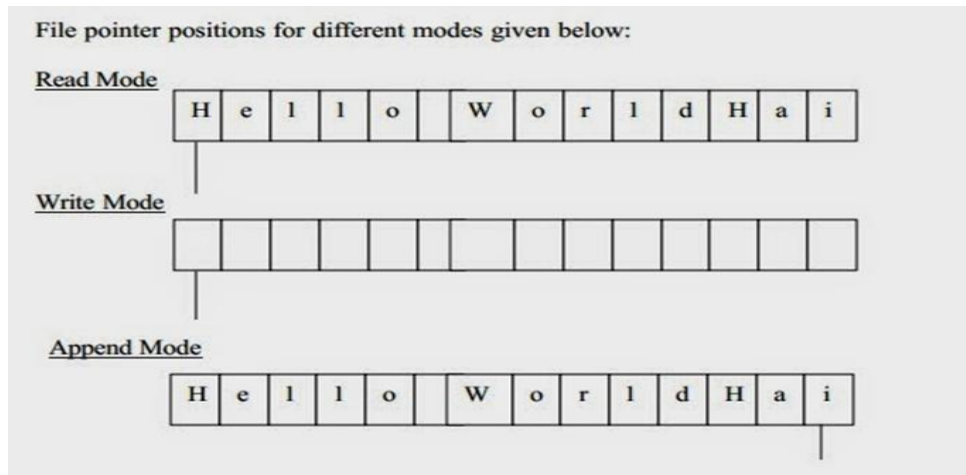
```
ifstream fin;
if(fin.eof( ))
{
    statements;
}
```

- This is used to execute set statements on reaching the end of the file by the object fin.

➤ File pointers and their manipulation:

- In C++, the file I/O operations are associated with the two file pointers:
 - Input pointer (get pointer)
 - Output pointer (put pointer)
- We use these pointers to move through files while reading or writing.
- Each time an input or output operation takes place, appropriate pointer is automatically advanced.
 - ifstream, like istream, has a pointer known as get pointer that points to the element to be read in the next input operation.
 - ofstream, like ostream, has a pointer known as put pointer that points to the location where the next element has to be written.
- There are three modes under which we can open a file:
 - Read only mode

- Write only mode
- Append mode



- When we open a file in read only mode, the input pointer is automatically set at the beginning so that we read the file from the beginning.
- When we open a file in write only mode, the existing contents are deleted and output pointer is set at the beginning
- If we want to open an existing file to add more data, the file is opened in append mode. This moves the file pointer to the end of the file.

➤ Functions for manipulation of file pointers:

- To move file pointers to any desired position inside a file, file stream classes support the following functions.
 - `seekg()` - Moves get file pointer to a specific location
 - `seekp()` - Moves put file pointer to a specific location
 - `tellg()` - Returns the current position of the get pointer
 - `tellp()` - Returns the current position of the put pointer
- The `seekp()` and `tellp()` are member functions of **ofstream**
- The `seekg()` and `tellg()` are member functions of **ifstream**.
- All four functions are available in the class **fstream**.

**Important
3 Marks**

✓ `seekg()`:

- Move the get pointer to a specified location from the beginning of a file.
- There are two types:
 - `seekg(long);`
 - `seekg(offset, seekdir);`
- The `seekg(long)` moves the get pointer to a specified location from the beginning of a file.

- Example: `inf.seekg(20);`
- The `seekg(offset, seekdir)` has two arguments: `offset` and `seekdir`.
- The `offset` indicates the number of bytes the get pointer is to be moved from `seekdir` position.
- `seekdir` takes one of the following three seek direction constants.

Constant	Meaning
<code>ios::beg</code>	seek from beginning of file
<code>ios::cur</code>	seek from current location
<code>ios::end</code>	seek from end of file

- Syntax: **`stream_objectname.seekg(offset, origin_value);`**
- Example : Some of the pointer offset calls and their actions are shown in the following table

seekg() function option	Action performed
<code>object.seekg(0, ios::beg)</code>	Take get pointer to the beginning of the file
<code>object.seekg(0, ios::end)</code>	Go to end of the file
<code>object.seekg(0, ios::cur)</code>	Stay get pointer at the current position.
<code>object.seekg(m, ios::beg)</code>	Move forward by (m+1) bytes in the file
<code>object.seekg(-m, ios::end)</code>	Go backward by m bytes from the file end.

✓ **seekp ():**

- Move the put pointer to a specified location from the beginning of a file.
- There are two types:
 - `seekp(long);`
 - `seekp(offset, seekdir);`
- The `seekp(long)` moves the put pointer to a specified location from the beginning of a file.
- Example: `inf.seekp(20);`
- The `seekp(offset, seekdir)` has two arguments: `offset` and `seekdir`.
- The `offset` indicates the number of bytes the put pointer is to be moved from `seekdir` position.
- Syntax: **`stream_objectname.seekp(offset, origin_value);`**

seekp() function option	Action performed
<code>object.seekp(0, ios::beg)</code>	Go to beginning of the file for writing
<code>object.seekp(0, ios::end)</code>	Go to end of the file for writing
<code>object.seekp(0, ios::cur)</code>	Stay at the current position for writing

object.seekp(m, ios::beg)	Move forward by m bytes from the beginning for writing
object.seekp(-m, ios::end)	Go backward by m bytes from the end for writing

✓ **tellg () and tellp ():**

- tellg () returns the current position of the get pointer.

- Syntax: position = ifstream_object.tellg ();

- Example: int position
position= fin.tellg();

- tellp () returns the current position of the put pointer.

- Syntax: position = ifstream_object.tellp ();

- Example: int position
position= fin.tellp();

➤ **Basic operation on binary file in C++:**

- Basic operation on binary file is:
 - Searching
 - Appending data
 - Inserting data in sorted files
 - Deleting a record
 - Modifying data

CHAPTER 12 – DATA FILE HANDLING BLUE PRINT

VSA (1 marks)	SA (2 marks)	LA (3 Marks)	Essay (5 Marks)	Total
-	01 Question	01 Question	-	02 Question
-	Question no 15	Question no 23	-	05 Marks

Important Questions

2 Marks Question:

1. Differentiate between ifstream and ofstream. [March 2015]
2. What is a stream? Mention any one stream used in C++. [June 2015]
3. Write any two member functions belonging to of stream class. [March 2016]
4. Write any two member functions belonging to if stream class. [June 2016]
5. Differentiate between read () and write (). [March 2017]
6. Differentiate between put () and get () functions with reference to binary files. [June 2017]

3 Marks Question:

1. Give the function of put(), get() and getline() with respect to text files. [March 2015]
2. List the different modes of opening a file with their meaning in C++. [June 2015]
3. Give the functions for the following: [March 2016]
 - a. get()
 - b. getline()
 - c. read()
4. Mention the types of data files. Explain. [June 2016]
5. Explain any three modes of to open a file in C++. [March 2017]
6. Write the member function belong to ifstream. [June 2017]

MDRPUC

Chapter-13

DATABASE CONCEPTS

➤ One Mark Questions:

1. What is data?
 - Data is a collection of facts, numbers, letters or symbols that the computer process into meaningful information.
2. What is Information?
 - Information is processed data, stored, or transmitted by a computer.
3. What is Database?
 - A Database is a collection of logically related data organized in a way that data can be easily accessed, managed and updated.
4. What is a field?
 - Each column is identified by a distinct header called attribute or field.
5. What is a record?
 - A single entry in a table is called a record or row. A record in a table represents set of related data.
 - Records are also called the tuple.
6. What is an entity?
 - An **Entity** can be any object, place, person or class.
 - In E-R Diagram, an **entity** is represented using rectangles.
7. What is an instance?
 - The collection of information stored in the database at a particular moment is called an **instance of the database**.
8. What is an attribute?
 - It is defined as a named column of a relation.
 - Ex: In STUDENT table, Regno, Name, Age, Class, Combination and Marks.
9. What is domain?
 - It is defined as a set of allowed values for one or more attributes.
10. What is a relation?
 - A relation is defined as a table with columns and rows. Data can be stored in the form of a two-dimensional table.

11. What is a table?

- A table is a collection of data elements organized in terms of rows and columns. Table is the simplest form of data storage.

12. What is normalization?

- Normalization is a step by step process of removing the different kinds of redundancy and anomaly one step at a time from the database.

13. What is a key?

- It is a column or columns which identifies the each row or tuple.

14. Give the symbol notation for project and select?

- SELECT – sigma (σ)
- PROJECT – Pi (Π)

15. What is data mining?

- Data mining is concerned with the analysis and picking out relevant information.

➤ Two/Three Mark Questions:

1. **Mention the applications of database.**

- **Banking:** For customer information, accounts and loans, and banking transactions.
- **Colleges:** For student information, course registrations and grades.
- **Credit card transactions:** For purchases on credit cards and generation of monthly statements.
- **Finance:** For storing information about holdings, sales and purchases of financial instruments such as stocks and bonds.
- **Sales:** For customer, product, and purchase information.
- **Telecommunication:** For keeping records of call made, generating monthly bills, maintaining balance on prepaid calling cards, and storing information about the communication networks.
- **Aadhaar database:** This is the biggest database in the world storing a data about 60 million people residing in India.

2. **Explain database users.**

- To design, use and maintain the database, many peoples are involved. The people who work with the database include:
 - End Users, System Analysts, Application programmers, Database Administrators (DBA)
- **End Users (Database Users)**
 - Database users are those who interact with the database in order to query and update the database, and generate reports.

- **System Analysts**
 - System analysts determine the requirement of end users; (especially naïve users), to create a solution for their business need and focus on non-technical and technical aspects.
- **Application programmers**
 - These are the computer professionals who implement the specifications given by the system analysts and develop the application programs.
- **Database Administrators (DBA)**
 - DBA is a person who has central control over both data and application.
 - Some of the responsibilities of DBA are authorization access, schema definition and modification, new software installation and security enforcement and administration.

3. **What is Data Independence? Mention the two types.**

- The capacity to change data at one layer does not affect the data at another layer is called **data independence**.
- Two types of data independence are:
 - Physical Data Independence
 - Logical Data Independence

4. **Explain physical data independence.**

- It is the capacity to change the internal level without having to change either the schemas at the conceptual or external level.
- Changes to the internal schema may be needed because some physical files had to be reorganized.
- Physical data independence refers to the data insulation of an application from the physical storage structure only, it is easier to achieve than logical data independence.
- The physical data independence are:
 - File Organization
 - Database Architecture
 - Database Models

5. **What is the difference between serial and direct access file organization.**

✓ **Serial File Organization:**

- Organization is continuous and simple.
- Data processing, which requires the use of all records, is best suited to use this method.

✓ **Direct Access File Organization**

- The type of storage device used is comparatively expensive.
- It is less efficient in the usage of storage space compared to the sequential organization.

6. **Explain ISAM with example.**

- The index sequential file organization is a combination of Sequential file organization and an Index file.
- Also referred as ISAM (indexed sequential access method).
- Data is stored physically in adjacent storage locations and there exists a logical relationship among the data stored by using ordering field.
- An additional file called as **Index file** would be created, which contains n number of records.
- Each record of index file has two fields:
 - The field is of the same data type as the ordering key field and
 - The second field is a pointer to a disk block (a block address).

7. **Give the advantages and disadvantages of ISAM.**

- **Advantages**
 - Search time is less.
 - There are fewer index entries than there are records in the data file.
 - Quick access to the records even when the volume of records is high.
- **Disadvantages**
 - Additional file (index file) has to be created.
 - Wastage of storage space by creating and maintaining the index file.
 - Always indirect retrieval of data because first search begins in the index files then moves to the data file (No direct retrieval).

8. **Classify the various types of keys used in database.**

The different types of keys are:

✓ **Primary key:**

- It is a field in a table which uniquely identifies each row/record in a database table.
- Primary keys must contain unique values.
- A primary key column cannot have NULL values.
- Ex: In Relation **STUDENT**, Regno serves as a primary key.

✓ **Candidate Key:**

- When more than one or group of attributes serve as a unique identifier, they are each called as candidate key.

✓ **Alternate Key**

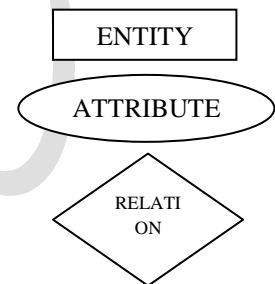
- The alternate key of any table are those candidate keys which are not currently selected as the primary key.
- This is also known as **secondary key**.

✓ **Foreign key**

- A key used to link two tables together is called a foreign key.
- This is sometimes called a referencing key.
- Foreign key is a field that matches the primary key column of another table.

9. **Explain different notations of E-R diagram.**

- **Entity:** An **entity** is represented using rectangles.
- **Attribute:** Attributes are represented by means of eclipses.
- **Relationship:** Relationship is represented using diamonds shaped box.



10. **Explain any three components of E-R model.**

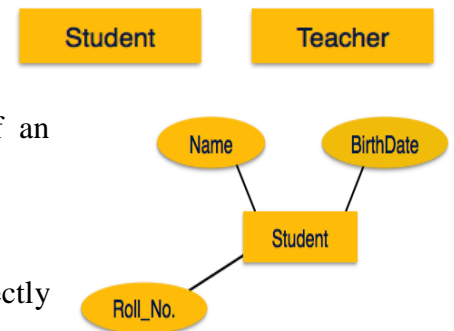
- ER-Diagram is a visual representation of data that describes how data is related to each other.

• **Entity:**

- An **Entity** can be any object, place, person or class.
- In E-R Diagram, an **entity** is represented using rectangles.
- Rectangles are named with the entity set they represent.

• **Attribute:**

- An **Attribute** describes a property or characteristic of an entity.
- Attributes are represented by means of eclipses.
- Every eclipse represents one attribute and is directly connected to its entity (rectangle).
- For example, Roll_No, Name and Birth date can be attributes of a student



• **Relationship:**

- A relationship type is a meaningful association between entity types.
- Relationship is represented using diamond shaped box.
- Relationship types are represented on the E-R diagram by a series of lines.

11. **What is a Relationship? Classify and give example.**

- A Relationship describes relations between entities.
- Relationship is represented using diamonds shaped box.

- There are three types of relationship that exist between entities.
 - Binary Relationship
 - Recursive Relationship
 - Ternary Relationship
- **Binary Relationship:** It means relation between two entities. This is further divided into three types.

1. **One to One:**

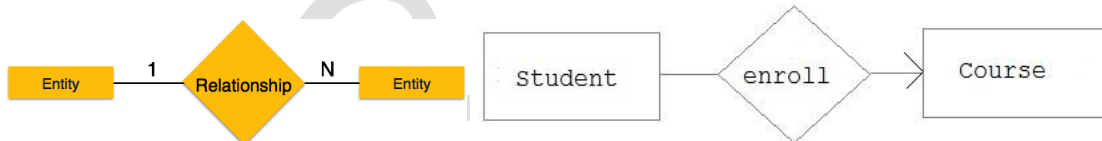
- This type of relationship is rarely seen in real world.



- The above example describes that one student can enroll only for one course and a course will also have only one Student. This is not what you will usually see in relationship.

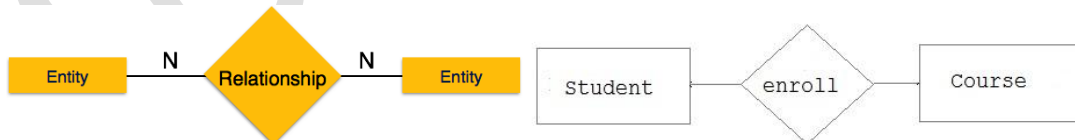
2. **One to Many:**

- It reflects business rule that one entity is associated with many number of same entity.
- For example, Student enrolls for only one Course but a Course can have many Students.



- The arrows in the diagram describes that one student can enroll for only one course.

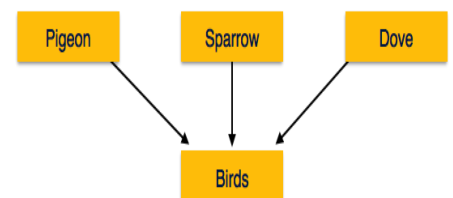
3. **Many to Many:**



- The above diagram represents that many students can enroll for more than one course.

12. **What is generalization?**

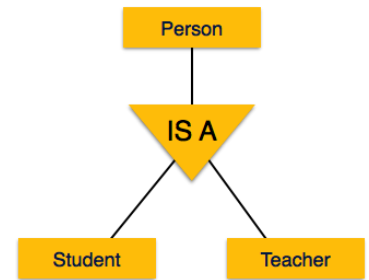
- In generalization, a number of entities are brought together into one generalized entity based on their similar characteristics.
- For example, pigeon, house sparrow, crow and dove can all be generalized as Birds.



13. **What is specification?**

- Specialization is the opposite of generalization.

- In specialization, a group of entities is divided into sub-groups based on their characteristics.
- Take a group 'Person' for example. A person has name, date of birth, gender, etc.
- Similarly, in a school database, persons can be specialized as teacher, student, or a staff, based on what role they play in school as entities.



14. What is Relation algebra?

- Relational algebra is a procedural query language that consists of a set of operations that take one or more relations as input and result into a new relation as an output.
- The relational algebraic operations can be divided into
 1. Basic set-oriented operations: Union, Set different, Cartesian product
 2. Relational-oriented operations: Selection, Projection, Division, Joins

15. What is Cartesian product?

- It is a binary operation, and it is denoted by the symbol \times .
- The Cartesian product of two relations R and S, denoted by $R \times S$, defines a new relation, which is the concatenation of each tuple of relation R with each tuple of relation S.

16. Give an example for relation Selection.

- Notation:- $\sigma_{(\text{Attribute} = \text{Value})}(\text{Relation})$
- Example: Consider a STUDENT relation

<u>Reg No</u>	<u>Name</u>	<u>Age</u>	<u>Combination</u>	<u>Marks</u>
101	Ramesh	17	PCMB	410
140	Mahesh	18	PCMC	505
110	Suresh	17	CEBA	475
121	Jagdeesh	20	PCMB	530

- $\sigma_{(\text{Age} = 17)}(\text{STUDENT})$

<u>Reg No</u>	<u>Name</u>	<u>Age</u>	<u>Combination</u>	<u>Marks</u>
101	Ramesh	17	PCMB	410
110	Suresh	17	CEBA	475

17. Give an example for relation Projection.

- Notation – $\prod_{A_1, A_2, A_n}(\text{Relation})$
- $\prod_{(\text{Reg_no, Combination})}(\text{Relation})$

<u>Reg No</u>	<u>Combination</u>
101	PCMB
140	PCMC
110	CEBA
121	PCMB

18. **What is Data warehouse?**

- A data warehouse is a repository of an organization's electronically stored data.
- Data warehouses are designed to facilitate reporting and supporting data analysis.
- The concept of data warehouses was introduced in late 1980's.

19. **List the components of data warehouse.**

- The components of data warehouse are:
 - Data Source
 - Data Transformation
 - Reporting
 - Metadata
- Additional components are Dependent data marts, Logical Data marts, Operational Data store.

➤ **Five Mark Questions:**

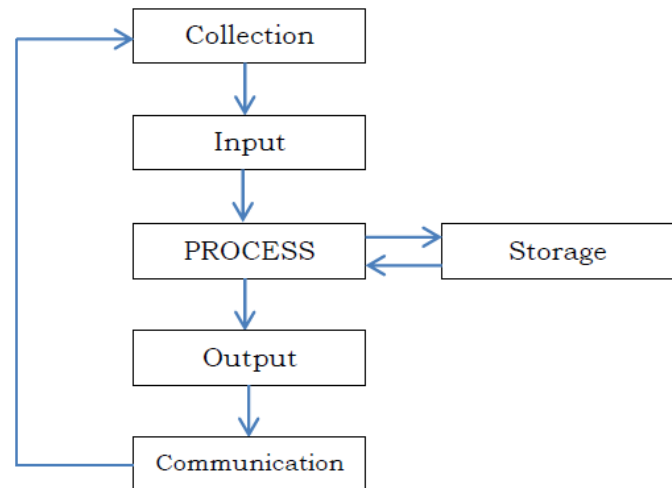
1. **Give the difference between Manual and Computerized data processing.**

	Manual Data Processing	Computerized Data Processing
1	The volume of data, which can be processed, is limited.	The volume of data, which can be processed is large
2	Requires large quantity of paper	Requires less quantity of paper
3	Speed and accuracy is executed is limited	Faster and Accurate
4	Labour cost is high	Labour cost is low
5	Storage medium is paper.	Storage medium is Hard disk etc.

2. **Explain Data processing cycle.**

- **Data Collection:** It is the process of systematic gathering of data from various sources that has been systematically observed, recorded and organized.
- **Data Input:** The raw data is put into the computer using a keyboard, mouse or other devices such as the scanner, microphone and the digital camera.
- **Data Processing:** Processing is the series of actions or operations on the input data to generate outputs.

- **Data storage:** Data and information should be stored in memory so that it can be accessed later.
- **Output:** The result obtained after processing the data must be presented to the user in user understandable form. The output can be generated in the form of report as hard copy or soft copy.
- **Communication:** Computers now-a-days have communication ability which increases their power. With wired or wireless communication connections, data may be input from a far place, processed in a remote area and stored in several different places and then transmitted by modem as an e-mail or posted to the website where the online services are rendered.



3. Explain the features or advantages of Database.

- **Redundancy can be minimized or controlled:** In DBMS environment if redundancy is present, then it can be controlled by propagating updates in all the places where ever redundant data is present.
- **Data Integrity:** Data Integrity refers to the correctness of the data in the database. In other words, the data available in the database is reliable data.
- **Data Sharing:** In DBMS, data is stored in the centralized database and all the permitted users can access the same piece of information required at the same time.
- **Database Security:** DBMS provides a variety of security mechanisms for the user to protect his or her data stored in the database.
- **Supports Concurrent access:** DBMS supports concurrent access to the same data stored in the database by applying locking and time stamp mechanisms.

4. Explain the concept of data abstraction.

- A major purpose of a database system is to provide users with an abstract view of the data. That is the system hides certain details of how the data are stored and maintained.
- There are three level of data abstraction.
 - Physical Level(Internal level)
 - Conceptual Level (Logical level)
 - View Level(External level)

- **Physical Level:**

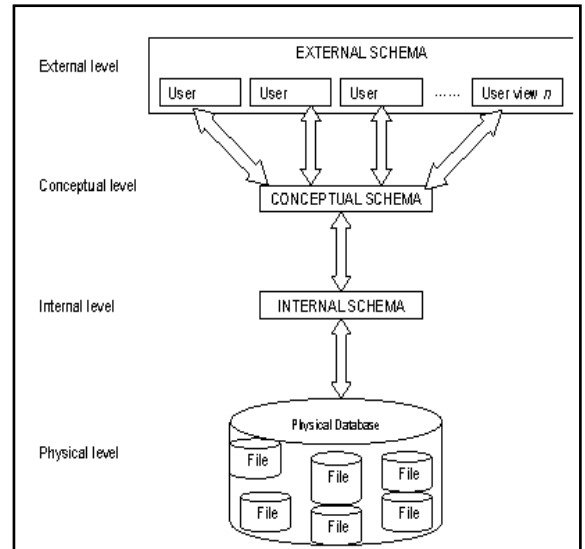
- It is the lowest level of abstraction that describes how the data are actually stored.
- The physical level describes complex low-level data structures in detail.
- It contains the definition of stored record and method of representing the data fields and access aid used.

- **Conceptual Level:**

- It is the next higher level of abstraction that describes what data are stored in the database and what relationships exist among those data.
- It also contains the method of deriving the objects in the conceptual view from the objects in the internal view.

- **View Level:**

- It is the highest level of abstraction that describes only part of the entire database.
- It also contains the method of deriving the objects in the external view from the objects in the conceptual view.



5. Explain DBMS Architecture.

- The design of Database Management System highly depends on its architecture.
- It can be centralized or decentralized or hierarchical.
- Database architecture is logically divided into three types.
 - Logical one-tier in 1-tier Architecture
 - Logical two-tier Client/Server Architecture.
 - Logical three-tier Client/Server Architecture.

➤ Logical one-tier in 1-tier Architecture:

- DBMS is the only entity where user directly sits on DBMS and uses it.
- Any changes done here will directly be on DBMS itself.
- It does not provide handy tools for end users and preferably database designers and programmers use single tier architecture.

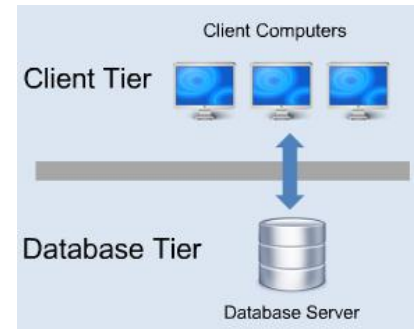


➤ Two-tier Client / Server Architecture:

- Two-tier Client / Server architecture is used for User Interface program and Application

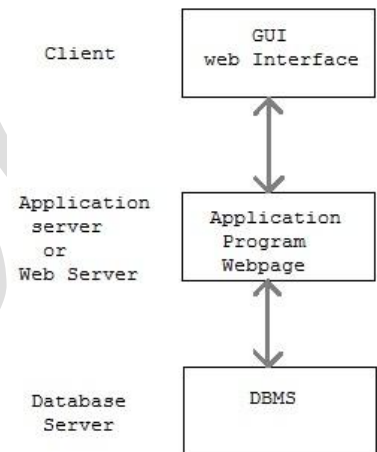
Programs that runs on client side.

- An interface called ODBC (Open Database Connectivity) provides an API that allows client side program to call the DBMS.
- Most DBMS vendors provide ODBC drivers. A client program may connect to several DBMS's. In this architecture some variation of client is also possible for example in some DBMS's more functionality is transferred to the client including data dictionary, optimization etc.



➤ **Three-tier Client / Server Architecture:**

- Three-tier Client / Server database architecture is commonly used architecture for web applications. Intermediate layer called **Application server** or Web Server stores the web connectivity software and the business logic (constraints) part of application used to access the right amount of data from the database server.
- This layer acts like medium for sending partially processed data between the database server and the client.



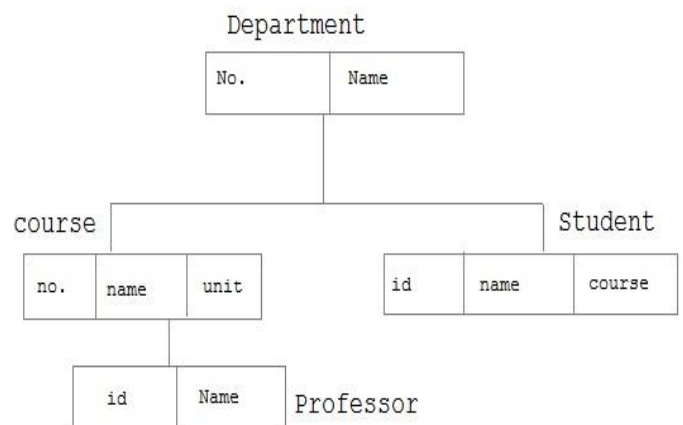
6. Explain Database Model.

- **Data model is a collection of conceptual tools for describing data, data relationship, data semantics and constraints.**
- A data model generally consists of
 - **Data model theory**, which is a formal description of how data may be structured and used.
 - **Data model instance**, which is a practical data model designed for a particular application.
- The process of applying model theory to create a data model instance is known as **data modeling**.
- In history of database design, three models have been in use.

- Hierarchical Model
- Network Model
- Relational Model

7. Explain Hierarchical data model.

- The Hierarchical data model organizes data in a tree structure.
- In this data model, data is represented by a



collection of **records** and the relationships are represented by **links**.

- In this model each entity has only one parent but can have several children. At the top of hierarchy there is only one entity which is called **Root node**.

✓ **Advantages:**

- **Simplicity:** The relationship between the various layers is logically simple.
- **Data Security:** The data security is provided by the DBMS.
- **Data Integrity:** There is always link between the parent segment and the child segment under it.
- **Efficiency:** It is very efficient because when the database contains a large number of one to many relationships and when the user requires large number of transaction.

✓ **Disadvantages:**

- Implementation complexity
- Database management problem
- Lack of structural Independence.
- Operational Anomalies

8. Explain Network data model.

- In 1971, the Conference on Data Systems Languages (CODASYL) formally defined the network models.

- In this model, data is represented by a collection of records and the relationships are represented by links.

- Each record is collection of fields,

which contains only one data value. A link is an association between two records.

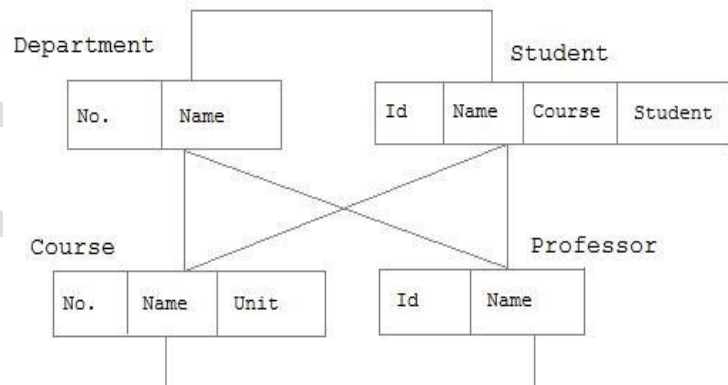
- In the network model, entities are organized in a graph, in which some entities can be accessed through several paths.

✓ **Advantages:**

- It is simple and easy to implement.
- It can handle many relationships within the organization.
- It has better data independence compared to hierarchical model.

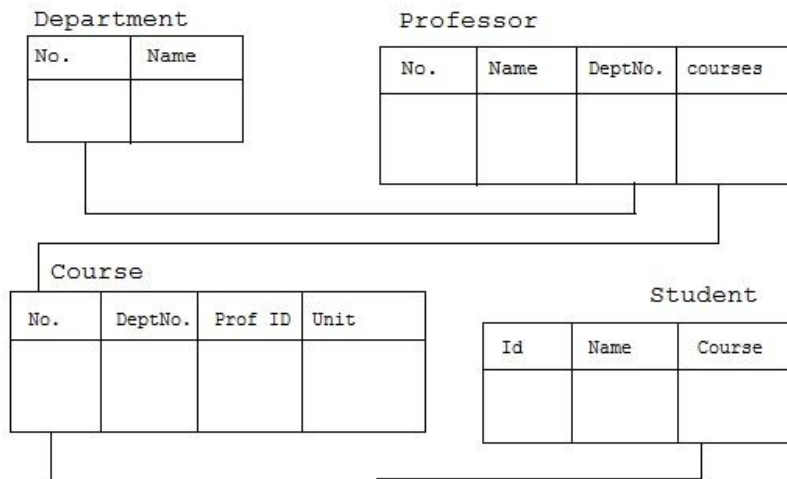
✓ **Disadvantages:**

- More complex system of database structure
- Lack of structural dependence.



9. Explain Relation Data Model.

- The relation data model was developed by E.F Codd in 1970.
- Unlike, hierarchical and network model, there are no physical links.
- All data is maintained in the form of tables consisting of rows and columns.
- Each row (record) represents an entity and a column (field) represents an attribute of the entity.
- In this model, data is organized in two-dimensional tables called **relations**. The tables or relation are related to each other.



10. Explain any 5 Codd's rule for database management.

- E.F Codd was a Computer Scientist who invented **Relational model** for Database management.
- Based on relational model, **Relation database** was created.
- ✓ **Rule zero**
 - This rule states that for a system to qualify as an **RDBMS**, it must be able to manage database entirely through the relational capabilities.
- ✓ **Rule 1: Information rule**
 - All information (including meta-data) is to be represented as stored data in cells of tables. The rows and columns have to be strictly unordered.
- ✓ **Rule 2: Guaranteed Access**
 - Each unique piece of data (atomic value) should be accessible by:
 - **Table Name + primary key (Row) + Attribute (column).**
 - **NOTE:** Ability to directly access via POINTER is a violation of this rule.
- ✓ **Rule 3: Systematic treatment of NULL**
 - **Null** has several meanings; it can mean missing data, not applicable or no value. It should be handled consistently. Primary key must not be null. Expression on **NULL** must give null.

✓ **Rule 4: Active Online Catalog**

- This rule states that the structure description of whole database must be stored in an online catalog i.e. data dictionary, which can be accessed by the authorized users.

✓ **Rule 5: Powerful language**

- One well defined language must be there to provide all manners of access to data.
- Example: **SQL**. If a file supporting table can be accessed by any manner except SQL interface, then its a violation to this rule.

✓ **Rule 6: View Updation rule**

- All view that is theoretically updatable should be updatable by the system.

11. Write comparing RA and SQL.

	RA	SQL
1	Relation Algebra	Structured Query Language
2	Is closed (the result of every expression is a relation)	Is a superset of relation algebra
3	Simple semantics	Complicated Semantics
4	It is used for reasoning, query, optimization etc	It is an end-user language.
5	Relationally Complete	Relationally Complete

CHAPTER 13 – DATABASE CONCEPTS BLUE PRINT

VSA (1 marks)	SA (2 marks)	LA (3 Marks)	Essay (5 Marks)	Total
01 Question	01 Question	01 Question	01 Question	04 Question
Question No 06	Question No 16	Question No 24	Question No 35	11 Marks

Important Questions**One Marks Questions:**

1. Define Primary key [March 2015]
2. What is a database? [June 2015, June 2016]
3. Define Data Mining. [March 2016]
4. Define an Entity. [March 2017, June 2017]

Two Marks questions:

1. What is data independence? Mention the types of data independence? [March 2015]
2. Write any two advantages of database system. [June 2015]
3. What are the advantages and disadvantages of ISAM. [March 2016]

4. Define Primary key and Secondary keys. [June 2016]
5. Write the difference between data and information. [March 2017]
6. Mention the database users. [March 2017, June 2017]

Three Marks Questions:

1. Briefly explain one-tier database architecture. [March 2015]
2. Write the different symbols used in E-R diagram with their significance. [June 2015]
3. Explain relational data model with example. [March 2016]
4. Define hierarchical data model. Give one advantage and disadvantage. [June 2016]
5. Mention any three advantages of random/direct access file organization. [June 2017]

Extra Questions:

6. Give the different notations for E-R diagram.
7. What is an entity-relationship diagram? Explain its components Entity and Attribute?
8. Define DBMS. Write any one feature of it.
9. What is a relationship? List different types of relationships.
10. Give the advantages and disadvantages of indexed sequential file organization.

Five Marks Questions:

1. What is data warehouse? Briefly explain its components. [March 2015]
2. Define the following database terms:
a. Data Model b. Tuple c. Domain d. Primary key e. Foreign key [June 2016]
3. Write the difference between manual and electronic data processing. [March 2016]
4. Explain any five applications of database. [June 2016]
5. Briefly explain the data processing cycle. [March 2017]
6. Write the difference between Hierarchical data model and network data model. [June 2017]

Extra Questions:

7. What is normalization? Explain second normal form with an example.
8. What is database model? Explain Hierarchical model.
9. Define any 5 Codd's rule.
10. Explain 3-level DBMS architecture.

Chapter-14

SQL COMMANDS

➤ What is SQL?

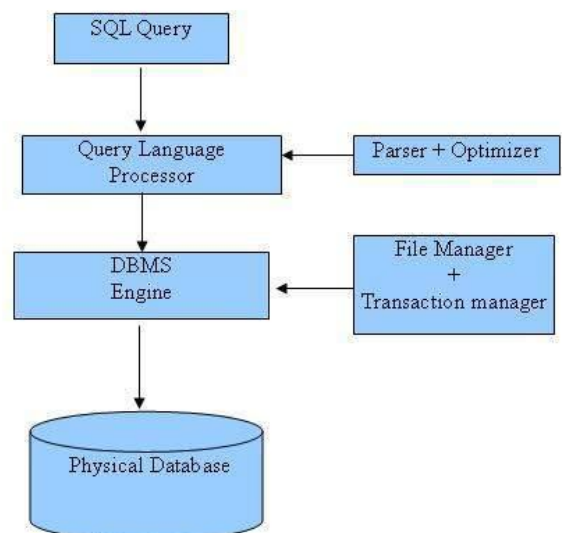
- **Structured Query Language** and it helps to make practice on SQL commands which provides immediate results.
- SQL is Structured Query Language, which is a computer language for storing, manipulating and retrieving data stored in relational database.
- SQL is the standard language for Relation Database System.
- All relational database management systems like MySQL, MS Access, and Oracle, Sybase, Informix, and SQL Server use SQL as standard database language.

➤ Why SQL?

- Allows users to create and drop databases and tables.
- Allows users to describe the data.
- Allows users to define the data in database and manipulate that data.
- Allows users to access data in relational database management systems.
- Allows embedding within other languages using SQL modules, libraries & pre-compilers.
- Allows users to set permissions on tables, procedures, and views

➤ SQL Architecture:

- When you are executing an SQL command for any RDBMS, the system determines the best way to carry out your request and SQL engine figures out how to interpret the task.
- There are various components included in the process.
- These components are:
 - Query Dispatcher
 - Optimization Engines
 - Classic Query Engine
 - SQL Query Engine, etc.
- Classic query engine handles all non-SQL queries but SQL query engine won't handle logical files.
- Simple diagram showing SQL Architecture:



➤ SQL Commands:

- The standard SQL commands to interact with relational databases are CREATE, SELECT, INSERT, UPDATE, DELETE and DROP.
- These commands can be classified into groups based on their nature:

➤ DDL - Data Definition Language:

- *DDL defines the conceptual schema providing a link between the logical and the physical structure of the database.*
- The functions of the Data Definition Language (DDL) are:
 1. DDL defines the physical characteristics of each record, filed in the record, field's data type, field's length, field's logical name and also specify relationship among those records.
 2. DDL describes the schema and subschema.
 3. DDL indicate the keys of records.
 4. DDL provides data security measures.
 5. DDL provides for the logical and physical data independence.
- Few of the basic commands for DDL are:

Command	Description
CREATE	Creates a new table, a view of a table, or other object in database
ALTER	Modifies an existing database object, such as a table.
DROP	Deletes an entire table, a view of a table or other object in the database.

➤ DML - Data Manipulation Language:

- *DML provides the data manipulation techniques like selection, insertion, deletion, updation, modification, replacement, retrieval, sorting and display of data or records.*
- DML facilitates use of relationship between the records.
- DML provides for independence of programming languages by supporting several high-level programming languages like COBOL, PL/1 and C++.
- Few of the basic commands for DML are:

Command	Description
SELECT	Retrieves certain records from one or more tables
INSERT	Creates a record
UPDATE	Modifies records
DELETE	Deletes records

➤ DCL - Data Control Language:

- These SQL commands are used for providing security to database objects.
- The different DCL commands are:

Command	Description
GRANT	Gives a privilege to user
REVOKE	Takes back privileges granted from user

➤ TCL – Transaction Control Language:

- It includes commands to control the transactions in a database system.
- The commonly used commands are:

Command	Description
COMMIT	Make all the changes made by the statements issued permanent.
ROLLBACK	Undoes all changes since the beginning of transaction or since a save point.

➤ Data Types in SQL:

- The following are the most common data types of SQL:

SL No	DATA TYPE	DESCRIPTION
1	NUMBER	A variable-length column. Allowed values are zero, positive and negative numbers
2	CHAR	A variable length field up to 255 character in length
3	VARCHAR/VARCHAR2	A variable length field up to 2000 character in length
4	DATE/TIME	A fixed length field. The time is stored as a part of the date. The default format is DD/MON/YY
5	LONG	A variable length field up to 2 GB in length
6	RAW	A variable length field used for binary data up to 2000 in length
7	LONG RAW	A variable length field used for binary data up to 2GB in length

1. NUMBER:

- Used to store a numeric value in a field column.
- It may be decimal, integer or real value.
- **General syntax:** **NUMBER(n, d)**

- Where **n** specifies the number of digits and **d** specifies the number of digits to right of the decimal point.

- **Example:** marks NUMBER(3), average NUMBER(2, 3)

2. CHAR:

- Used to store a character type data in a column.

- **General syntax:** CHAR(size)

- Where size represents the maximum (255 Characters) number of characters in a column.

- **Example:** name CHAR(15)

3. VARCHAR/VARCHAR2:

- It is used to store variable length alphanumeric data.

- **General syntax:** VARCHAR(size) / VARCHAR2(size)

- Where size represents the maximum (2000 Characters) number of characters in a column.

- **Example:** address VARCHAR2(50)

4. DATE:

- It is used to store date in columns.

- SQL supports the various date formats other than the standard DD-MON-YY.

- **Example:** dob DATE

5. TIME:

- It is used to store time in columns.

- SQL supports the various time formats other than the standard hh-mm-ss.

- Every DATE and TIME can be added, subtracted or compared as it can be done with other data types.

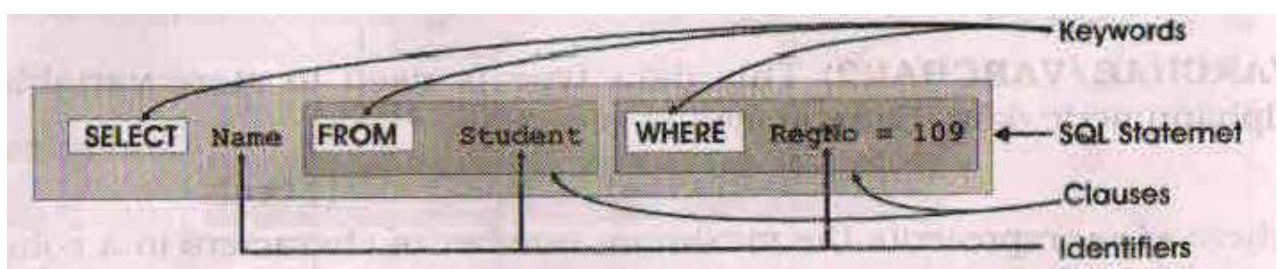
6. LONG:

1. It is used to store variable length strings of up to 2GB size.

2. **Example:** description LONG

➤ Structure of SQL command:

- Any SQL command is a combination of keywords, identifiers and clauses.
- Every SQL command begins with a keyword (CREATE, SELECT, DELETE and so on) which as a specific meaning to the language.



- SELECT, FROM and WHERE are keywords.
- The clauses are “FROM student” and “WHERE RegNo=109”.
- Here SELECT and FROM are mandatory, but WHERE is optional.
- Name, Student, RegNo, are identifier that refers to objects in the database.
- Name and RegNo are column names, while Student is a table name.
- The equal sign is an operator and 109 is a numeric constant.

➤ What is an Operator in SQL?

- An operator is a reserved word or a character used primarily in an SQL statement's WHERE clause to perform operation(s), such as comparisons and arithmetic operations.
- Operators are used to specify conditions in an SQL statement and to serve as conjunctions for multiple conditions in a statement.
 - Arithmetic operators (+, -, *, / %)
 - Comparison operators (>, <, >=, <=, =, !=, <>, !<, !>)
 - Logical operators (AND, OR, NOT, IN, BETWEEN, EXISTS, ALL, ANY, LIKE, UNIQUE)

➤ SQL Logical Operators:

- Here is a list of all the logical operators available in SQL.

Operator	Description
ALL	The ALL operator is used to compare a value to all values in another value set.
AND	The AND operator allows the existence of multiple conditions in an SQL statement's WHERE clause.
ANY	The ANY operator is used to compare a value to any applicable value in the list according to the condition.
BETWEEN	The BETWEEN operator is used to search for values that are within a set of values, given the minimum value and the maximum value.
EXISTS	The EXISTS operator is used to search for the presence of a row in a specified table that meets certain criteria.
IN	The IN operator is used to compare a value to a list of literal values that have been specified.
LIKE	The LIKE operator is used to compare a value to similar values using wildcard operators.
NOT	The NOT operator reverses the meaning of the logical operator with which it is used. Eg: NOT EXISTS, NOT BETWEEN, NOT IN, etc. This is a negate operator.
OR	The OR operator is used to combine multiple conditions in an SQL statement's WHERE clause.

IS NULL	The NULL operator is used to compare a value with a NULL value.
UNIQUE	The UNIQUE operator searches every row of a specified table for uniqueness (no duplicates).

➤ Implementation of SQL Commands

✓ CREATE TABLE

- The SQL **CREATE TABLE** statement is used to create a new table.
- Creating a basic table involves naming the table and defining its columns and each column's data type.
- **Syntax:** Basic syntax of **CREATE TABLE** statement is as follows:

```
CREATE TABLE Table_name
(
    column1 datatype,
    column2 datatype,
    column3 datatype,
    ....
    columnN datatype,
    PRIMARY KEY( one or more columns )
);
```

- Here **CREATE TABLE** is the keyword followed by the `Table_name`, followed by an open parenthesis, followed by the column names and data types for that column, and followed by a closed parenthesis.
- For each column, a name and a data type must be specified and the column name must be a unique within the table definition.
- Column definitions are separated by commas (,).
- Uppercase and lowercase letters makes no difference in column names.
- Each table must have at least one column.
- SQL commands should end with a semicolon (;).
- Example: Create a table “STUDENT” that contains five columns: RegNo, Name, Combination, DOB and Fees.

```
CREATE TABLE STUDENT
(
    RegNo          NUMBER (6),
    Name           VARCHAR2 (15),
    Combination    CHAR (4),
    DOB            DATE,
    Fees           NUMBER (9, 2),
    PRIMARY KEY ( RegNo )
);
```

- It creates an empty STUDENT table which looks like this:

RegNo	Name	Combination	DOB	Fees
-------	------	-------------	-----	------

- Viewing the table information:
 - The **DESCRIBE** or **DESC** command displays name of the columns, their data type and size along with the constraints.

```
SQL> DESCRIBE STUDENT;
```

Name	Null?	Type
REGNO	NOT NULL	NUMBER(6)
NAME		VARCHAR2(15)
COMBINATION		CHAR(4)
DOB		DATE
FEES		NUMBER(4,2)

✓ ALTER Statement:

- The table can be modified or changed by using the ALTER command.
- Syntax: Basic syntax of ALTER TABLE statement is as follows:

1. ALTER TABLE Table_name
ADD (column_name1 DataType, Column_name2 DataType.....);
2. ALTER TABLE Table_name
MODIFY (column_name1 DataType, Column_name2 DataType.....);
3. ALTER TABLE Table_name
DROP (column_name1 DataType, Column_name2 DataType.....);

- Example:

```
SQL> ALTER TABLE STUDENT ADD (Address VARCHAR2 (30));
Table altered.

SQL> ALTER TABLE STUDENT MODIFY (Address VARCHAR2 (40));
Table altered.

SQL> ALTER TABLE STUDENT DROP (ADDRESS);
Table altered.
```

- Using the ALTER TABLE command the following tasks cannot be performed
 - Changing a table name.
 - Changing the column name.
 - Decreasing the size of a column if table data exists.
 - Changing a column's data type.

✓ DROP TABLE:

- The SQL **DROP TABLE** statement is used to remove a table definition and all data, indexes, triggers, constraints, and permission specifications for that table.
- Syntax: Basic syntax of **DROP TABLE** statement is as follows:

```
DROP TABLE Table_name;
```

- Example:

```
SQL> DROP TABLE STUDENT;
Table dropped.
```

✓ INSERT:

- The SQL **INSERT INTO** Statement is used to add new rows of data to a table in the database.
- Syntax:
- There are two basic syntaxes of INSERT INTO statement as follows:

```
INSERT INTO TABLE_NAME [(column1, column2, column3,...columnN)]
VALUES (value1, value2, value3,...valueN);
```

- Here, column1, column2,...columnN are the names of the columns in the table into which you want to insert data.
- You may not need to specify the column(s) name in the SQL query if you are adding values for all the columns of the table. But make sure the order of the values is in the same order as the columns in the table.
- **METHOD 1:** The SQL INSERT INTO syntax would be as follows:

```
INSERT INTO TABLE_NAME VALUES (value1,value2,value3,...valueN);
```

- Example: Following statements would create six records in STUDENT table:

```
SQL> INSERT INTO STUDENT VALUES(1401,'RAMESH','PCMC','07-AUG-99',14000);
1 row created.
SQL> INSERT INTO STUDENT VALUES(1402,'JOHN','PCMB','15-SEP-99',13500);
1 row created.
SQL> INSERT INTO STUDENT VALUES(1403,'GANESH','PCME','19-AUG-99',16000);
1 row created.
SQL> INSERT INTO STUDENT VALUES(1404,'MAHESH','PCMC','14-JAN-98',17650);
1 row created.
SQL> INSERT INTO STUDENT VALUES(1405,'SURESH','PCMB','03-MAR-98',11500);
1 row created.
SQL> INSERT INTO STUDENT VALUES(1410,'ARUN','PCMC','01-APR-04',13000);
```

- **METHOD 2:** The SQL INSERT INTO syntax would be as follows:

```
SQL> INSERT INTO STUDENT (REGNO, NAME, FEES) VALUES (1411, 'SHREYA',24000);
1 row created.
```

```
SQL> INSERT INTO STUDENT (REGNO, COMBINATION,FEES) VALUES(1412,
'PCMB',21000);
1 row created.
```

- All the above statements would produce the following records in STUDENT table:

```
SQL> SELECT * FROM STUDENT;

  REGNO  NAME          COMB  DOB          FEES
-----  -
  1401  RAMESH          PCMC  07-AUG-99      14000
  1402  JOHN            PCMB  15-SEP-99      13500
  1403  GANESH          PCME  19-AUG-99      16000
  1404  MAHESH          PCMC  14-JAN-98      17650
  1405  SURESH          PCMB  03-MAR-98      11500
  1410  ARUN            PCMC  01-APR-04      13000
  1411  SHREYA          PCMB
  1412                                PCMB          21000

8 rows selected.
```

✓ UPDATE:

- SQL provides the ability to change data through UPDATE command.
- The UPDATE command used to modify or update an already existing row or rows of a table.
- The basic syntax of UPDATE command is given below.

```
UPDATE      Table_name
SET         column_name = value
           [, column_name =value .....]
[WHERE     condition];
```

Example:

```
SQL> UPDATE STUDENT SET COMBINATION='CEBA' WHERE REGNO=1411;
1 row updated.
SQL> UPDATE STUDENT SET NAME='AKASH' WHERE REGNO=1412;
1 row updated.
```

✓ DELETE command:

- In SQL, an already existing row or rows are removed from tables through the use of DELETE command.
- The basic syntax of DELETE command is given below.

```
DELETE      Table_name
[WHERE     condition];
```

Example:

```
SQL> DELETE STUDENT WHERE REGNO=1412;
1 row deleted.
```

✓ **SELECT:**

- SQL **SELECT** statement is used to fetch the data from a database table which returns data in the form of result table. These result tables are called result-sets.
- Syntax: The basic syntax of SELECT statement is as follows:

SELECT column1, column2, columnN FROM Table_name;	Compulsory Part
[WHERE condition(s)] [GROUPBY column-list] [HAVING condition(s)] [ORDER BY column-name(s)];	Optional Part

- Here, column1, column2...are the fields of a table whose values you want to fetch. If you want to fetch all the fields available in the field, then you can use the following syntax:

```
SELECT * FROM table_name;
```

- Example: Consider the STUDENT table having the following records:

```
SQL> SELECT * FROM STUDENT;
```

REGNO	NAME	COMB	DOB	FEES
1401	RAMESH	PCMC	07-AUG-99	14000
1402	JOHN	PCMB	15-SEP-99	13500
1403	GANESH	PCME	19-AUG-99	16000
1404	MAHESH	PCMC	14-JAN-98	17650
1405	SURESH	PCMB	03-MAR-98	11500
1410	ARUN	PCMC	01-APR-04	13000
1411	SHREYA	CEBA		24000

7 rows selected.

- Following is an example, which would fetch REGNO, NAME and COMBINATION fields of the customers available in STUDENT table:

```
SQL> SELECT REGNO, NAME, COMBINATION FROM STUDENT;
```

REGNO	NAME	COMB
1401	RAMESH	PCMC
1402	JOHN	PCMB
1403	GANESH	PCME
1404	MAHESH	PCMC
1405	SURESH	PCMB
1410	ARUN	PCMC
1411	SHREYA	CEBA

7 rows selected.

✓ **DISTINCT:**

- The SQL **DISTINCT** keyword is used in conjunction with SELECT statement to eliminate all the duplicate records and fetching only unique records.

- There may be a situation when you have multiple duplicate records in a table. While fetching such records, it makes more sense to fetch only unique records instead of fetching duplicate records.
- Syntax: The basic syntax of DISTINCT keyword to eliminate duplicate records is as follows:

```
SELECT DISTINCT      column1, column2,.....columnN
FROM                 Table_name
WHERE                [condition]
```

- Example: Consider the STUDENT table having the following records:

```
SQL> select * from student;

  REGNO NAME          COMB DOB          FEES
-----
  1401 RAMESH         PCMC 07-AUG-99     14000
  1402 JOHN           PCMB 15-SEP-99       13500
  1403 GANESH         PCMC 19-AUG-99       16000
  1404 MAHESH         PCMC 14-JAN-98       17650
  1405 SURESH         PCMB 03-MAR-98       11500
  1410 ARUN           PCMC 01-APR-04       13000
  1411 SHREYA        CEBA

7 rows selected.
```

- First, let us see how the following SELECT query returns duplicate combination records:

```
SQL> SELECT COMBINATION FROM STUDENT ORDER BY COMBINATION;

COMB
----
CEBA
PCMB
PCMB
PCMC
PCMC
PCMC
PCME

7 rows selected.
```

- Now, let us use DISTINCT keyword with the above SELECT query and see the result:

```
SQL> SELECT DISTINCT COMBINATION FROM STUDENT
ORDER BY COMBINATION;
```

- This would produce the following result where we do not have any duplicate entry:

```
SQL> SELECT DISTINCT COMBINATION FROM STUDENT ORDER BY COMBINATION;

COMB
----
CEBA
PCMB
PCMC
PCME
```

✓ WHERE clause – (Extracting specific rows)

- The SQL WHERE clause is used to specify a condition while fetching the data from single table or joining with multiple tables.
- If the given condition is satisfied then only it returns specific value from the table. You would use WHERE clause to filter the records and fetching only necessary records.

- The WHERE clause is not only used in SELECT statement, but it is also used in UPDATE, DELETE statement, etc., which we would examine in subsequent chapters.
- Syntax: The basic syntax of SELECT statement with WHERE clause is as follows:

```
SELECT      column1, column2, columnN
FROM        Table_name
WHERE       [condition]
```

- You can specify a condition using comparison or logical operators like >, <, =, LIKE, NOT, etc.
- Following is an example which would fetch REGNO, NAME and FEES fields from the STUDENT table where FEES is greater than 15000:

```
SQL> SELECT REGNO, NAME, FEES FROM STUDENT WHERE FEES>15000;
```

REGNO	NAME	FEES
1403	GANESH	16000
1404	MAHESH	17650
1411	SHREYA	24000

- Following is an example, which would fetch REGNO, NAME and COMBINATION fields from the STUDENT table for a COMBINATION is 'PCMC'.
- Here, it is important to note that all the strings should be given inside single quotes (") where as numeric values should be given without any quote as in above example:

```
SQL> SELECT REGNO, NAME, COMBINATION FROM STUDENT WHERE COMBINATION='PCMC';
```

REGNO	NAME	COMB
1401	RAMESH	PCMC
1404	MAHESH	PCMC
1410	ARUN	PCMC

- The SQL AND and OR operators are used to combine multiple conditions to narrow data in an SQL statement. These two operators are called *conjunctive operators*.
- These operators provide a means to make multiple comparisons with different operators in the same SQL statement.

✓ The AND Operator:

- The AND operator allows the existence of multiple conditions in an SQL statement's WHERE clause.
- Syntax: The basic syntax of AND operator with WHERE clause is as follows:

```
SELECT      column1, column2, columnN
FROM        Table_name
WHERE       [condition1] AND [condition2]...AND [conditionN];
```

- You can combine N number of conditions using AND operator. For an action to be taken by the SQL statement, whether it be a transaction or query, all conditions separated by the AND must be TRUE.
- Example: Consider the STUDENT table having the following records:

REGNO	NAME	COMB	DOB	FEES
1401	RAMESH	PCMC	07-AUG-99	14000
1402	JOHN	PCMB	15-SEP-99	13500
1403	GANESH	PCME	19-AUG-99	16000
1404	MAHESH	PCMC	14-JAN-98	17650
1405	SURESH	PCMB	03-MAR-98	11500
1410	ARUN	PCMC	01-APR-04	13000
1411	SHREYA	CEBA		24000

- Following is an example, which would fetch REGNO, NAME and DOB fields from the STUDENT table where fees is less than 15000 AND combination is 'PCMC':

```
SQL> SELECT REGNO, NAME, DOB FROM STUDENT WHERE FEES<15000 AND COMBINATION='PCMC';
```

REGNO	NAME	DOB
1401	RAMESH	07-AUG-99
1410	ARUN	01-APR-04

✓ The OR Operator:

- The OR operator is used to combine multiple conditions in an SQL statement's WHERE clause.
- Syntax: The basic syntax of OR operator with WHERE clause is as follows:

```
SELECT      column1, column2, columnN
FROM        Table_name
WHERE       [condition1] OR [condition2]...OR [conditionN];
```

- You can combine N number of conditions using OR operator. For an action to be taken by the SQL statement, whether it be a transaction or query, only any ONE of the conditions separated by the OR must be TRUE.
- Following is an example, which would fetch REGNO, NAME and DOB fields from the STUDENT table where fees is less than 15000 OR combination is 'PCMC':

```
SQL> SELECT REGNO, NAME, DOB FROM STUDENT WHERE FEES<15000 OR COMBINATION='PCMC';
```

REGNO	NAME	DOB
1401	RAMESH	07-AUG-99
1402	JOHN	15-SEP-99
1404	MAHESH	14-JAN-98
1405	SURESH	03-MAR-98
1410	ARUN	01-APR-04

✓ ORDER BY – (Sorting the data)

- The SQL **ORDER BY** clause is used to sort the data in ascending or descending order, based on one or more columns. Some database sorts query results in ascending order by default.
- Syntax: The basic syntax of ORDER BY clause is as follows:

```
SELECT      column-list
FROM        Table_name
[WHERE      condition]
[ORDER BY   column1, column2, .. columnN] [ASC | DESC];
```

- You can use more than one column in the ORDER BY clause. Make sure whatever column you are using to sort, that column should be in column-list.
- Example: Consider the STUDENT table having the following records:

REGNO	NAME	COMB	DOB	FEES
1401	RAMESH	PCMC	07-AUG-99	14000
1402	JOHN	PCMB	15-SEP-99	13500
1403	GANESH	PCME	19-AUG-99	16000
1404	MAHESH	PCMC	14-JAN-98	17650
1405	SURESH	PCMB	03-MAR-98	11500
1410	ARUN	PCMC	01-APR-04	13000
1411	SHREYA	CEBA		24000

- Following is an example, which would sort the result in ascending order by NAME:

```
SQL> SELECT * FROM STUDENT ORDER BY NAME;
```

REGNO	NAME	COMB	DOB	FEES
1410	ARUN	PCMC	01-APR-04	13000
1403	GANESH	PCME	19-AUG-99	16000
1402	JOHN	PCMB	15-SEP-99	13500
1404	MAHESH	PCMC	14-JAN-98	17650
1401	RAMESH	PCMC	07-AUG-99	14000
1411	SHREYA	CEBA		24000
1405	SURESH	PCMB	03-MAR-98	11500

7 rows selected.

- Following is an example, which would sort the result in descending order by NAME:

```
SQL> SELECT * FROM STUDENT ORDER BY NAME DESC;
```

REGNO	NAME	COMB	DOB	FEES
1405	SURESH	PCMB	03-MAR-98	11500
1411	SHREYA	CEBA		24000
1401	RAMESH	PCMC	07-AUG-99	14000
1404	MAHESH	PCMC	14-JAN-98	17650
1402	JOHN	PCMB	15-SEP-99	13500
1403	GANESH	PCME	19-AUG-99	16000
1410	ARUN	PCMC	01-APR-04	13000

7 rows selected.

✓ Working out simple calculations.

- Whenever we want to perform simple calculations such as $10 / 5$, we can perform using SELECT statement which causes an output on monitor.
- But SELECT requires table name to operate.
- One can make use of the dummy table provided by SQL called DUAL which is a single row and single column table.
- It is used when data from table is not required.
- For example, when a calculation is to be performed such as $10*3$, $10/2$ etc. and to display the current system date, we could use the following queries.

SQL> SELECT 10*5 FROM DUAL;	SQL> SELECT SYSDATE FROM DUAL;
10*5 ----- 50	SYSDATE ----- 22-DEC-16

✓ SQL Functions:

- The SQL functions serve the purpose of manipulating data items and returning a result.
- There are many built in functions included in SQL and can be classified as **Group Functions and Scalar Functions**.
- **Group Functions:**
 - Functions that act on set of values are called group functions.
 - A group functions can takes entire column of data as its arguments and produces a single data item that summarizes the column.
 - Following are the SQL group functions.

Function	Description
AVG	Returns average value of 'N', ignoring NULL values
COUNT(expr)	Returns the number of rows where 'expr' is not NULL
COUNT(*)	Returns the number of rows in the table including duplicates and those with NULL values
MIN	Returns minimum value of 'expr'
MAX	Returns maximum value of 'expr'
SUM	Returns sum of values 'N'

- **Scalar Functions:**

- Functions that act on only one value at a time are called scalar functions.
- We can further classify the functions using the type of data with they are designed to work.

Function	Description
Numeric Functions	Work with numbers. Examples: ABS, POWER, ROUND, SQRT
String Functions	Work with character based data. Examples: LOWER, INITCAP, UPPER, SUBSTR, LENGTH, LTRIM, RTRIM
Date Functions	Work with Date data types. Example: ADD_MONTHS, LAST_DAY, MONTHS_BETWEEN, NEXT_DAY
Conversion Functions	These functions are used to convert one type of data to another. Example: TO_NUMBER, TO_CHAR, TO_DATE

Consider the EXAMINATION table:

RegNo	Name	CC	Phy	Che	Mat	Cs	Total	City
101	Ajay	C1	98	100	97	99	394	Hassan
102	Banu	C2	38	50	37	49	174	Belur
103	Chandan	C2	100	100	97	99	396	Mysuru
104	John	C3	78	80	67	79	304	Alur
105	Kaleem	C1	88	80	91	79	338	Hassan
106	Raheem	C2	100	98	97	79	374	Hassan
107	Sanjay	C3	47	60	56	78	241	Alur
108	Tarun	C3	33	34	77	28	172	Arasikere
109	Uday	C2	100	98	97	79	374	Hassan
110	Venki	C3	47	60	56	78	241	Belur

✓ COUNT () Function:

- This function is used to count the number of values in a column.
- COUNT (*) is used to count the number of rows in the table including duplicates and those with NULL values.
- Example 1:
 - SELECT COUNT (*) FROM EXAMINATION;
 - The above query returns 10.
- Example 2:
 - SELECT COUNT (RegNo) FROM EXAMINATION WHERE CC = 'C3' ;
 - The above query returns 4.

✓ AVG () Function:

- This function is used to find the average of the values in a numeric column.
- Example 1:
 - SELECT AVG (Cs) FROM EXAMINATION;
 - The above query returns 74.7

✓ SUM () Function:

- This function is used to find the sum of the values in a numeric column.
- Example:
 - SELECT SUM (Phy) FROM EXAMINATION;
 - The above query returns 729

✓ MAX () Function:

- This function is used to find the maximum values in a column.
- Example:
 - SELECT MAX (Phy) FROM EXAMINATION;
 - The above query returns 100

✓ MIN () Function:

- This function is used to find the minimum values in a column.
- Example:
 - SELECT MIN (Phy) FROM EXAMINATION;
 - The above query returns 33

✓ GROUP BY (Grouping Result)

- The SQL **GROUP BY** clause is used in collaboration with the SELECT statement to arrange identical data into groups.
- The **GROUP BY** clause follows the **WHERE** clause in a SELECT statement and precedes the **ORDER BY** clause.
- **Syntax:** The basic syntax of **GROUP BY** clause is given below.

```
SELECT      column1, column2
FROM        Table_name
WHERE       [ conditions ]
GROUP BY   column1, column2
ORDER BY   column1, column2
```

- Example 1: To find the number of students in each college.

```
SELECT      CC, COUNT (CC)
FROM        EXAMINATION
GROUP BY    CC;
```

- Example 2: To find the number of students, sum, average, maximum, minimum marks in computer science from each city.

```
SELECT      City, COUNT (City), SUM (Cs), AVG (Cs), MAX (Cs), MIN (Cs)
FROM        EXAMINATION
GROUP BY    City;
```

➤ SQL CONSTRAINTS:

- Constraints are the rules enforced on data columns on table.
- These are limiting the type of data that can go into a table.
- This ensures the accuracy and reliability of the data into the database.
- SQL allows two types of constraints.
 - **Column level constraints:** These constraints are defined along with the column definition when creating or altering a table structure. These constraints apply only to individual columns.
 - **Table level constraints:** These constraints are defined after all the table columns when creating or altering a table structure. These constraints apply to groups of one or more columns.
- Following are the commonly used constraints available in SQL.

Constraints	Description
NOT NULL	Ensures that a column cannot have NULL value
UNIQUE	Ensures that all values in column are different
PRIMARY KEY	Uniquely identified eac row in a database table.
FOREIGN KEY	Uniquely identified each rown in any other database table
DEFAULT	Provides a default value for a column when none is specified
CHECK	Ensures that all values in a column satisfy certain condition.

✓ NOT NULL Constraint:

- By default column can hold NULL values.
- When a column is defined as NOT NULL then the column becomes a mandatory column.
- It implies that a value must be entered into the column if the row is to be inserted for storage in the table.

- Example: Consider the following CREATE TABLE command creates a new table called PRODUCT and add six columns, two which PID and Description specify not to accept NULLs.

```
CREATE TABLE PRODUCT
(
    PID                CHAR (4)        NOT NULL,
    Description        VARCHAR2 (25), NOT NULL
    CompanyId         CHAR (10),
    DOM                DATE,
    Type               CHAR (10),
    Price              NUMBER (10,2)
);
```

✓ UNIQUE Constraints:

- This constraint ensures that no rows have the same value in the specified column(s). A table must have many unique keys.
- Example: UNIQUE constraint applied on PID of PRODUCT table ensures that no rows have the same PID value, as shown below

```
CREATE TABLE PRODUCT
(
    PID                CHAR (4)        NOT NULL UNIQUE,
    Description        VARCHAR2 (25), NOT NULL
    CompanyId         CHAR (10),
    DOM                DATE,
    Type               CHAR (10),
    Price              NUMBER (10,2)
);
```

✓ PRIMARY KEY Constraints:

- A primary key is a field which uniquely identifies each row in a database table. A primary key in a table has special attributes:
- By default this column is NOT NULL. It defines the column as a mandatory column i.e. the column cannot be left blank.
- The data held in this column must be unique.
- Example:

```
CREATE TABLE PRODUCT
(
    PID                CHAR (4)        PRIMARY KEY,
    Description        VARCHAR2 (25), NOT NULL
    CompanyId         CHAR (10),
    DOM                DATE,
    Type               CHAR (10),
    Price              NUMBER (10,2)
);
```


✓ FOREIGN KEY Constraint:

- A FOREIGN KEY is used to link two tables together.
- A foreign key is a column whose values are derived from the PRIMARY KEY of some other table.
- Example:

```
CREATE TABLE PRODUCT
(
    PID          CHAR (4)          PRIMARY KEY,
    Description  VARCHAR2 (25), NOT NULL
    CompanyId   CHAR (10) REFERENCES COMPANY (CID)
    DOM         DATE,
    Type        CHAR (10),
    Price       NUMBER (10,2)
);
CREATE TABLE COMPANY
(
    CID          CHAR (10) PRIMARY KEY,
    CProfile    VARCHAR2 (200),
    Noofproducts NUMBER (20),
    DOE         DATE
);
```

✓ DEFAULT Constraints:

- A default value can be specified for a column using the DEFAULT clause.
- The DEFAULT constraint provides a default value to a column when the INSERT INTO command does not provide a specific value.
- Example:

```
CREATE TABLE PRODUCT
(
    PID          CHAR (4)          PRIMARY KEY,
    Description  VARCHAR2 (25), NOT NULL
    CompanyId   CHAR (10),
    DOM         DATE,
    Type        CHAR (10),
    Price       NUMBER (10, 2) DEFALUT 1000.00
);
```

✓ CHECK Constraints:

- The CHECK Constraint is used to establish a TRUE/FALSE condition that is applied to the data placed in a column.
- If a value does not meet the condition, it cannot be placed in the column.
- Example:

```
CREATE TABLE PRODUCT
```

```
(
    PID          CHAR (4)          CHECK (PID LIKE 'P%'),
    Description  VARCHAR2 (25),
    CompanyId   CHAR (10),
    DOM         DATE,
    Type        CHAR (10),
    Price       NUMBER (10, 2) CHECK (Price>0)
);
```

✓ TABLE Constraints:

- When a constraint is applied to a group of columns of the table, it is called a table constraint.
- Column constraint is defined along with the end of the column.
- Table constraints are defined at the end of the table.
- Example:

```
CREATE TABLE PRODUCT
(
    PID          CHAR (4)          NOT NULL,
    Description  VARCHAR2 (25) NOT NULL,
    CompanyId   CHAR (10),
    DOM         DATE,
    Type        CHAR (10),
    Price       NUMBER (10, 2),
    PRIMARY KEY (PID, Description)
);
```

➤ Joins

- The SQL **Joins** clause are used to fetch/retrieve data from two or more tables based on the join condition which is specified in the WHERE condition.
- Basically data tables are related to each other with keys. We can use these keys relationship in SQL joins.

✓ SQL Join Types:

- There are different types of joins available in SQL:
 - **INNER JOIN**: returns rows when there is a match in both tables.
 - **LEFT JOIN**: returns all rows from the left table, even if there are no matches in the right table.
 - **RIGHT JOIN**: returns all rows from the right table, even if there are no matches in the left table.
 - **FULL JOIN**: returns rows when there is a match in one of the tables.
 - **SELF JOIN**: is used to join a table to itself as if the table were two tables, temporarily renaming at least one table in the SQL statement.

- **CARTESIAN JOIN:** returns the Cartesian product of the sets of records from the two or more joined tables.

➤ **Creating VIEWS:**

- Database Views are created using the CREATE VIEW statement.
- Views can be created from a single table, multiple tables or another view.
- To create a view, a user must have the appropriate system privilege according to the specific implementation.
- Syntax: The basic CREATE VIEW syntax is as follows:

```
SQL> CREATE VIEW      view_name AS
      SELECT           column1, column2.....
      FROM             table_name
      WHERE            [ condition ];
```

➤ **Privileges and Roles:**

- Privileges: Privileges defines the access rights provided to a user on a database object.
- There are two types of privileges:
 - **System Privileges:** This allows the user to CREATE, ALTER, or DROP database objects.
 - **Object privileges:** This allows the user to EXECUTE, SELECT, INSERT, UPDATE or DELETE data from database objects to which privileges apply.

CHAPTER 14 – STRUCTURED QUERY LANGUAGE BLUE PRINT				
VSA (1 marks)	SA (2 marks)	LA (3 Marks)	Essay (5 Marks)	Total
-	01 Question	-	01 Question	02 Question
-	Question no 17	-	Question no 36	06 Marks

Important Questions

Two Marks Questions:

1. Give the syntax and example for DELETE command in SQL. [March 2015]
2. List the data types supported in SQL. [June 2015]
3. Write the syntax for DELETE and INSERT commands in SQL. [March 2016]
4. Mention the logical operators used in SQL. [June 2016]
5. Give the syntax and example of UPDATE command in SQL. [March 2017]

6. What is the difference between ORDER BY and GROUP BY clause used in SQL? Give example for each. [June 2017]
7. List the relational operators supported by SQL.
8. Why the DROP command used? Write its syntax.
9. What are the difference between DROP and DELETE command?
10. Give the syntax and example for CREATE VIEW command in SQL.
11. Classify the built-in functions in SQL.

Five Marks Question:

1. Explain various group functions in SQL. [March 2015, March 2017, June 2017]
2. What is data definition language? Explain SELECT and UPDATE command. [June 2015]
3. Describe any five logical operators available in SQL. [March 2016]
4. What is SQL? Explain the different SQL commands.
5. What is the purpose of CREATE command? Write the syntax and example of CREATE command.
6. Explain SELECT statement with syntax and with minimum 3 examples.
7. Explain 5 variations of SELECT command.
8. What are SQL constraints? Explain any two constraints.

Chapter-15

NETWORKING CONCEPTS

➤ Introduction:

- A computer network is a interconnection of two or more computers that are able to exchange information's.
- Two computers are said to be inter connected if they are capable of exchanging information.

➤ Network Goals/Advantages of Networking:

- **Resource Sharing:**
 - The aim is to make all programs, data and peripherals available to anyone on the network irrespective of the physical location of the resources and the user.
- **Reliability:**
 - A file can have copies on two or three different machines, so if one of them is unavailable (hardware crash), the other copies could be used.
 - For military, banking, air reservation and many other applications it is of great importance.
- **Cost Factor:**
 - Personal computers have better price/performance ratio than micro computers.
 - So it is better to have PC's, one per user, with data stored on one shared file server machine.
- **Communication Medium.**
 - Using a network, it is possible for managers, working far apart, to prepare financial report of the company.
 - The changes at one end can be immediately noticed at another and hence it speeds up co-operation among them.

➤ Need of Networking:

- File sharing provides sharing and grouping of data files over the network.
- Printing sharing of computer resources such as hard disk and printers etc.
- E-mail tools for communication with the e-mail address.
- Remote access able to access data and information around the globe.
- Sharing the database to multiple users at the same time by ensuring the integrity.

➤ Evolution of Networking:

- In 1969 U.S. Department of Defense sponsored a project named ARPANET (Advanced Research Projects Agency Network).
- The goal of the project was to connect various universities and US Defense.
- In mid 80's National Science Foundation created a new high capacity network called NSFnet, which was more powerful than ARPANET.
- In 1990 the Internet came into picture.

➤ Internet:

- The internet is worldwide network of computer network evolved from the first network ARPANET.
- Internet is an interconnection of large and small networks around the globe.
- The common use of internet standards allows users connected to one network to communicate with users on another network.

➤ Interspace:

- Interspace is a client/server software program that allows multiple users to communicate online with real-time audio, video and text chat in dynamic 3D environments.
- Interspace provides the most advanced form of communication available on the Internet today.

➤ Elementary Terminology of Networks:**1. Nodes (Workstations):**

- The term nodes refer to the computers that are attached to a network and are seeking to share the resources of the network.
- Of course, if there were no nodes (also called workstations), there would be no network at all.

2. Server:

- A Server is also a computer that facilitates the sharing of data, software, and hardware resources like printers, modems etc on the network.
- Servers can be of two types:
 - i. Non-dedicated servers
 - ii. Dedicated servers

✓ Non-dedicated Servers:

- On small networks, a workstation that can double up as a server is known as non-dedicated server since it is not completely dedicated to the cause of serving.

- Such servers can facilitate the resource-sharing among workstations on a proportionately smaller scale.
- Since one computer works as a workstation as well as server, it is slower and requires more memory.
- The (small) networks using such a server are known as PEER-TO-PEER networks.

✓ **Dedicated Servers:**

- On bigger network installations, there is a computer reserved for server's job and its only job is to help workstations access data, software and hardware resources.
- It does not double-up as a workstation and such a server is known as dedicated server.
- The networks using such a server are known as MASTER-SLAVE networks.
- On a network, there may be several servers that allow workstations to share specific resources. For example, file server, printer server and modem server.

3. Network Interface Unit (NIU) :

- A Network Interface Unit is an interpreter that helps establish communication between the server and workstations.
- The NIU is also called Terminal Access Point (TAP).

4. MAC address:

- The MAC address refers to the physical address assigned by NIC manufacturer.

➤ **OSI Reference Model:**

- The Open Systems Interconnection (OSI) model is a reference tool for understanding data communications between any two networked systems.
- It divides the communications process into seven layers.
- The three lowest layers focus on passing traffic through the network to end system.
- The top four layers come into play in the end system to complete the process.

1. Layer 1 – Physical Layer:

- The physical layer is concerned with transmitting raw bits over a communication channel.
- It also deals with mechanical, electrical and timing interfaces.

2. Layer 2 - Data Link Layer:

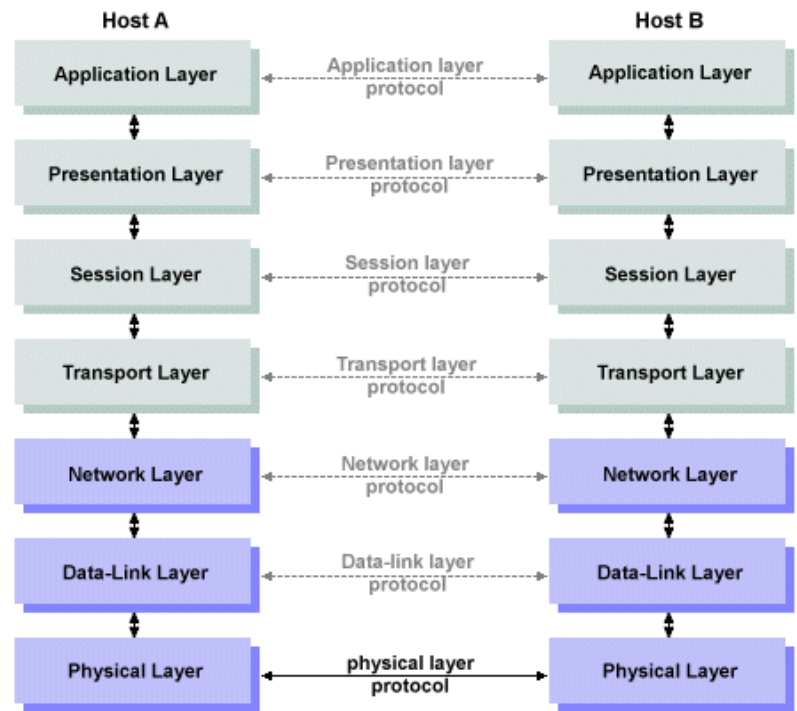
- The main function of the data link layer is to take a raw transmission facility and transform it into a line that appears free of transmission errors in the network layer.

3. Layer 3 – Network Layer:

- The network layer is concerned with controlling the operation of the subnet.
- The main function is determining how packets are routed from source to destination.

4. Layer 4 – Transport Layer:

- The basic function of the transport layer is to accept data from the above (session) layer, split it up into smaller units if needed, pass these to the network layer, and ensure that the pieces all arrive correctly at the other end.

**5. Layer 5 – Session Layer:**

- The session layer allows users on different machines to establish sessions between them.
- It includes dialog control, token management and synchronization.

6. Layer 6 – Presentation Layer:

- The presentation layer is concerned with the syntax and semantics of the information transmitted concerned with moving bits around the layer.

7. Layer 7 – Application Layer:

- The application layer contains a variety of protocols that are commonly needed by the user.
- For example HTTP which is the bases for the World Wide Web to access web pages.

➤ TCP/IP Model:

- The TCP/IP model uses four layers to perform the functions of the seven-layer OSI model.

1. Layer 1 – Network Access Layer:

- The lowest layer of the TCP/IP protocol hierarchy.
- It defines how to use the network to transmit an IP data.
- It encompasses the functions of physical and data link layer of OSI reference model.

2. Layer 2 – Internet Layer:

- Provides services that equivalent to OSI network layer.
- The primary concern of the protocol at this layer is to manage the connections across

network as information is passed from source to destination.

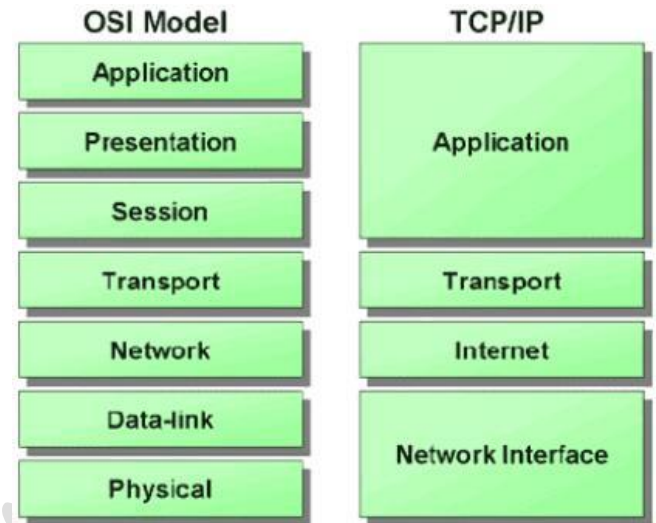
- The internet Protocol (IP) is the primary protocol at this layer.

3. Layer 3 – Transport Layer:

- It is designed to allow peer entities on the source and destination hosts to carry on a conversation.
- Two end-to-end transport protocols have defined here TCP and UDP.

4. Layer 4 – Application Layer:

- It includes the OSI session, presentation and application layers.
- An application is any process that occurs above the transport layer.
- This includes all of the processes that involve user interaction.
- The application determines the presentation of the data and controls the session.



TCP/IP and the OSI model

➤ Network Protocol:

- *A protocol is a set of rules and procedures that determine how a computer system receives and transmits data.*

✓ TCP/IP Protocol:

- Transmission Control Protocol / Internet Protocol.
- It is the basic communication language or protocol of the Internet.
- TCP/IP is a two-layer program, the higher layer **Transmission Control Protocol (TCP)** manages the assembling of a message or file into smaller packets that are transmitted over the internet.
- The lower layer **Internet Protocol (IP)** handles the address part of each packet so that it gets to the right destination.

✓ HTTP Protocol:

- Hypertext Transfer Protocol.
- It provides a standard for web browsers and servers to communicate.
- The HTTP is an application layer network protocol built on top of TCP.
- HTTP clients (web browsers) and servers communicate via HTTP request and response messages.

✓ FTP Protocol:

- File Transfer Protocol.
- It is a standard Internet Protocol for transmitting files between computers on the internet.
- FTP is an application protocol that uses the Internet's TCP/IP protocols.
- It is also commonly used to download programs and other files to your computer from other servers.

✓ SMTP Protocol:

- Simple Mail Transfer Protocol.
- It is a TCP/IP protocol used in sending and receiving e-mail.
- It is limited in its ability to queue messages at the receiving end; it is usually used with one of two other protocols such as POP3 or IMAP.

✓ SLIP:

- Serial Line Internet Protocol was the first protocol for relaying the IP packets over dial-up lines.
- It defines an encapsulation mechanism, but little else.
- There is no support for dynamic address assignment, link testing or multiplexing different protocols over a single link.

✓ PPP:

- Point to Point Protocol is the standard for transmission of IP packets over serial lines.
- The PPP is currently the best solution for dial-up internet connections, including ISDN.
- PPP is a layered protocol, starting with a link control protocol (LCP) for link establishment, configuration and testing.
- PPP supports both synchronized and unsynchronized lines.

➤ Types of network:

- A computer network means a group of networked components, i.e., computers are linked by means of a communication system.
- There are three types of network.
 - Local Area Network (LAN)
 - Wide Area Network (WAN)
 - Metropolitan Area Network (MAN)

✓ Local Area Network:

- Privately owned small networks that are confined to a localized area (e.g., an office, a building or a factory) are known as Local Area Networks (LANs).
- The key purpose of a LAN is to serve its users in resource sharing.

- The hardware as well as software resources are shared through LANs.
 - LAN users can share data, information, programs, printers, hard disk, modems, etc.
 - It is fast with speed from 10 MBPS to 10 GBPS.
 - LAN Configuration consists of:
 - **A File Server:** Stores all of the software that controls the network, as well as the software that can be shared by the computers attached to the network.
 - **A Workstation:** Computers connected to the file server. These are less powerful than the file server.
 - **Cables:** Used to connect the network interface cards on each computer.
- ✓ **Metropolitan Area Network (MAN)**
- Metropolitan Area Network is the networks spread over a city.
 - MAN typically covers an area of between 5 and 50 KM.
 - The purpose of a MAN is also the sharing of hardware and software resources among its users.
 - For example, cable TV networks that are spread over a city can be termed as metropolitan area networks.
- ✓ **Wide Area Network (WAN)**
- The networks spreads across countries are known as WANs.
 - A wide Area Network (WAN) is a group of computers that are separated by large distances and tied together.
 - The largest WAN in existence is the internet.
 - It can even be a group of LAN that are spread across several locations and connected and together to look like one big LAN.
 - The WANs link computers to facilitate fast and efficient exchange of information at lesser costs and higher speeds.

➤ Difference between LAN and WAN

SI No	LAN	WAN
1	Local Area Network	Wide Area Network
2	Diameter of not more than a few kilometre	Span entire country
3	A total data rate of at least several mbps	Data rate is less compared to LAN
4	Complete ownership by a single organization	Owned by multiple organization
5	Very low error rates	Comparatively high error rates.

➤ Network Topologies

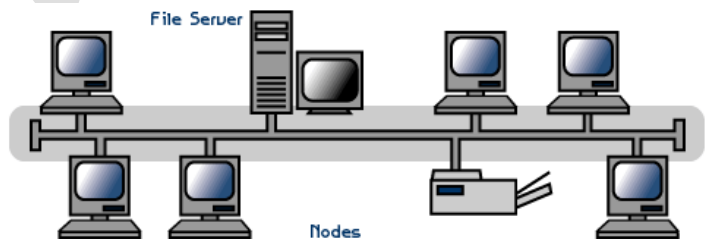
- Network Topology refers to the arrangement of computers and other devices in a network.
- Need for Topologies are: Cost, Flexibility, and Reliability.
- Network topologies can be classified as follows:
 1. Bus Topology
 2. Star Topology
 3. Ring Topology
 4. Mesh Topology
 5. Hybrid Topology

✓ Linear or bus Topology:

- A linear bus topology consists of a main run of cable with a terminator at each end.
- All nodes (file server, workstations, and peripherals) are connected to the linear cable.
- In the bus network topology, every workstation is connected to a main cable called the **bus**.
- Therefore, in effect, each workstation is directly connected to every other workstation in the network.

• Advantages of a Linear Bus Topology

- Easy to connect a computer or peripheral to a linear bus.
- Requires less cable length than a star topology.



• Disadvantages of a Linear Bus Topology

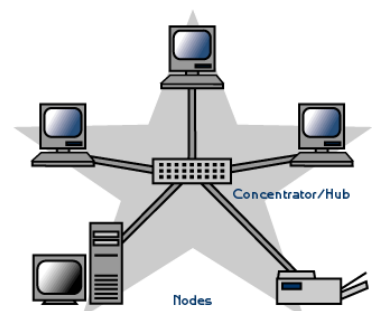
- Entire network shuts down if there is a break in the main cable.
- Terminators are required at both ends of the backbone cable.
- Difficult to identify the problem if the entire network shuts down.
- Not meant to be used as a stand-alone solution in a large building.

✓ Star Topology

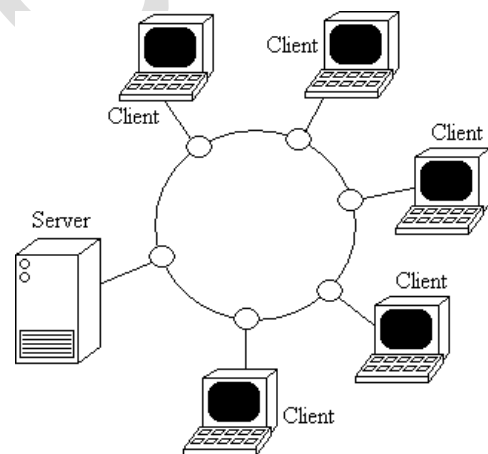
- In this type of topology, all the computers are connected to a single hub or a switch through a cable. This hub is the central node and all others nodes are connected to the central node.

• Advantages of a Star Topology

- Easy to install and wire.
- No disruptions to the network when connecting or removing devices.



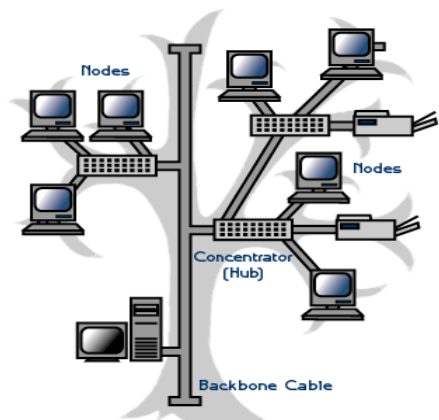
- Easy to detect faults.
- **Disadvantages of a Star Topology**
 - Requires more cable length than a linear topology.
 - If the hub, switch, or concentrator fails, nodes attached are disabled.
 - More expensive than linear bus topologies because of the cost of the hubs, etc.
- ✓ **Ring topology**
 - In a ring topology, all computers are connected via cable that loops in a ring or circle.
 - A ring topology is a circle that has no start and no end.
 - Each node connected to two neighboring computers.
 - Data accepted from one node transmitted to another.
 - Data travels in one direction, from the node to node around the ring.
 - Signal amplified at each node before being passed.



- **Advantages of Ring Topology**
 - Short cable length
 - No wiring closet space required
 - Suitable for optical fibers.
 - Each client has equal access to resources.
- **Disadvantages**
 - Node failure causes network failure
 - Difficult to diagnose faults
 - Network reconfiguration is difficult

✓ **Tree Topology:**

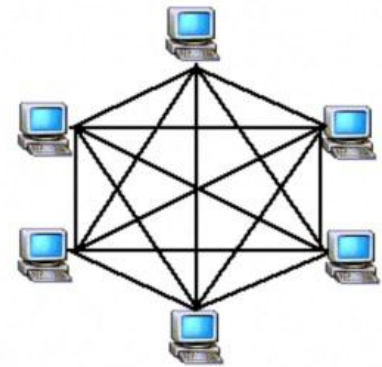
- A tree topology combines characteristics of linear bus and star topologies.
- It consists of groups of star-configured workstations connected to a linear bus backbone cable.
- The tree network topology uses two or more star networks connected together.
- The central computers of the star networks are connected to a main bus. Thus, a tree network is a bus network of star networks.
- Best suited for applications having hierarchical flow of data and control.
- **Advantages of a Tree Topology**
 - Point-to-Point wiring for individual segments.



- Supported by several hardware and software vendors.
- Network can be easily extended.
- **Disadvantages of a Tree Topology**
 - Use large cable length.
 - If the backbone line breaks, the entire segment goes down.
 - More difficult to configure and wire than other topologies.

✓ **Mesh Topology:**

- In this topology each node is connected to two or more than two nodes.
- It is a **point-to-point** connection to other nodes or devices.
- Traffic is carried only between two devices or nodes to which it is connected.
- This topology is robust, provides security and privacy.
- Overall cost of this network is too high.



➤ **Data Communication Terminologies:**

- **Data channel:** The information / data carry from one end to another in the network by channel.
- **Baud & bits per second (bps):** It's used to measurement for the information carry of a communication channel.
- **Bandwidth:** It is amount of information transmitted or receives per unit time. It is measuring in Kbps/Mbps etc unit.

➤ **Transmission Medium:**

- The first layer of computer networks is dedicated to the transmission media.
- Due to variety of transmission media and network writing methods, selecting the most appropriate media can be confusing.
- The factors to be considered are:
 - Transmission rate, Distance, cost, easy of installation and resistance to environmental condition.
- There are two type of transmission media:
 - a. Guided
 - b. Unguided
- Guided media are:
 - Twisted Pair: Unshielded Twisted Pair (UTP), Shielded Twisted pair (STP)
 - Co-axial cable: Thinnet, Thicknet

- Optical Fiber cable
- Unguided media are:
 - Radio wave Transmission
 - Microwave Transmission
 - Satellite Communication
 - Infrared
 - Laser

➤ Twisted Pair Cable

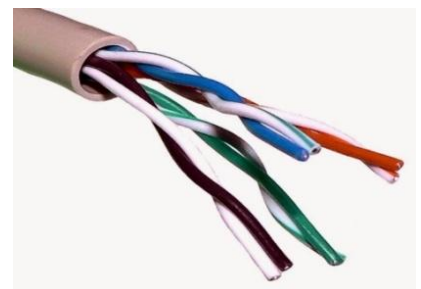
- One of the oldest and still most common transmission media is twisted pair.
- A twisted pair consists of two insulated copper wires.
- The wires are twisted together in helical form.
- Twisting is done because two parallel wires constitute a fine antenna.
- The most common application of the twisted pair is the telephone system (RJ-11).
- Twisted pair can run several kilometers without amplification, but for longer distance the signal becomes too weak and repeaters are needed.
- Due to low cost, twisted pairs are widely used.
- Twisted pairs are used in LAN (RJ-45)

✓ Types of Twisted Pair Cable:

- There are two types of twisted pair cables available. These are:

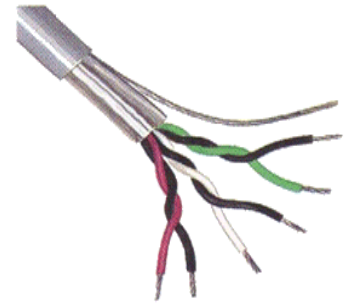
1. Unshielded Twisted Pair (UTP) Cable:

- UTP is the copper media inherited from telephone, which is being used for increasingly higher data rates.
- A UPT cable contains 2 to 4200 twisted pair.
- UTP is flexible, low cost media; it can be used for voice or data communication.
- It is available in the following five categories:
 1. CAT1: Voice-Grade communications only; No data transmission
 2. CAT2: Data-grade transmission up to 4 Mbps
 3. CAT3: Data-Grade transmission up to 10 Mbps
 4. CAT4: Data-grade transmission up to 16 Mbps
 5. CAT5: Data-grade transmission up to 1000 Mbps
- The UTP cables can have a maximum segment length of 100 meters.



2. Shielded Twisted Pair (STP) Cable:

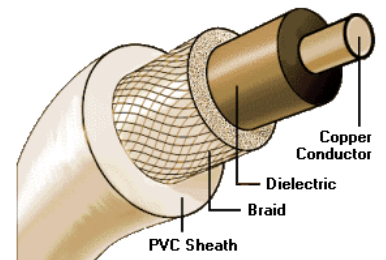
- This type of cables comes with shielding of individual pairs of wires, which further protects it from external interference.
- But these also, like UTP, can have a maximum segment length of 100 meters.
- The advantage of STP over UTP is that it offers greater protection from interference and crosstalk due to shielding.
- But it is definitely heavier and costlier than UTP and requires proper grounding at both ends.



Advantages	Disadvantages
<ul style="list-style-type: none"> • Easy to install • Flexible • It is very inexpensive 	<ul style="list-style-type: none"> • Data transmission rate are very low • It is incapable to carry a signal over long distance without the use of repeaters • Low bandwidth

➤ **Coaxial Cable**

- This type of cable consists of a solid wire core surrounded by one or more foil or wire shields, each separated by some kind of plastic insulator.
- The inner core carries the signal, and the shield provides the ground.
- The coaxial cable has high electrical properties and is suitable for high speed communication.
- While it is less popular than twisted pair, it is widely used for television signals.



✓ **Types of Coaxial Cables:**

- The two most commonly used types of coaxial cables are Thicknet and Thinnet.
 1. **Thicknet:** This form of coaxial cable is thicker than Thinnet. The Thicknet coaxial cable segments (while joining nodes of a network) can be up to 500 meters long.
 2. **Thinnet:** This form of coaxial cable is thinner and it can have maximum segment length of 185 meters i.e. using these cables, nodes having maximum distance of 185 meters can be joined.

Advantages	Disadvantages
<ul style="list-style-type: none"> • Data transmission rate is better compared to Twisted pair • Used for broadband connection. • Higher bandwidth up to 400 Mbps 	<ul style="list-style-type: none"> • Difficult to install, manage and reconfigure. • Expensive than twisted pair

➤ Optical Fibers

- Optical Fibers consist of thin strands of glass or glass like material which are so constructed that they carry light from a source at one end of the fiber to a detector at the other end.
- The light sources used are either light emitting diodes (LEDs) or LASER Diodes (LDs).
- It transmits light rather than electronic signals eliminating the problem of electrical interference.
- OFC has ability to transmit signals over much longer distances than coaxial cable and twisted pair.
- The bandwidth of the medium is potentially very high. For LEDs, this range is between 20-150 Mbps and higher rates are possible using LDs.
- It also capacity to carry information at vastly greater speed.



Advantages	Disadvantages
<ul style="list-style-type: none"> • Transmit data over long distance with high security. • Data transmission is high. • Provide better noise immunity. • Bandwidth is up to 100 Gbps 	<ul style="list-style-type: none"> • Difficult to install • Expensive as compared to other guided media • Difficult to repair.

➤ Comparison table of Guided Transmission media:

Cable Parameter	Twisted Pair Cable	Co-axial Cable	Optical Fibre Cable
Data Transfer Rate	10 Mbps – 10 Gbps	100 Mbps	More than 100 Gbps
Data Transfer Range	100 Meters	185 Mts – 500 Mts	Large distance
Interference	More	Less than T.P	Nil
Cost of cable	Lest cost	More than T.P	Very expensive

➤ Radio Wave

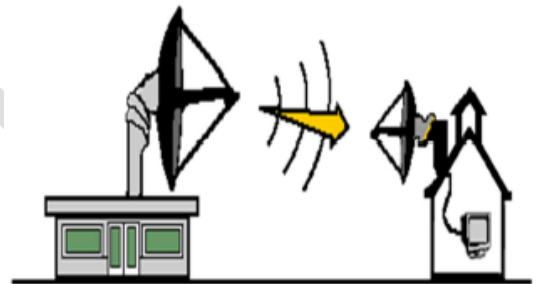
- The transmission making use of radio frequencies is termed as radio-wave transmission.
- Any radio setup has two parts:
 - a. The transmitter
 - b. The receiver
- The transmitter takes some sort of message, encodes it onto a sine wave and transmits it with radio wave.
- The receiver receives the radio wave and decodes the message from the sine wave it receives.

- Both the transmitter and receiver use antennas to radiate and capture the radio signals.

Advantages	Disadvantages
<ul style="list-style-type: none"> • Provide mobility • Inexpensive. • It proves cheaper than digging trenches for laying cables. • Free from land acquisition rights. 	<ul style="list-style-type: none"> • It is an insecure communication. • Susceptible to weather effects like rains, thunder storms etc

➤ **Microwave:**

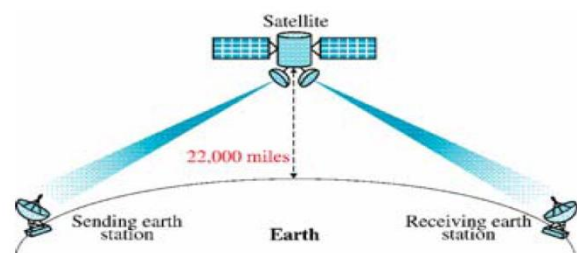
- Microwave transmission is line of sight transmission.
- The transmit station must be in visible contact with the receive station.
- This sets a limit on the distance between stations depending on the local geography.
- Microwave operates at high operating frequencies of 3 to 10 GHz.
- This allows carrying large quantities of data due to their large bandwidth.



Advantages	Disadvantages
<ul style="list-style-type: none"> • Maintenance easy than cables. • Suitable when cable cannot be used. • Free from land acquisition rights. • Low cost land purchase (Tower occupies small area) • Carry high quantities of information due to their high operating frequencies. 	<ul style="list-style-type: none"> • Repeaters are required for long distance communication. • Less Bandwidth available. • Reflected from flat surfaces like water and metals.

➤ **Satellite communication:**

- A satellite consists of transponders (unit that receive on one frequency and retransmit on another) that are set in geostationary orbits directly over the equator.
- Satellite communication is special case of microwave relay system.
- These geostationary orbits are 22,000 - 36,000 Km from the Earth’s surface.
- The uplink is the transmitter of data to the satellite.
- The downlink is the receiver of data.



- Uplinks and downlinks are also called Earth stations because they are located on the Earth.

Advantages	Disadvantages
<ul style="list-style-type: none"> • The area coverage through satellite transmission is large. • No line of sight restrictions. • Earth station which receives the signals can be fixed position or relatively mobile. 	<ul style="list-style-type: none"> • Very expensive • Installation is complex. • Signals sent to the stations can be tampered by external interference.

Apart from microwaves, radio waves and satellites, two other unguided media are also very popular. These are **infrared** and **laser** waves.

➤ Infrared:

- This type of transmission uses infrared light to send data.
- You can see the use of this type of transmission in everyday life - TV remotes, automotive garage doors, wireless speakers etc., all make use of infrared as transmission media.
- The infrared light transmits data through the air and can propagate throughout a room (bouncing off surfaces), but will not penetrate walls.
- The infrared transmission has become common in PDAs (Personal digital assistants) e.g., hand held devices like palm pilots etc.
- The infrared transmission is considered to be a secure one.

➤ Laser:

- The laser transmission requires direct line-of-sight.
- It is unidirectional like microwave, but has much higher speed than microwaves.
- The laser transmission requires the use of a laser transmitter and a photo-sensitive receiver at each end.
- The laser transmission is point-to-point transmission, typically between buildings.
- Disadvantage: It can be adversely affected by the weather.

➤ Switching techniques:

- Switching techniques are used for transmitting data across networks.
- There are three types of switching:
 - Circuit Switching
 - Message Switching
 - Packet Switching

✓ Circuit Switching:

- In this technique, first the complete physical connection between two computers is established and then data are transmitted from the source computer to the destination computer.
- That is, when a computer places a telephone call, the switching equipment within the telephone system seeks out a physical copper path all the way from sender telephone to the receiver's telephone.
- The important property of this switching technique is to setup an end-to-end path connection between computers before any data can be sent.

✓ Message Switching:

- In this technique, the source computer sends data or the message to the switching office first, which stores the data in its buffer.
- It then looks for a free link to another switching office and then sends the data to this office.
- This process is continued until the data are delivered to the destination computers.
- It is also known as store and forward. i.e., store first in switching office, forward later, one jump at a time.

✓ Packet Switching:

- Packet switching can be seen as an option that tries to combine the advantages of circuit and message switching and to minimize the disadvantage of both.
- In Packet switching, a message is broken into smaller parts called **packets**.
- A fixed size of packet which can be transmitted across the network is specified.

➤ Communication Modes:

- The way in which data is transmitted from one place to another is called *data transmission mode*.
- It is also called the *data communication mode*.
- It indicates the direction of flow of information.
- Sometimes, data transmission modes are also called *directional modes*.
- Different types of data transmission modes are as follows:
 1. Simplex mode
 2. Half-duplex mode
 3. Full-duplex mode

✓ Simplex Mode

- In simplex mode, data can flow in only one direction.
- In this mode, a sender can only send data and cannot receive it.

- Similarly, a receiver can only receive data but cannot send it.
- Data sent from computer to printer is an example of simplex mode.
- In simplex mode, it is not possible to confirm successful transmission of data.
- It is also not possible to request the sender to re-transmit information.
- This mode is not widely used.

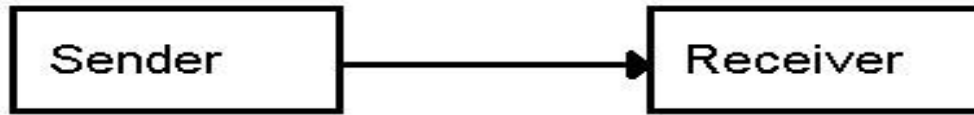


Figure: Simplex Mode

✓ Half-Duplex Mode

- In half-duplex mode, data can flow in both directions but only in one direction at a time.
- In this mode, data is sent and received alternatively.
- It is like a one-lane bridge where two-way traffic must give way in order to cross the other.
- The Internet browsing is an example of half duplex mode.
- The user sends a request to a Web server for a web page.
- It means that information flows from user's computer to the web server.
- Web server receives the request and sends data of the requested page.
- The data flow the Web server to the user's computer.
- At a time a user can a request or receive the data of web page.

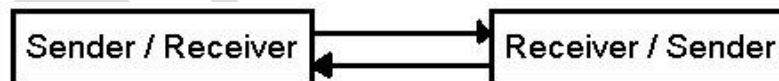


Figure: Half-Duplex Mode

✓ Full-Duplex Mode

- In full duplex-mode, data can flow in both directions at the same time.
- It is the fastest directional mode of data communication.
- The telephone communication system is an example of full-duplex communication mode.
- Two persons can talk at the same time.

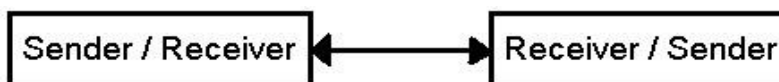


Figure: Full-Duplex Mode

➤ Network Devices:

✓ Modem:

- Modem means **Modulation/ Demodulation**.

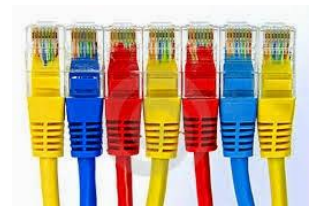
- A modem is a computer peripheral that allows you to connect and communicate with other computers via telephone lines.
- **Modulation:** A modem changes the digital data from your computer into analog data, a format that can be carried by telephone lines.
- **Demodulation:** The modem receiving the call then changes the analog signal back into digital data that the computer can digest.
- The modem modulates the signal at the sending end and demodulates at the receiving end.
- Modems are of two types:

- **Internal modems:** The modems that are fixed within the computer
- **External modems:** The modems that are connected externally to a computer as other peripherals are connected.



✓ RJ-45:

- RJ-45 is short for Registered Jack-45 is an eight-wire connector, which is commonly used to connect computers on the local area networks i.e., LANs especially Ethernets.
- The standard connector for unshielded twisted pair cabling is an RJ-45 connector.



✓ Ethernet Card:

- The computer that are part of Ethernet, have to install our special card called Ethernet card.
- It is LAN architecture developed by Xerox Corp association with DEC and Intel.
- It make use of Bus or Star topology and data transfer rates of 10 Mbps.
- An Ethernet card contains connections for either coaxial or twisted pair cables (or both).
- If it is designed for coaxial cable, the connection will be BNC.
- If it is designed for twisted pair, it will have a RJ-45 connection.
- Some Ethernet cards also contain an AUI connector. This can be used to attach coaxial, twisted pair, or fiber optical cables to an Ethernet card.



✓ Hub:

- A hub is a hardware device used to connect several computers together.



- A concentrator is device that provides a central connection point for cables from workstations, servers, and peripherals.
- In a star topology, twisted-pair wire is run from each workstation to a central concentrator.
- Types of hub:
 - **Active hubs:**
 - It electrically amplifies the signal as it moves from one connected device to another.
 - Active concentrators are used like repeaters to extend the length of a network.
 - **Passive hubs:**
 - It allows the signal to pass from one computer to another without any change.

✓ **Switch:**

- The switch is a telecommunication device grouped as one of computer network components.
- The switch is like Hub but built in with advanced features.
- The switch connects the source and destination directly which increases the speed of the network.

✓ **Repeater:**

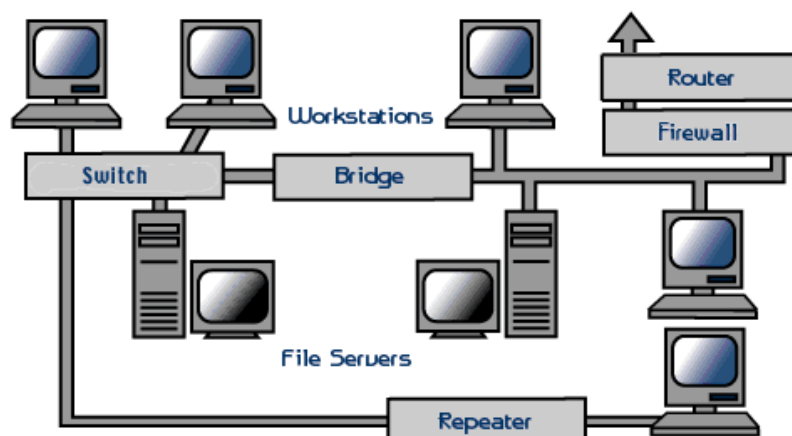
- A Repeater is network device that amplifies and restore signals for long-distance transmission.
- It is used in long network lines, which exceed the maximum rated distance for a single run.

✓ **Bridge:**

- A Bridge is a network device that establishes an intelligent connection between two local networks with the same standard but with different type's cables.

✓ **Router:**

- A router works like a bridge but can handle different protocols.
- A Router is a network device that is used to separate different segments in a network to improve performance and reliability.



✓ Gateway:

- The term gateway is applied to a device, system or software application which has internetwork capability of joining dissimilar network.
- It is node on network that provides entry to another network.
- It performs data translation and protocol conversions which is suitable to other network.
- Example: It needs to convert Ethernet traffic from the LAN to SNA (System Network Architecture). It then routes SNA traffic to Mainframe. When Mainframe answers, the reverse process occurs.
- Gateway can be implemented on software, hardware or a combination of both.
- Gateway is that only the data format is translated, not the data itself.

➤ Wireless and Mobile Computing:

- *Wireless* refers to the method of transferring information between a *computing device*, such as a personal data assistant (PDA), and a *data source*, such as an agency database server, without a physical connection.

✓ Wireless communication:

- Wireless communication is simply data communication without the use of landlines.
- This may involve cellular telephone, two way radio, fixed wireless, LASER or satellite communication.
- Mobile computing means that the computing device is not continuously connected to the base or central network.
- Mobile devices include PDAs, Laptop computers and smart phones.

✓ GSM:


- GSM is short for Global System for Mobile communications, which is one of the leading digital cellular systems.
- The GSM standard for digital cell phones was established in Europe in the mid 1980s.
- GSM uses narrowband TDMA, which allows eight simultaneous calls on the same radio frequency.
- TDMA is short for Time Division Multiple Access, a technology for delivering digital wireless service using time-division multiplexing (TDM).

✓ TDMA:

- Time Division Multiple Access.

- TDMA works by dividing a radio frequency into time slots and then allocating slots to, multiple calls. In this way, a single frequency can support multiple, simultaneous data channels.
- ✓ **SIM card:**
 - The SIM - Subscriber Identity Module - is a chip card; the size of a first class postage stamp.
 - A SIM is a computer chip that gives a cellular device its unique phone number.
 - It has memory (16 to 64 KB), processor and the ability to interact with the user.
- ✓ **CDMA:**
 - CDMA is short for *Code-Division Multiple Access*, a digital cellular technology that uses *spread-spectrum* techniques.
 - CDMA is a form of *spread spectrum*, which simply means that data is sent in small pieces over a number of the discrete frequencies available for use at any time in the specified range.
- ✓ **WLL**
 - Wireless in Local Loop (WLL or WiLL),
 - It is meant to serve subscribers at homes or offices.
 - In WLL services, the telephone provided is expected to be as good as wired phone.
 - Its voice quality must be high - a subscriber carrying out long conversation must not be irritated with quality; one must be able, to use speakerphones, cordless phones and parallel phones.
- ✓ **GPRS**
 - GPRS stands for General Packet Radio Service.
 - GPRS is used for wireless communication using a mobile device.
 - With the service you can access the internet, send emails and large data, download games and watch movies.
- **EDGE:**
 - The new EDGE air interface has been developed specifically to meet the bandwidth needs of 3G. *Enhanced Data rates for Global Evolution* (EDGE) is a radio based high-speed mobile data standard.
 - It allows data transmission speeds of 384 Kbps.
 - EDGE was formerly called GSM384. This means a maximum bit rate of 48 kbps per time slot.
 - EDGE is considered an intermediate step in the evolution of 3G WCDMA.

The “G” in wireless networks refers to the “Generation” of the underlying wireless network technology.



COMPARISON OF 1G TO 5G TECHNOLOGIES

Technology	1G	2G/2.5G	3G	4G	5G
Deployment	1970/1984	1980/1999	1990/2002	2000/2010	2014/2015
Bandwidth	2kbps	14-64kbps	2mbps	200mbps	>1gbps
Technology	Analog cellular	Digital cellular	Broadbandwidth/ cdma/ip technology	Unified ip & seamless combo of LAN/WAN/WLAN/PAN	4G+WWWW
Service	Mobile telephony	Digital voice, short messaging	Integrated high quality audio, video & data	Dynamic information access, variable devices	Dynamic information access, variable devices with AI capabilities
Multiplexing	FDMA	TDMA/CDMA	CDMA	CDMA	CDMA
Switching	Circuit	Circuit/circuit for access network & air interface	Packet except for air interface	All packet	All packet
Core network	PSTN	PSTN	Packet network	Internet	Internet
Handoff	Horizontal	Horizontal	Horizontal	Horizontal & Vertical	Horizontal & Vertical

➤ Applications in Networking:

✓ SMS:

- *Short Message Service (SMS)* is the transmission of short text messages to and from a mobile phone, fax machine and/or IP address.
- Messages must be no longer than some fixed number of alpha-numeric characters and contain no images or graphics.

✓ E-mail:

- Electronic mail (e-mail) is sending and receiving message by computer.
- **Advantages:**
 - Low cost: Electronic mail is an extremely cost-effective way to move information around, especially when it must be moved quickly.
 - Speed: Electronic mail can be delivered almost as fast as the wire can carry it.

✓ Voice Mail:

- The voice-mail refers to e-mail systems that support audio.
- Users can leave spoken messages for one another and listen to the messages by executing the appropriate command in the e-mail system.

✓ Chat:

- Chatting is the most fantastic thing on Internet.

- Chatting is like a text-phone.
 - In telephonic conversations, you say something, people hear it and respond, and you hear their responses on the spot and can reply instantly.
- ✓ **Video Conferencing:**
- A two-way videophone conversation among multiple participations is called Video Conferencing.
- **Wi-fi:**
- Wi-Fi is short for Wireless Fidelity, which lets you connect to the internet without a direct line from your PC to the ISP.
 - For Wi-Fi to work, you need:
 - A broadband internet connection.
 - A wireless router, which relays your internet connection from the “wall” to the PC.
 - A laptop or desktop with a wireless internet card or external wireless adapter.
- **Wi-Fi Hotspots:**
- A hotspot is a venue that offers Wi-Fi access.
 - The public can use a laptop, Wi-Fi phone or other suitable portable devices to access the internet through a Wi-Fi hotspot.
 - Hotspots are public locations with free or fee-based wireless internet access.
- **WiMax:**
- WiMax is wireless digital communication system.
 - WiMax can provide Broadband Wireless Access (BWA) up to 30 miles for fixed stations and 3-10 miles for mobile stations.
 - WiMax requires a tower called WiMax Base Station, similar to cell phone tower, which is connected to the internet using a standard wired high-speed connection.
- **VIRUS:**
- VIRUS – “Vital Information Resource Under Siege”.
 - A computer virus is a computer program that can replicate itself and spread from one computer to another.
 - Depend on the nature of a virus, it may cause damage of your hard disk contents, and/or interfere normal operation of your computer.
- ✓ **Characteristics of a computer virus:**

- It is able to replicate.
 - Reduced memory or disk space.
 - Modification of data.
 - Files overwritten or damaged.
 - Hard drive erased.
- ✓ **Types of Virus:**
- **File Infectors:**
 - Infect executable files or attach themselves to a program file and create duplicate files.
 - **Boot sector Virus:**
 - Install themselves on the beginning tracks of the hard drive or the Master Boot record.
 - **Macro Virus:**
 - Infect data file like spread sheets or databases of several software packages.
 - **Network Virus:**
 - E-mail or any data transfer files to spread themselves on the network.
 - **Trojan Horse:**
 - A Trojan Horses is code hidden in a program such as a game as spreadsheet that looks safe to run but has hidden side effects.
 - **Worm:**
 - A worm is a program designed to replicate. The program may perform any variety of additional tasks as well.
- ✓ **How Computer Viruses Spread?**
- It moves from computer to computer by attaching themselves to files or boot records of disks.
 - A virus travel from file to another on the same computer if the infected file executed, from computer memory to a file on the disk, on a disk that is carried from one computer to another.
- ✓ **Damage:**
- Can destroy file allocation table (FAT)
 - Can create bad sectors on the disk
 - Can decrease the space on the hard disks by duplicating file.
 - Can format specific tracks on the disk.
 - Can destroy specific executable files
 - Can cause the system to hang.
- ✓ **Virus Protection:**
- The following guidelines to lead virus free computing life.
 - Never use a CD without scanning it for viruses.

- Always scan files downloaded from the internet.
- Never boot your PC from floppy.
- Write protect your disks and make regular backup..
- Use licensed software.
- Password protects your PC.
- Install and use antivirus software.
- Keep antivirus software up to date.
- Some of the antivirus are: Kaspersky, Quick Heal, K7, Norton 360, AVG, Avasta, McAfee.

➤ Network Security:

- Network security consists of the provisions and policies adopted by a network administrator to prevent and monitor unauthorized access, misuse, modification of a computer network and network accessible resources.
- The problem encountered under network security are:
 1. **Physical Security holes:** When individuals gain unauthorized physical access to a computer and tamper with files.
 2. **Software Security holes:** When badly written programs or privileged software are compromised into doing things that they shouldn't be doing.
 3. **Inconsistent usage holes:** When a system administrator assembles a combination of hardware and software such that the system is seriously flawed from a security point of view.

➤ Protection Methods:

1. **Authorization** - Authorization is performed by asking the user a legal login ID. If the user is able to provide a legal login ID, He/she is considered an authorized user.
2. **Authentication** - Authentication also termed as password protection as the authorized user is asked to provide a valid password and if he/she is able to do this, he/she considered to be an authentic user.
3. **Encryption Smart cards**– conversion of the form of data from one form to another form. An encrypted smart card is a hand held smart card that can generate a token that a computer system can recognize. Every time a new and different token is generated, which even though carked or hacked, cannot be used later.
4. **Biometric System** – It involves unique aspect of a person's body such as Finger-prints, retinal patterns etc to establish his/her Identity.
5. **Firewall** - A system designed to prevent unauthorized access to or from a private network is called firewall. It can be implemented in both hardware and software or combination or both.

- There are several types of firewall techniques-
- **Packet filter** - accepts or rejects of packets based on user defined rules.
- **Application gateway** - security mechanism to specific application like FTP and Telnet servers.
- **Circuit level gateway** - applies security mechanism when a connection is established.
- **Proxy Server** - Intercepts all messages entering and leaving the network.

✓ **Cookies :**

- Cookies are messages that a web server transmits to a web browser so that the web server can keep track of the user's activity on a specific web site. Cookies have few parameters name, value, expiration date.

✓ **Hackers and crackers :**

- **Hackers** are more interested in gaining knowledge about computer systems and possibly using this knowledge for playful pranks.
- **Crackers** are the malicious programmers who break into secure systems.

✓ **Cyber Law:**

- It is a generic term, which refers to all the legal and regulatory aspects of internet and the World Wide Web.

✓ **India's IT Act:**

- In India the cyber laws are contained in the IT Act 2000. Aims to provide legal infrastructure for e-commerce in India by governing transactions through internet and other electronic medium.

CHAPTER 15 – NETWORKING CONCEPTS BLUE PRINT				
VSA (1 marks)	SA (2 marks)	LA (3 Marks)	Essay (5 Marks)	Total
02 Question	01 Question	-	01 Question	04 Question
Question no 07, 08	Question no 18	-	Question no 37	09 Marks

Important Questions

➤ **1 Marks Question:**

1. Expand FTP. [March 2015, June 2017]
2. What is network topology? [March 2015, March 2016]

3. Expand URL. [June 2015]
4. Define bus topology. [June 2015]
5. Define Local Area Networking. [March 2016]
6. Define Networking. [June 2016]
7. Give an example for full duplex communication mode. [June 2016]
8. What is Chatting? [March 2017]
9. What is a Server? [March 2017]
10. Give the syntax of URL. [June 2017]

Extra One Mark Questions

11. Name the first computer network?
12. What is client/workstation?
13. What is server?
14. What are the various types of network?
15. What is Modem and Hub?
16. What are cookies?
17. What hackers and crackers?
18. Expand the terms: TCP/IP, GPRS, GSM, EDGE, Wi-Fi.

➤ 2 Marks Question:

1. Explain half duplex communication mode. [March 2015]
2. Mention any two antivirus software. [March 2016]
3. Write the difference between LAN and WAN. [June 2016]
4. What is communication (transmission) mode? Explain simplex mode. [March 2017]
5. Write the difference between half duplex and full duplex communication modes. [June 2017]

➤ 5 Marks Question:

1. Explain any five network devices. [March 2015]
2. Give the measures for prevention virus. [June 2015, June 2017]
3. Explain the network security in detail. [March 2016, June 2016]
4. What is networking? Explain the goals of networking. [June 2017]

Chapter-16

INTERNET AND OPEN SOURCE CONCEPTS

➤ Introduction:

- **Internetwork:** An internetwork is a collection of individual networks, connected by intermediate networking devices, that functions as a single large network.
- **Classification of Internetworks:**
 - **Internet:** The globe public network.
 - **Intranets:** The wholly owned/private internetworks.
 - **Extranets:** The hybrid internetworks: private networks/ internetworks connected through the internet
- *The term “open source” software is used to refer to those categories of software/ programs whose licenses do not impose much condition.*

➤ Terminology and Definitions:

✓ Free Software:

- *Free software means the software is freely accessible and can be freely used, changed, improved, copied and distributed by all who wish to do so.*
- No payments are needed to be made for free software.

✓ Open Source Software:

- Open Source Software, on the other hand, can be freely used but it does not have to be free of charge.
- Open Source doesn't just access to the source code. The distribution terms of Open Source Software must comply with the following criteria.
 1. Free Redistribution
 2. Source Code
 3. Derived works
 4. No Discrimination Against persons or groups
 5. No Discrimination Against Fields or Groups
 6. Distribution of License
 7. License Must Not be Specific to a Product
 8. The license must Not Restrict other Software
 9. License Must Be technology Natural

✓ **OSS:**

- OSS refers to open source software, which refers to software whose source code is available to customers and it can be modified and redistributed without any limitation.

✓ **FLOSS:**

- FLOSS refers to Free Libre and open Source Software or to Free Livre and Open Source Software.
- The term FLOSS is used to refer to software which is both free software as well as open source software.
- Here the words Libre (a Spanish word) and Livre (a Portuguese's word) mean freedom.

✓ **GNU:**

- GNU is a Unix-like computer operating system developed by the GNU project.
- It is composed wholly of free software.
- It refers to GNU's Not Unix. GNU Project emphasizes on freedom and thus its logo type show a GNU, an animal living in freedom

✓ **FSF:**

- FSF is Free Software Foundation. FSF is a non-profit organization created for the purpose of supporting free software movement.
- Richard Stallman founded FSF in 1985 to support GNU project and GNU licenses.

✓ **OSI:**

- OSI is Open Source Initiative. It is an organization dedicated to cause of promoting open source software.
- Bruce Perens and Eric Raymond were the founders of OSI that was founded in February 1998.

✓ **W3C:**

- W3C is acronym for World Wide Web Consortium.
- W3C is responsible for producing the software standards for World Wide Web.
- The W3C was created by Tim Berners-Lee in 1994.

✓ **Proprietary Software:**

- Proprietary software or closed source software is the software that is neither open nor freely available.
- Its use is regulated and further distribution and modifications is either forbidden or requires special permission by the supplier or vendor.
- Source of proprietary software is normally not available.

✓ **Freeware:**

- The term freeware has no clear definition, but is generally used for software, which is available free of cost and which allows copying and further distribution, but not modification and whose source code is not available.
- Freeware is distributed in Binary Form (ready to run) without any licensing fees.

✓ **Shareware:**

- Shareware is software, offered as trial version (for limited period of time) with certain features only available after the license is purchased.
- Its source code is not available and modifications to the software are not allowed.

✓ **WWW (World Wide Web).**

- The World Wide Web (WWW) is a set of protocols that allows you to access any documents on the Net through a naming system based on URLs.
- WWW also specifies a way -- the Hypertext Transfer Protocol (HTTP) - to request and send a document over the internet.
- Attributes of WWW
 - **User friendly** - www resources can be easily used with the help of browser.
 - **Multimedia documents** - A web page may have graphic, audio, video, and animation etc at a time.
 - **Hypertext and hyperlinks** - The dynamic links which can move towards another web page is hyperlink.
 - **Interactive** - www with its pages support and enable interactivity between users and servers.
 - **Frame** - display of more than one section on single web page.
- **Advantages of WWW:**
 - Availability of mainly free information.
 - Low cost of initial connection.
 - It is accessible from anywhere.
 - Facilities rapid interactive communication.
 - Facilities the exchange of huge information.
 - Facilitates the establishment of professional contact.
 - Facilitates access to different sources of information.
- **Disadvantages of WWW:**
 - Danger of overload and excess information
 - It requires an efficient information search strategy

- The search can be slow
 - It is difficult to filter and prioritize information
 - No guarantee of finding what one is looking for
 - Net becomes overloaded because of large number of users
 - No quality control over available data
- ✓ **Telnet (Remote Login):**
- Telnet (Teletype network) is an order Internet utility that lets you log on to remote computer systems.
 - Telnet program gives you a character-based terminal window on another system.
 - You get a login prompt on that system. If you've permitted access, you can work on that system, just as you would if you were sitting next to it.
- ✓ **Web Browser:**
- *A Web Browser is software application that enables the user to view web pages, navigate web sites and move from one website to another.*
 - Some of the web browsers are Google Chrome, Internet Explorer, Netscape Navigator, Mozilla Firefox, and Opera.
- ✓ **Web Server:**
- An internet host computer that many store thousand of websites.
 - *A Web Server is a WWW server that responds made by web browsers.*
 - Example: Apache, IIS, PWS (Personal web server for Windows 98).
- ✓ **Web sites:**
- *A Web site is collection of web pages, images, videos and other digital assets and hosted on a particular domain on the WWW.*
 - Each web site has a unique address called URL.
- ✓ **Web page:**
- *A document that can be viewed in a web browser and residing on a website is a web page.*
 - The web pages use HTTP.
 - **Home page** - A web page that is the starting page and acts as an indexed page is home page.
 - **Web portal** - That facilitates various type of the functionality as website.
 - For example: www.pue.kar.nic.in , www.karnataka.gov.in.
- ✓ **URL and Domain :**
- The internet structure of the www is built on a set of rules called Hypertext Transfer Protocol (HTTP) and a page description language called Hypertext Markup Language (HTML).

- HTTP uses Internet addresses in a special format called a URL.
- A URL (Uniform Resources Locator) specifies the distinct address for each resource on the internet
- URLs look like this: **type://address/path**
- In URL,
 - type specifies the type of server in which the file is located,
 - address is the address of the server,
 - path is the location within the file structure of the server, the path includes the list of folders(or directories) where the desired file is located.
- Consider the URL

http://www.yahoo.com or http://www.facebook.com

✓ Domain name:

- An internet address which is a character based is called a Domain name.
- A domain name is away to identify and locate computer and resources connected to the internet.
- This type of domain name is also called hostname.
- Some most common domains are:

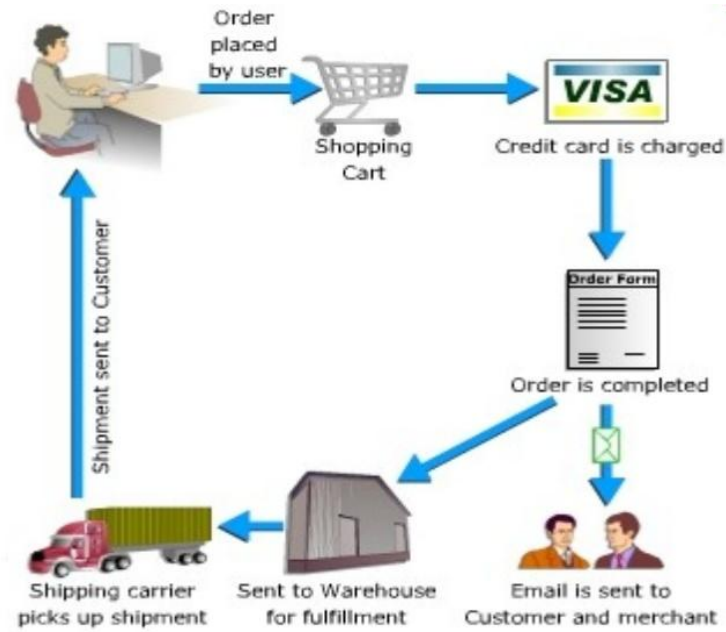
Domain ID	Meaning	Domain Cou	Meaning
.com	Commercial	.au	Australia
.gov	Government	.in	India
.mil	Military	.nz	New Zealand
.ac	Academic	.ca	Canada
.org	Organization	.jp	Japan
.edu	Education	.uk	United Kingdom

➤ E-Commerce:

- E-Commerce is the trade of goods and services with the help of telecommunications and computers.
- E-Commerce involves the automation of a variety of business to consumer transaction through reliable and secure connections.

➤ E-Commerce Process:

- A consumer uses a Web browser to connect to the home page of a merchant's Web site on the Internet.



- The consumer browses the catalog of products featured on the site and selects items to purchase. The selected items are placed in the electronic equivalent of a shopping cart.
- When the consumer is ready to complete the purchase of selected items, it provides a bill to and ship to address for purchase and delivery
- When the credit card number is validated and the order is completed at the Commerce Server site, the merchant's site displays a receipt confirming the customer's purchase.
- The Commerce Server site then forwards the order to a Processing Network for payment processing and fulfillment.

➤ Different types of E-Commerce

- Business-to-Business (B2B)
- Business-to-Consumer (B2C)
- Consumer-to-Business (C2B)
- Consumer-to-Consumer (C2C)
- Business-to-Business (B2B)

✓ Business-to-Business (B2B):

- The exchange of services, information and/or products from one business to another business partners.
- Examples: Intel selling microprocessor to Dell

✓ Business-to-Consumer (B2C):

- The exchange of services, information and/or product from business to a consumer.
- Example: Dell selling me a laptop. Websites: Flipkart, Amazon, Snapdeal.

✓ **Consumer-to-Business (C2B):**

- Customer directly contact with business vendors by posting their project work with set budget online so that needy companies review it and contact the customer directly with bid.
- Example: guru.com, freelancer.com

✓ **Consumer-to-Consumer (C2C)**

- E-commerce is simply commerce between private individuals or consumers.
- Example: Ram buying smartphone from Sham using OLX

➤ **Advantages of e-commerce**

- Buying & selling can be done online at any time (24 hours) money.
- It provides faster payments through Electronic Fund Transfer.
- Online payment reduces work of carrying money to the shop and also saves money.
- Customer can search for competitive prices quickly before purchasing the items.
- Wider choice for item selection.
- Without going to the shops customers can view the products through websites thus saves time.

➤ **Disadvantages of e-commerce**

- Initial cost is high.
- E-Commerce websites needs to be protected from virus attacks, hackers.
- Needs more security.
- Some company may charge more for shipping or other transport.
- There is the possibility of credit card number theft.
- Mechanical failures can cause unpredictable effects on the total processes.

➤ **IPR Issues:**

- IPR stands for Intellectual Property Rights.
- Intellectual Property Rights (IPR) means copyrights, patents, and trademarks, designs etc held by a person or company who have invented or designed a product.
- Copyright, trademarks, industrial designs, patents, integrated circuits are the different forms of Intellectual property.
- The main benefits of IPR are wealth creation, legitimate ownership, talent attraction, image of a trustworthy organization.
- **WIPO – World Intellectual Property Organization.**
- IPR-related issues in India like patents, trademarks, copyrights, designs are governed by the Patents Act 1970 and patents Rules 2003, Trademarks Act 1999, Trademarks Rules 2002, Copyright Act 1957, Design Act 2000 and Rules 2001.

CHAPTER 16 – INTERNET AND OPEN SOURCE CONCEPTS BLUE PRINT				
VSA (1 marks)	SA (2 marks)	LA (3 Marks)	Essay (5 Marks)	Total
01 Question	-	01 Question	-	02 Question
Question no 9	-	Question no 25	-	04 Marks

Important Questions

➤ 1 Marks Question:

1. What is Open source software? [March 2015]
2. What are Freeware? [March 2016]
3. Define E-Commerce. [June 2016]
4. What is telnet?
5. Expand OSS and FLOSS.

➤ 3 Marks Question:

1. Explain Free software.
2. What is meant by shareware? Write its limitations [March 2016]
3. What is Web browser? Mention any two web browser. [March 2015, June 2015]
4. Give the advantages of WWW.
5. Define the terms webpage, website, web server. [June 2016]
6. Explain URLs.
7. What is E-Commerce? Explain types of E-commerce. [June 2015, March 2017]
8. What are the advantages and disadvantages of E-commerce?
9. Explain IPR.

Chapter-17

WEB DESIGNING

➤ Introduction to HTML:

- HTML stands for **Hypertext Markup Language**, and it is the most widely used Language to write Web Pages.
- Hypertext refers to the way in which Web pages (HTML documents) are linked together.
- Thus the link available on a webpage is called *Hypertext*.
- HTML documents are also called **web pages**.
- Now, HTML is being widely used to format web pages with the help of different tags available in HTML.

➤ HTML Document Structure:

- A typical HTML document will have following structure: Document declaration tag

```

<HTML>
  <HEAD>
    DOCUMENT HEADER RELATED
    TAGS
  </HEAD>
  <BODY>
    DOCUMENT BODY RELATED TAGS
  </BODY>
</HTML>

```

✓ HTML Tags:

- HTML markup tags are usually called HTML tags.
- These tags are keywords (tag name) surrounded by angle braces like **<Tag Name>**.
- The first pair of tags is the start tag and the second tag is the end tag.
- End tag contains a forward slash before the tag name.
- Start tag and end tag are also called opening tags and closing tags.
- Except few tags, most of the tags have their corresponding closing tags.
- For example **<html>** has its closing tag **</html>** and **<body>** tag has its closing tag **</body>** tag.

Tags	Description
<HTML>	This tag encloses the complete HTML document and mainly comprises of document header which is represented by <HEAD>...</HEAD> and document body which is represented by <BODY>...</BODY> tags.

<HEAD>	This tag represents the document's header which can keep other HTML tags like <TITLE>, <LINK> etc.
<TITLE>	The <TITLE> tag is used inside the <head> tag to mention the document title.
<BODY>	This tag represents the document's body which keeps other HTML tags like <H1>, , <P> etc.

➤ HTML Basic Tags:

Tag Name	Description	Syntax
Heading	Different sizes for your headings	<H1>, <H2>, <H3>, <H4>, <H5>, and <H6>.
Paragraph	Way to structure your text into different paragraphs.	<P> </P>
Line Break	It starts from the next line.	
Horizontal Lines	Used to visually break up sections of a document.	<HR>

➤ HTML Text Formatting Tags:

Tag	Description	Tag	Description
	Defines bold text	<I>	Defines italic text
	Defines emphasized text	<U>	Underline
	Defines strong text	<SMALL>	Defines small text
<SUB>	Defines subscripted text	<SUP>	Defines superscripted text
<INS>	Defines inserted text		Defines deleted text

➤ HTML Images:

- Images are very important to beautify as well as to depict many complex concepts in simple way on your web page.
- This will take you through simple steps to use images in your web pages.

✓ Insert Image:

- You can insert any image in your web page by using tag.
- Following is the simple syntax to use this tag.

➤ HTML Hyper Links:

- A webpage can contain various links that take you directly to other pages and even specific parts of a given page. These links are known as hyperlinks.
- Hyperlinks allow visitors to navigate between Web sites by clicking on words, phrases, and images. Thus you can create hyperlinks using text or images available on a webpage.

** Next page**

- **Anchor element** allows you to link various WebPages or different sections on the same page. The syntax of Anchor element is given below:

<A>.....

- The various attributes of the Anchor element are HREF, NAME, TITLE, TARGET and ALT
 - **Href:** The href (**h**yperlink **r**eference) attribute specifies the location of the file or resource that you want to provide a link to.
 - **Name:** The name attribute specifies a location within the current or the existing document.
 - **Title:** The title attribute specifies a title for the file which you are providing a link.
 - **Target:** The target attribute specifies a position in the webpage where the browser displays a file.
 - **Alt:** The alt attribute specifies the alternative text which is displayed when an image used as a hyperlink is not displayed.

➤ HTML Lists:

- HTML offers web authors three ways for specifying lists of information.
- All lists must contain one or more list elements.
- Lists may contain:
 1. **** - An unordered list. This will list items using plain bullets.
 2. **** - An ordered list. This will use different schemes of numbers to list your items.
 3. **<DL>** - A definition list. This arranges your items in the same way as they are arranged in a dictionary.

➤ HTML Tables:

- The HTML tables allow web authors to arrange data like text, images, links, other tables, etc. into rows and columns of cells.
- Basic TABLE tags:
- **<TABLE> </TABLE>**

- This tag defines a table in HTML. If the BORDER attribute is present, your browser displays the table with a border.
- **<CAPTION>.....</CAPTION>**
 - This tag defines the caption for the title of the table.
- **<TR>..... </TR>**
 - This tag specifies a table row within a table.
- **<TH>..... </TH>**
 - This tag specifies a table header cell. By default the text in this cell is bold and centered.
- **<TD>..... </TD>**
 - This tag specifies a table data cell. By default the text in this cell is aligned left and centered vertically.
- Example:

```

<HTML>
<HEAD>
  <TITLE> TABLE ELEMENTS </TITLE>
</HEAD>
<BODY>
  <TABLE BORDER="1">
    <CAPTION> THIS IS THE TABLE CAPTION </CAPTION>
    <TR>
      <TH>REGNO</TH>
      <TH>NAME</TH>
      <TH>MARKS</TH>
    </TR>
    <TR>
      <TD>101</TD>
      <TD>AKASH</TD>
      <TD>535</TD>
    </TR>
    <TR>
      <TD>102</TD>
      <TD>KARTHIK</TD>
      <TD>578</TD>
    </TR>
  </TABLE>
</BODY>
</HTML>

```

**THIS IS THE TABLE
CAPTION**

REGNO	NAME	MARKS
101	AKASH	535
102	KARTHIK	578

◆ **Some of the attributes associated with various tags are:**

Attribute	Description	Syntax
BORDER	Draws a border around the table of certain pixels wide	<TABLE BORDER="4">
BGCOLOR	Specifies the background colour of the entire table	<TABLE BGCOLOR="IVORY">
ALGIN (LEFT, CENTER, RIGHT)	Identifies the horizontal alignment of a table	<TABLE ALIGN="LEFT"> <TABLE ALIGN="CENTER"> <TABLE ALIGN="RIGHT">
VALGIN (TOP, MIDDLE, BOTTOM)	Identifies the vertical alignment of a table	<TABLE VALIGN="TOP"> <TABLE VALIGN="MIDDLE"> <TABLE VALIGN="BOTTOM">
CELLSPACING	Specifies the space in pixels between cells	<TABLE CELLSPACING="4">
CELLPADDING	Specifies the space in pixels between cell border and cell data	<TABLE CELLPADDING="4">
COLSPAN	Allows a number of columns to be combined in a cell	<TD COLSPAN="2">
ROWSPAN	Allows a number of rows to be combined in a cell	<TD ROWSPAN="2">

➤ **FORMS:**

- A form is a web page which allows the user to enter information; it also allows the user to interact with the contents of the form.
- To insert a form we use the <FORM> </FORM> tags.
- The rest of the form elements such as text boxes, check boxes, and pull down menus and so on must be inserted between the form tags.
- The form container works as follows:

<FORM METHOD="how to send" ACTION="URL of script">

.....form data.....

</FORM>

- The <FORM> tag takes two attribute:
 - **METHOD:** This attribute accepts GET or POST as its value. POST is by far the more popular, as it allows for a greater amount of data to be sent. GET is a little easier for web programmer to deal with, and is best with single response, like a single textbox.
 - **ACTION:** It simply accepts the URL for the script that will process the data from your form.

✓ **Difference between GET and POST Methods of Form:**

- The POST method is used to send sensitive information's such as password, credit card number

etc. The GET method appends data along with the URL. It is less secure than POST method.

➤ <FORM> Elements:

- Form elements have properties such as Text boxes, Password boxes, Check boxes, Radio buttons, Submit, Reset etc.
- The properties are specified in the TYPE attribute of the HTML element <INPUT> </INPUT>
- <INPUT> Element's Properties

Element	Description
TYPE =	This value indicates the type of INPUT entry filed which can include text, password and so on.
NAME =	It represents a variable name passed to CGI application.
VALUE =	The data associated with the variable name to be passed to the CGI application.
CHECKED =	This value indicates the Button/box is checked by default.
SIZE =	This value indicates the number of characters in the text field.
MAXLENGTH =	This value indicates the maximum number of characters that can be accepted.

➤ INPUT TYPES:

1. TEXT BOXES:

- These boxes are used to provide input fields for text, phone, numbers, and dates and so on.
- Example: <INPUT TYPE="TEXT" NAME = " studentname" SIZE="30">

2. PASSWORD:

- These boxes are used to allow the entry of passwords.
- Example: <INPUT TYPE="PASSWORD" NAME="Secret">

3. CHECKBOX:

- These boxes are used to allow the user to select more than one option.
- Example: <INPUT TYPE="CHECKBOX">

4. RADIO Button:

- The Radio button allows user to select only one option among a list of options.
- Example: <INPUT TYPE="RADIO">

5. File Upload:

- You can use a file upload to allow users to upload files to your web server.
- Example: <INPUT TYPE="FILE" NAME="fileupload">

6. SUBMIT:

- This is the element that causes the browser to send the names and values of the other elements to the CGI application specified by the ACTION attribute of the FORM element.
- Example: `<INPUT TYPE="SUBMIT" VALUE="Submit the Form">`

7. RESET:

- It allows the user to clear all the input in the form.
- Example: `<INPUT TYPE="RESET" VALUE="Reset the Form">`

✓ Other Elements used in FORMS:

- **Text Area:** This element is enclosed by the tags `<TEXTAREA>` `</TEXTAREA>` it is an element that allows for free form text entry.

Example: `<TEXTAREA NAME="Remarks ROWS="3" COLS="50">`

Please give your opinion here.

`</TEXTAREA>`

- **<SELECT> tag:** The `<SELECT>` tag is used to create a drop-down list. The `<OPTION>` tag inside the `<SELECT>` tag defines the available options in the list.

➤ FRAMES:

- HTML Frames are used to divide your browser window into multiple sections where each section can load a separate HTML document.
- A collection of frames in the browser window is known as Frameset.

✓ Creating Frames:

- To use frames on a page we use `<FRAMESET>` tag instead of `<BODY>` tag.
- The `<FRAMESET>` tag defines how to divide the window into frames.
- The row attribute of `<FRAMESET>` tag defines horizontal frames.
- The cols attribute of `<FRAMESET>` tag defines vertical frames.
- Each Frame is indicated by `<FRAME>` tag.

• Example:

`<HTML>`

`<HEAD>`

`<TITLE> HTML FRAMES </TITLE>`

`</HEAD>`

`<FRAMESET ROWS="10%, 80%, 10%">`

`<FRAME name="top" src="/html top_frame.htm" />`

```

<FRAME name= "main" src="/html main_frame.htm" />
<FRAME name= "bottom" src="/html bottom_frame.htm" />
<NOFRAMES>
<BODY>
  </NOFRAMES>
  </FRAMESET>
</HTML>

```

✓ **Advantages of HTML:**

- HTML document browser interfaces are easy to build.
- It is easy to learn.
- There are some specialized structures in HTML.

✓ **Disadvantages of HTML:**

- It is a weak presentation tool.
- Weak markup tool.
- It is very instable.

➤ **XML:**

- XML stands for **eXtensible Markup Language**.
- XML is a markup language for documents containing structured information.
- Structured information contains both content (words, pictures etc.) and some indication of what role content plays.
- XML is a text-based markup language that is fast becoming the standard for data interchange on the web.

➤ **Difference between HTML and XML:**

HTML	XML
<ul style="list-style-type: none"> • Hypertext Markup Language 	<ul style="list-style-type: none"> • eXtensible Markup Language
<ul style="list-style-type: none"> • HTML is used to display data and to focus on formatting of data. 	<ul style="list-style-type: none"> • XML is used to describe data and focus on what data is.
<ul style="list-style-type: none"> • HTML tags are predefined 	<ul style="list-style-type: none"> • XML tags are not predefined (Create our own tag)
<ul style="list-style-type: none"> • HTML tags are not case sensitive 	<ul style="list-style-type: none"> • XML tags are case sensitive
<ul style="list-style-type: none"> • HTML is not extensible 	<ul style="list-style-type: none"> • XML is extensible

➤ DHTML:

- DHTML stands for **Dynamic Hyper Text Markup Language**.
- DHTML refers to Web content that changes each time it is viewed.
- For example, the same URL could result in a different page depending on any number of parameters, such as:
 - geographic location of the user
 - Time of day
 - Previous pages viewed by the user
 - Profile of the reader
- Dynamic HTML is collective term for a combination of HTML tags and options that can make web pages more animated and interactive than previous versions of HTML.

➤ Web Hosting:

- Web Hosting means of hosting web-server application on a computer system through which electronic content on internet is readily available to any web-browser client.
- Various types of web hosting services are:
 - Free Hosting
 - Virtual or shared Hosting
 - Dedicated Hosting
 - Co-location Hosting

✓ Free Hosting:

- This type of hosting is available with many prominent sites that offer to host some web pages for no cost.

✓ Virtual or shared Hosting:

- This type of hosting is where one's own web site domain (ex: www.yourname.com) is hosted on the web server of hosting company along with the other web sites.

✓ Dedicated Hosting:

- In this type of hosting, the company wishing to go online rents an entire web server from hosting company. This is suitable for large, high traffic sites.

✓ Co-location Hosting:

- In this type of hosting, the company owning the site instead of web hosting company. Suitable for those who need the ability to make changes?

➤ Web Scripting :

- *The process of creating and embedding scripts in a web page is known as web-scripting.*
- **Script:**
 - *A Script is a list of commands embedded in a web page.*
 - Scripts are interpreted and executed by a certain program or scripting –engine.
- **Types of Scripts:**
 1. **Client Side Script:**
 - Client side scripting enables interaction within a web page.
 - The client-side scripts are downloaded at the client-end and then interpreted and executed by the browser.
 - Some popular client-side scripting languages are VBScript, JavaScript, ActionScript.
 2. **Server-Side Scripts:**
 - Server-side scripting enables the completion or carrying out a task at the server-end and then sending the result to the client –end.
 - Some popular server-side Scripting Languages are PHP, Perl, ASP(Active Server Pages), JSP(Java Server Pages) etc.

CHAPTER 17 – WEB DESIGNING BLUE PRINT

VSA (1 marks)	SA (2 marks)	LA (3 Marks)	Essay (5 Marks)	Total
01 Question	-	01 Question	-	02 Questions
Question No 10	-	Question No 26	-	04 Marks

Important Questions

➤ 1 Marks Question:

1. Expand the following:

a. HTML	b. XML	c. DHTML	
---------	--------	----------	--

[June 2016]
2. What are HTML, XML, and DHTML?
3. Mention the use of HTML. [March 2015]
4. Write any one HTML tag. [June 2015]
5. What is DHTML? [March 2016]
6. What will be the extension of HTML language file?
7. What is the use of web page?
8. What is Script?

➤ 3 Marks Question:

1. Explain any three text formatting tags in HTML. [March 2015, March 2017]
2. Explain Anchor tag with syntax and example.
3. Explain different INPUT in HTML.
4. What is web hosting? Mention different types of web hosting. [June 2015, March 2016]
5. What is web scripting? Explain the different types of web scripting.
6. Write the difference between client side scripting and server side scripting.

ONE MARKS QUESTION AND ANSWERS

➤ Question No 1 – Typical Configuration of Computer System:

1. What is a motherboard?
 - The motherboard is the most important circuit board of the computer; all the components used in a computer are connected to the motherboard.
2. What is Microprocessor?
 - The CPU is fabricated as a single Integrated Circuit (IC) and is also known as Microprocessor. The microprocessor is plugged into the motherboard of the computer.
3. What is the purpose of registers in the CPU?
 - Registers in CPU are used to store temporary data and address which are required for processing.
4. How does the computer communicate with other devices?
 - The computer communicates with other devices with the help of buses.
5. What is system bus?
 - A data bus, control bus, and the address bus collectively called as the system bus.
6. What is the function of control bus?
 - The control bus carries signal, which are used to control the access and use of data and addresses.
7. What is a data bus?
 - Data bus provides a path to transfer data between CPU and memory.
8. What is a port?
 - A port is a socket on the computer used to connect external device to the computer.
9. What is an interface?
 - An interface is a hardware used to connect external device to the computer.
10. Expand PCI.
 - PCI stands for “**P**eripheral **C**omponent **I**nterconnect”.
11. How many bits of data are sent in a serial port?
 - The serial port can transfer one bit at a time.
12. Expand USB.
 - USB stands for “**U**niversal **s**erial **B**us”.
13. Give one feature of USB port.
 - With USB, a new device can be added to the computer without adding an adapter card or even turning the computer off.

14. What is meant by plug and play device?

- USB is a plug and play interface between computer and add-on device i.e. a new device can be added to the computer without adding an adapter card or even turning the computer off.

15. Name any one USB device.

- Pen Drive is an USB device.

16. Is device controller a hardware or software?

- Device controller is hardware.

17. What is cache memory?

- The cache memory is a very high speed memory placed between RAM and CPU.

18. Where is L1 cache located?

- L1 cache is located inside the CPU.

19. Where is L2 cache located?

- L2 cache is located outside the CPU on the motherboard. (or between RAM and CPU)

20. Expand SDRAM.

- Synchronous Dynamic Random Access Memory.

21. Give the expansion of DDRAM.

- Double Data Rate Random Access Memory.

22. Expand SMPS.

- SMPS stands for “**Switch Mode Power Supply**”.

23. What is the use of SMPS?

- In a PC the power is supplied to the different parts of computer is by SMPS which converts 230 Volts of AC to 5 to 12 DC Volts.

24. What is approximate power consumed by a PC?

- The approximate power consumed by a PC is about 400 watts.

25. Expand UPS.

- UPS stands for “**Uninterruptible Power Supply**”.

26. What is the use of UPS?

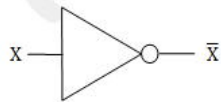
- UPS keeps a computer running for several minutes to hours after a power failure.

27. List the types of UPS.

- UPS is of two types Offline UPS and Online UPS.

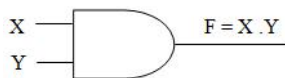
➤ **Question No 2 – Logic Gates:**

1. What is a logic gate?
 - A Gate is a simply an electronic circuit which operates on one or more input signals and always produces an output signal.
2. Mention the three basic logic gates.
 - The three basic logic gates are NOT, AND, OR
3. Which basic gate is named as Inverter?
 - NOT gate is named as Inverter.
4. What is truth table?
 - A table which represents all possible values of logical variables along with the all possible results of the given combination of values.
5. Write the logic circuit for NOT gate.



X	\bar{X}
0	1
1	0

6. Write the truth table for NOT gate.
7. Write the logic circuit for AND gate.



X	Y	F = X.Y
0	0	0
0	1	0
1	0	0
1	1	1

8. Write the truth table for AND gate.
9. Write the logic circuit for OR gate.



X	Y	F = X+Y
0	0	0
0	1	1
1	0	1
1	1	1

10. Write the truth table for OR gate.
11. What is meant by universal gates?
 - Universal gate is a gate using which all the basic gates can be designed.
12. Mention different universal gates.
 - NAND and NOR are the universal gates.
13. What is the output of the two input NAND gate for the input A=0, B=1?
 - When A=0, B=1 then the output of NAND gate is 1.
 - A NAND B = $\overline{0 \cdot 1} = \overline{0} = 1$
14. What are the values of inputs to a three input NAND gate, if its output is 1?
 - NAND gate produces 1 if any input represents a 0.

15. What are the values of inputs to a three input NAND gate, if its output is 0?
- NAND gate produces 0 if any input represents a 1.
16. What is the output of the two input OR gate for the input A=0, B=0?
- When A=0, B=0, then the output of OR gate is 0.
 - $A \text{ OR } B = A + B = 0 + 0 = 0$
17. What are the values of inputs to a three input OR gate, if its output is 0?
- OR gate produces 0 if any input represents a 0.
18. What are the values of inputs to a three input OR gate, if its output is 1?
- OR gate produces 1 if any input represents a 1.

➤ **Question No 3 – Data Structures:**

1. What is Data structure?
 - Data Structure is the way of collecting and organizing the data in such a way that we can perform operation on these data in an effective way.
2. What are primitive data structures?
 - Data structures that are directly operated upon the machine-level instructions are known as primitive data structures.
3. Give any two examples for primitive data structures.
 - The integers, float, character data, pointers are primitive data structures.
4. What are non-primitive data structures?
 - The Data structures that are derived from the primitive data structures are called Non-primitive data structure.
5. Mention any two examples for non- primitive data structures?
 - Array, stack, queues, linked list, tree and graph are non- primitive data structures
6. What are lists?
 - Lists are linear collection of data items.
7. What is meant by linear data structures?
 - Linear Data structures are kind of data structure that has homogeneous elements.
8. What are non-linear data structures?
 - A Non-Linear Data structures is a data structure in which data item is connected to several other data items.
9. Define an array.
 - An array is an ordered collection of elements of same data type that share common name.

10. Differentiate between one-dimensional and two-dimensional array.

- In one-dimensional array we use only one subscript to identify an element where as in two dimensional arrays we use two subscripts.

11. What do you mean by traversal operation?

- The processing of accessing each element exactly once to perform some operation is called traversing.

12. Define searching.

- The process of finding the location of a data element in the given collection of data elements is called as searching.

13. Mention the types of searching in the array.

- Linear Search and Binary Search.

14. Define sorting.

- The process of arrangement of data elements in ascending or descending order is called sorting.

15. What is a stack?

- A stack is an ordered collection of items in which an element may be inserted or deleted only at same end.

16. Name the data structure which is called LIFO.

- A LIFO (Last In First Out) data structure is also called as stack.

17. What is LIFO list?

- A stack data structure which follows LIFO (Last In First Out) list principle.

18. What are the operations that can be performed on stacks?

- Stack(), push(item), pop(), peek(), isEmpty(), size() are the operations of stack.

19. Define the term PUSH and POP operation in stack.

- The process of adding one element or item to the stack is represented by an operation called as the PUSH operation.
- The process of deleting one element or item from the stack is represented by an operation called as the POP operation.

20. Mention any one application of stacks.

- It is used to reverse a word.
- “Undo” mechanism in text editor.
- Polish Notation

21. What is a queue?

- A queue is an ordered collection of items where an item is inserted at one end called the “rear”

and an existing item is removed at the other end, called the “front”.

22. What is the other name of queue?

- Other name of queue is FIFO.

23. What is FIFO list?

- In a FIFO list, the first element added to the list will be the first one to be removed.

24. Mention the different types of queues.

- Different types of queues are Linear queue, Circular queue, Double ended queue, Priority queue.

25. What are the operations that can be performed on queue?

- Queue(), enqueue(item), dequeue(), isEmpty(), size().

26. What is a linked list?

- A linked list is a linear collection of data elements called nodes.

27. Which data structure creates relationship between data elements through links?

- Linked List creates relationship between data elements through links.

28. What is binary tree?

- A binary tree is an ordered tree in which each internal node can have maximum of two child nodes connected to it.

29. What do you mean by depth of a tree?

- The height or depth of a tree is defined to be the maximum number of nodes in a branch of tree.

30. How do you find the degree of tree?

- The degree of tree is the maximum degree of nodes in the tree. The degree of node is the maximum number of children that can exist for a node.

➤ Question No 4 – Classes and Objects:

1. What is class?

- A class is a collection of objects that have identical properties, common behavior and shared relationship.

2. What are the two types of members referenced in a class?

- The two types of members referred in a class are data members and member functions.

3. What are data members?

- The variables declared inside a class are known as data members.

4. What is a member function?

- The functions declared inside a class are known as member functions.

5. Mention the access specifiers used with a class.
 - Different access specifiers such as private, public, and protected.
6. Is it possible to access data outside a class?
 - Yes, public data members can be accessed outside a class.
7. Which type data members are accessible outside a class?
 - Public members can be accessed outside a class.
8. Which access specifier is implicitly used in a class?
 - Private is the default access specifiers of a class.
9. Define the term public access.
 - Public access means that member can be accessed any function inside or outside the class.
10. What is the significance of scope resolution operator (::)?
 - Scope resolution operator (::) is used to define the member function outside the class.
11. What is an object?
 - An object is an instance of a class. Objects are sometimes called as instance variables.
12. How are objects of a class declared? Give example.
 - Objects can be created with the following declaration.

```
Class_Name ObjectName1, ObjectName2...;
```
 - Example: Date Today, DOB;
13. Mention the operator used to access member of a class.
 - . (Dot) operator is used to access member of a class.
14. What is meant by array of objects?
 - An array having class type elements is known as array of objects.
15. Write an example to show how objects can be used as function arguments.
 - Date D1, D2;
 - D2.Total (D1);

➤ Question No 5 – Pointers:

1. What do you mean by pointer?
 - A pointer is a variable that holds a memory address of another variable.
2. Mention any one advantage of pointer?
 - Pointers save memory space.
 - Dynamically allocate and de-allocate memory.
3. What is address operator?
 - The '&' is the address operator and it represents address of the variable.

4. What is pointer operator?
 - The '*' is a pointer operator, which is also called indirection operator or dereference operator.
5. How to declare pointer?
 - The general syntax of pointer declaration is given below.
 - Syntax: Data_Type *Ptr_Variablename;
 - Example: int *iptr;
6. How to initialize pointer?
 - By using the & operator as follows:
 - int *p, q=10;
 - p = &q;
7. What is static memory?
 - Static memory allocation refers to the process of allocating memory during the compilation of the program i.e. before the program is executed.
8. What is dynamic memory?
 - Dynamic memory allocation refers to the process of allocating memory during the execution of the program or at run time.
9. What is free store?
 - Free store is a pool of memory available to allocated and de-allocated storage for the objects during the execution of the memory.
10. What is new operator in C++?
 - The new operator is used to create a heap memory space for variables.
11. What is delete operator in C++?
 - The delete operator is used to destroy the variable space which has been created by using the new operator dynamically.

➤ **Question No 6 – Database Concepts:**

1. What is data?
 - Data is a collection of facts, numbers, letters or symbols that the computer process into meaningful information.
2. What is Information?
 - Information is processed data, stored, or transmitted by a computer.
3. What is Database?
 - A Database is a collection of logically related data organized in a way that data can be easily accessed, managed and updated.

4. What is a field?
 - Each column is identified by a distinct header called attribute or field.
5. What is a record?
 - A single entry in a table is called a record or row. A record in a table represents set of related data.
 - Records are also called the tuple.
6. What is an entity?
 - An **Entity** can be any object, place, person or class.
 - In E-R Diagram, an **entity** is represented using rectangles.
7. What is an instance?
 - The collection of information stored in the database at a particular moment is called an **instance of the database**.
8. What is an attribute?
 - It is defined as a named column of a relation.
 - Ex: In STUDENT table, Regno, Name, Age, Class, Combination and Marks.
9. What is domain?
 - It is defined as a set of allowed values for one or more attributes.
10. What is a relation?
 - A relation is defined as a table with columns and rows. Data can be stored in the form of a two-dimensional table.
11. What is a table?
 - A table is a collection of data elements organized in terms of rows and columns. Table is the simplest form of data storage.
12. What is normalization?
 - Normalization is a step by step process of removing the different kinds of redundancy and anomaly one step at a time from the database.
13. What is a key?
 - It is a column or columns which identifies the each row or tuple.
14. Give the symbol notation for project and select?
 - SELECT – sigma (σ)
 - PROJECT – Pi (π)
15. What is data mining?
 - Data mining is concerned with the analysis and picking out relevant information.

➤ Question No 7 and 8 – Networking Concepts:

1. What is networking?
 - A computer network is an interconnection of two or more computers that are able to exchange information's.
2. What is client?
 - The term **nodes or workstation or client** refer to the computers those are attached to a network and are seeking to share the resources of the network.
3. What is server?
 - A Server is also a computer that facilitates the sharing of data, software, and hardware resources like printers, modems etc on the network.
 - Servers can be of two types: Non-dedicated servers, Dedicated servers
4. What is network topology?
 - Network Topology refers to the arrangement of computers and other devices in a network.
5. Expand 2G.
 - 2G stands for Second Generation.
6. What is virus?
 - VIRUS – “Vital Information Resource Under Siege”.
 - A computer virus is a computer program that can replicate itself and spread from one computer to another.
7. What is chatting?
 - On the internet, chatting is talking to other people who are using the internet at the same time you are.
8. What is cyber law?
 - It is a generic term, which refers to all the legal and regulatory aspects of internet and the World Wide Web.
9. What are cookies?
 - Cookies are messages that a web server transmits to a web browser so that the web server can keep track of the user's activity on a specific web site. Cookies have few parameters name, value, expiration date.
10. What are hackers?
 - Hackers are more interested in gaining knowledge about computer systems and possibly using this knowledge for playful pranks.

➤ Question No 9 – Internet and Open Source Concepts:

1. What is open source software?
 - The term “open source” software is used to refer to those categories of software/ programs whose licenses do not impose much condition.
2. What is free software?
 - Free software means the software is freely accessible and can be freely used, changed, improved, copied and distributed by all who wish to do so.
 - No payments are needed to be made for free software.
3. What is OSS and FLOSS?
 - OSS refers to open source software, which refers to software whose source code is available to customers and it can be modified and redistributed without any limitation.
 - FLOSS refers to Free Libre and open Source Software or to Free Livre and Open Source Software.
4. What is proprietary software?
 - Proprietary software or closed source software is the software that is neither open nor freely available.
5. What is Freeware?
 - The term freeware has no clear definition, but is generally used for software, which is available free of cost and which allows copying and further distribution, but not modification and whose source code is not available.
6. What are Browsers?
 - A Web Browser is software application that enables the user to view web pages, navigate web sites and move from one website to another.
 - Some of the web browsers are Google Chrome, Internet Explorer, Netscape Navigator, Mozilla Firefox, and Opera.
7. What is URL?
 - A URL (Uniform Resources Locator) specifies the distinct address for each resource on the internet
 - URLs look like this: type://address/path
8. What are Telnet?
 - Telnet (Teletype network) is an order Internet utility that lets you log on to remote computer systems.
 - Telnet program gives you a character-based terminal window on another system.

9. What is domain?
- An internet address which is a character based is called a Domain name.
10. What is domain affiliation?
- .com, .org, .in, .au are the domain affiliation
11. Define E-Commerce.
- E-Commerce is the trade of goods and services with the help of telecommunications and computers.
12. Expand IPR.
- IPR stands for Intellectual Property Rights.

➤ **Question No 10 – Internet and Open Source Concepts:**

1. What is HTML?
- HTML stands for Hypertext Markup Language, and it is the most widely used Language to write Web Pages.
2. What will be the extension of HTML file?
- .htm or .html is the extension of HTML file.
3. What is the use of a web page?
- Web page is a resource of information about a company/organization which is accessible through web browser.
4. What is XML?
- XML stands for **eXtensible Markup Language**.
 - XML is a markup language for documents containing structured information.
5. What is DHTML?
- DHTML stands for **Dynamic Hyper Text Markup Language**.
 - DHTML refers to Web content that changes each time it is viewed.
6. What do you mean by web hosting?
- Web Hosting means of hosting web-server application on a computer system through which electronic content on internet is readily available to any web-browser client.
7. What is web scripting?
- The process of creating and embedding scripts in a web page is known as web-scripting.
8. What is script?
- A Script is a list of commands embedded in a web page.

A
PRACTICAL RECORD
BOOK
OF
COMPUTER SCIENCE (PCMC)
SECOND PUC



MORARJI DESAI RESIDENTIAL PU
SCIENCE COLLEGE, DUDDA, HASSAN

MORARJI DESAI RESIDENTIAL PU SCIENCE COLLEGE, DUDDA, HASSAN



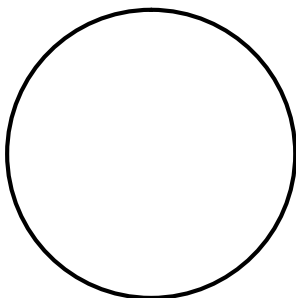
Laboratory Certificate

This is to certify that Mr. / Mrs.
has satisfactorily completed the course of experiments in practical
COMPUTER SCIENCE prescribed by the **Pre-University,
Bangalore** for **SECOND PUC (P.C.M.C)** course in the laboratory of
this college in the year 2018-19.

Signature of the Lecturer

Head of the Department

Date:



Name of the Candidate :

Register Number :

Examination Centre :

Date of Practical Examination:

CONTENTS

Program No	Program Name	Page No
SECTION- A (C++ AND DATA STRUCTURES)		
01	Write a C++ program to find the frequency presence of an element in an array.	01
02	Write a C++ program to insert an element into an array at a given position.	03
03	Write a C++ program to delete an element from an array from a given position.	05
04	Write a C++ program to sort the element of an array in ascending order using insertion sort.	07
05	Write a C++ program to search for a given element in an array using binary search method.	09
06	Write a C++ program to create a class with data members principal, time and rate. Create a member function to accept data values, to compute simple interest and to display the result.	11
07	Write a C++ program to create a class with data members a, b, c and member functions to input data, compute the discriminates based on the following conditions and print the roots. <ul style="list-style-type: none"> ➤ If discriminates = 0, print the roots are equal and their value. ➤ If discriminates > 0, print the real roots and their values. ➤ If discriminates < 0, print the roots are imaginary and exit the program. 	13
08	Write a C++ program to find the area of square/ rectangle/ triangle using function overloading.	15
09	Write a C++ program to find cube of a number using inline function.	17
10	Write a C++ program to find sum of the series $1 + x + x^2 + x^3 + \dots x^n$ using constructors.	18
11	Create a base class containing the data member roll number and name. Also create a member function to read and display the data using the concept of single level inheritance. Create a derived class that contains marks of two subjects and total marks as the data members.	20
12	Create a class containing the following data members Register No, Name and Fees. Also create a member function to read and display the data using the concept of pointers to objects.	22
13	Write a C++ program to perform push items into the stack.	24
14	Write a C++ program to perform pop items into the stack.	26
15	Write a C++ program to perform Enqueue and Dequeue.	29

Program No	Program Name	Page No																		
SECTION – B (SQL)																				
01	<p>Generate the electricity bill for one customer Create a table for house hold Electricity bill with the following fields.</p> <table border="1" data-bbox="572 383 1075 566"> <thead> <tr> <th>Field Name</th> <th>Type</th> </tr> </thead> <tbody> <tr> <td>RR_NO</td> <td>VARCHAR2(10)</td> </tr> <tr> <td>CUS_NAME</td> <td>VARCHAR2(15)</td> </tr> <tr> <td>BILLING_DATE</td> <td>DATE</td> </tr> <tr> <td>UNITS</td> <td>NUMBER(4)</td> </tr> </tbody> </table> <p>Insert 10 records into the table.</p> <ol style="list-style-type: none"> Check the structure of table and note your observation. Add two fields to the table. <ol style="list-style-type: none"> BILL_AMT NUMBER(6,2) DUE_DATE DATE Compute the bill amount for each customer as per the following rules. <ol style="list-style-type: none"> MIN_AMT Rs. 50 First 100 units Rs 4.50/Unit >100 units Rs. 5.50/Unit Compute due date as BILLING_DATE + 15 Days List all the bills generated. 	Field Name	Type	RR_NO	VARCHAR2(10)	CUS_NAME	VARCHAR2(15)	BILLING_DATE	DATE	UNITS	NUMBER(4)	32								
Field Name	Type																			
RR_NO	VARCHAR2(10)																			
CUS_NAME	VARCHAR2(15)																			
BILLING_DATE	DATE																			
UNITS	NUMBER(4)																			
02	<p>Create a student database and compute the results. Create a table for class of students with the following fields.</p> <table border="1" data-bbox="572 1115 1075 1444"> <thead> <tr> <th>Field Name</th> <th>Type</th> </tr> </thead> <tbody> <tr> <td>ID_NO</td> <td>NUMBER(4)</td> </tr> <tr> <td>S_NAME</td> <td>VARCHAR2(15)</td> </tr> <tr> <td>SUB1</td> <td>NUMBER(3)</td> </tr> <tr> <td>SUB2</td> <td>NUMBER(3)</td> </tr> <tr> <td>SUB3</td> <td>NUMBER(3)</td> </tr> <tr> <td>SUB4</td> <td>NUMBER(3)</td> </tr> <tr> <td>SUB5</td> <td>NUMBER(3)</td> </tr> <tr> <td>SUB6</td> <td>NUMBER(3)</td> </tr> </tbody> </table> <ol style="list-style-type: none"> Add records into the table for 10 students for Student ID, Student Name and marks in 6 subjects using INSERT command. Display the description of the fields in the table using DESC command. Alter the table and calculate TOTAL and PERC_MARKS. Compute the RESULT as “PASS” or “FAIL” by checking if the student has scored more than 35 marks in each subject. List the contents of the table. Retrieve all the records of the table. Retrieve only ID_NO and S_NAME of all the students. List the students who have result as “PASS”. List the students who have result as “FAIL”. Count the number of students who have passed. Count the number of students who have failed. List the students who have percentage greater than 60. Sort the table according to the order of ID_NO. 	Field Name	Type	ID_NO	NUMBER(4)	S_NAME	VARCHAR2(15)	SUB1	NUMBER(3)	SUB2	NUMBER(3)	SUB3	NUMBER(3)	SUB4	NUMBER(3)	SUB5	NUMBER(3)	SUB6	NUMBER(3)	35
Field Name	Type																			
ID_NO	NUMBER(4)																			
S_NAME	VARCHAR2(15)																			
SUB1	NUMBER(3)																			
SUB2	NUMBER(3)																			
SUB3	NUMBER(3)																			
SUB4	NUMBER(3)																			
SUB5	NUMBER(3)																			
SUB6	NUMBER(3)																			

03	<p>Generate the Employee details and compute the salary based on the department. Create the following table EMPLOYEE</p> <table border="1" data-bbox="572 181 1075 365"> <thead> <tr> <th>Field Name</th> <th>Type</th> </tr> </thead> <tbody> <tr> <td>EMP_ID</td> <td>NUMBER(4)</td> </tr> <tr> <td>DEPT_ID</td> <td>NUMBER(2)</td> </tr> <tr> <td>EMP_NAME</td> <td>VARCHAR2(15)</td> </tr> <tr> <td>EMP_SALARY</td> <td>NUMBER(5)</td> </tr> </tbody> </table> <p>Create another table DEPARTMENT</p> <table border="1" data-bbox="572 450 1075 598"> <thead> <tr> <th>Field Name</th> <th>Type</th> </tr> </thead> <tbody> <tr> <td>DEPT_ID</td> <td>NUMBER(2)</td> </tr> <tr> <td>DEPT_NAME</td> <td>VARCHAR2(20)</td> </tr> <tr> <td>SUPERVISOR</td> <td>VARCHAR2(20)</td> </tr> </tbody> </table> <p>Assume the DEPARTMENT names as Purchase (Id-01), Accounts (Id-02), Sales (Id-03), and Apprentice (Id-04)</p> <p>Enter 10 rows of data for table EMPLOYEE and 4 rows of data for DEPARTMENT table.</p> <p>Write the SQL statements for the following:</p> <ol style="list-style-type: none"> 1. Find the names of all employees who work for the Accounts department. 2. How many employees work for Accounts department? 3. What are the Minimum, Maximum and Average salary of employees working for Accounts department? 4. List the employees working for particular supervisor. 5. Retrieve the department names for each department where only one employee works. 6. Increase the salary of all employees in the sales department by 15%. 7. Add a new Column to the table EMPLOYEE called BONUS NUMBER (5) and compute 5% of the salary to the said field. 8. Delete all the rows for the employee in the Apprentice department. 	Field Name	Type	EMP_ID	NUMBER(4)	DEPT_ID	NUMBER(2)	EMP_NAME	VARCHAR2(15)	EMP_SALARY	NUMBER(5)	Field Name	Type	DEPT_ID	NUMBER(2)	DEPT_NAME	VARCHAR2(20)	SUPERVISOR	VARCHAR2(20)	39
Field Name	Type																			
EMP_ID	NUMBER(4)																			
DEPT_ID	NUMBER(2)																			
EMP_NAME	VARCHAR2(15)																			
EMP_SALARY	NUMBER(5)																			
Field Name	Type																			
DEPT_ID	NUMBER(2)																			
DEPT_NAME	VARCHAR2(20)																			
SUPERVISOR	VARCHAR2(20)																			
SECTION- C HTML																				
01	Write a HTML program to create a study time-table	43																		
02	Create an HTML program with table and Form.	46																		

SECTION - A

C++ AND DATA STRUCTURES

PROGRAM 1:

Write a C++ program to find the frequency presence of an element in an array.

```

#include<iostream.h>
#include<conio.h>
class Frequency
{
private:
    int a[10], n, ele, count;           //Data member
public:
    void readdata( );
    void findfreq( );                 //Member functions declaration
    void display( );
};

void Frequency::readdata( )           //Member function definition
{
    cout<<"Enter the size of the array:"<<endl;
    cin>>n;
    cout<<"Enter the array elements:"<<endl;
    for(int i=0; i<n; i++)
        cin>>a[i];
    cout<<"Enter the element to find the frequency"<<endl;
    cin>>ele;
}

void Frequency::findfreq( )           //Member function definition
{
    count=0;
    for(int i=0; i<n; i++)
        if(ele == a[i])               //Traversing Operation
            count++;
}

void Frequency::display( )           //Member function definition
{
    if(count > 0)
        cout<<ele<<" Occurs"<<count<<" Time";
    else
        cout<<ele<<" Does Not Exists";
}

void main( )                           //Main function
{
    Frequency f;

```



```
clrscr();  
f.readdata();  
f.findfreq();  
f.display();  
getch();  
}
```

OUTPUT 1:

```
Enter the size of the array:  
5  
Enter the array elements  
2    6    2    8    1  
Enter the element to find the frequency:  
2  
2 Occurs 2 Time
```

OUTPUT 2:

```
Enter the size of the array:  
5  
Enter the array elements  
2    7    5    12    9  
Enter the element to find the frequency:  
4  
4 Does Not Exist
```



PROGRAM 2:

Write a C++ program to insert an element into an array at a given position.

```

#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
class Insertion
{
    private:
        int a[10], n, pos, ele, i;           //Data Members
    public:
        void readdata( );
        void insert( );                     // Member Function Declaration
        void display( );
};

void Insertion::readdata( )
{
    cout<<"Enter the size of the array"<<endl;
    cin>>n;
    cout<<"Enter the elements for the array"<<endl;
    for(i=0; i<n; i++)
        cin>>a[i];
    cout<<"Enter the position of the element in the array"<<endl;
    cin>>pos;
    cout<<"Enter the element to be inserted"<<endl;
    cin>>ele;
}

void Insertion::insert( )
{
    if(pos>n)
    {
        cout<<"Out of array limits!!!";
        getch( );
        exit(0);
    }
    else
    {
        for(i=n; i>=pos; i--)
            a[i+1] = a[i];                 // Shift array elements
        a[pos] = ele;                       // Insert the given element
        n = n+1;                             // Size of the array is incremented by 1
    }
}

```



```
    }  
}  
  
void Insertion::display( )  
{  
    cout<<"Array elements after insertion are:"<<endl;  
    for(i=0; i<n; i++)  
        cout<<a[i]<<"\t";  
}  
  
void main( )  
{  
    Insertion i;  
    clrscr();  
    i.readdata();  
    i.insert();  
    i.display();  
    getch();  
}
```

OUTPUT 1:

```
Enter the size of the array  
5  
Enter the elements for the array  
5    9    14    16    23  
Enter the position of the element in the array  
4  
Enter the element to be inserted  
35  
Array elements after insertion are:  
5    9    14    16    35    23
```

OUTPUT 2:

```
Enter the size of the array  
3  
Enter the elements for the array  
12    8    20  
Enter the position of the element in the array  
4  
Enter the element to be inserted  
15  
Out of array limits!!!
```



PROGRAM 3:

Write a C++ program to delete an element from an array from a given position.

```
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
class Deletion
{
    private:
        int a[10], n, pos, i;
    public:
        void readdata( );
        void delet( );
        void display( );
};

void Deletion::readdata( )
{
    cout<<"Enter the size of the array"<<endl;
    cin>>n;
    cout<<"Enter the elements for the array:"<<endl;
    for (i=0; i<n; i++)
        cin>>a[i];
    cout<<"Enter the position to an delete an element:\n";
    cin>>pos;
}

void Deletion::delet( )
{
    if(pos>n)
    {
        cout<<"Out of array limits...!!!";
        getch( );
        exit(0);
    }
    else
    {
        for(i=pos; i<n; i++)
            a[i] = a[i+1];                // Move higher position element
        n = n-1;                          // Reduce size of the array by 1
    }
}
```



```
void Deletion::display( )
{
    cout<<"After deletion the array elements are"<<endl;
    for(i=0; i<n; i++)
        cout<<a[i]<<"\t";
}

void main( )
{
    Deletion d;
    clrscr();
    d.readdata( );
    d.delet( );
    d.display( );
    getch( );
}
```

OUTPUT 1:

```
Enter the size of the array
5
Enter the elements for the array:
4      9      14      28      34
Enter the position to an delete an element:
3
After deletion the array elements are
4      9      14      34
```

OUTPUT 2:

```
Enter the size of the array
3
Enter the elements for the array:
9      4      17
Enter the position to an delete an element:
4
Out of array limits...!!!
```



PROGRAM 4:

Write a C++ program to sort the element of an array in ascending order using insertion sort.

```
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>

class Sort
{
    private:
        int a[10], n, i;
    public:
        void readdata( );
        void insertionsort( );
        void display( );
};

void Sort::readdata( )
{
    cout<<"Enter the size of the array:"<<endl;
    cin>>n;
    cout<<"Enter the elements for the array:"<<endl;
    for(i=0; i<n; i++)
        cin>>a[i];
}

void Sort::insertionsort( )
{
    int j, temp;
    for(i=1; i<n; i++)
    {
        j = i;
        while(j >= 1)
        {
            if(a[j] < a[j-1])
            {
                temp = a[j];
                a[j] = a[j-1];
                a[j-1] = temp;
            }
            j = j-1;
        }
    }
}
```



```

}

void Sort::display( )
{
    for(i=0; i<n; i++)
        cout<<a[i]<<"\t";
        cout<<endl;
}

void main( )
{
    Sort s;
    clrscr();
    s.readdata( );
    cout<<"Unsorted List....."<<endl;
    cout<<"*****"<<endl;
    s.display( );
    s.insertionsort( );
    cout<<"Sorted List....."<<endl;
    cout<<"*****"<<endl;
    s.display( );
    getch( );
}

```

OUTPUT 1:

```

Enter the size of the array:
5
Enter the elements for the array:
25    40    14    8    33
Unsorted List.....
*****
25    40    14    8    33
Sorted List.....
*****
8    14    25    33    40

```

OUTPUT 2:

```

Enter the size of the array:
5
Enter the elements for the array:
-32   -4    0    -5   -23
Unsorted List.....
*****
-32   -4    0    -5   -23
Sorted List.....
*****
-32   -23   -5   -4    0

```



PROGRAM 5:

Write a C++ program to search for a given element in an array using binary search method.

```
#include<iostream.h>
#include<conio.h>

class Search
{
    private:
        int a[10], n, ele, loc, beg, end, mid, i;
    public:
        void readdata( );
        void bsearch( );
        void display( );
};

void Search::readdata( )
{
    cout<<"Enter the size of the array:"<<endl;
    cin>>n;
    cout<<"Enter the array elements in sorted order:"<<endl;
    for(i=0;i<n;i++)
        cin>>a[i];
    cout<<"Enter the element to search:"<<endl;
    cin>>ele;
}

void Search::bsearch( )
{
    loc = -1; // Assume that element does not exist
    beg = 0; // First element of the array
    end = n-1; // Second element of the array
    while(beg <= end)
    {
        mid = (beg+end)/2;
        if(ele == a[mid]) // Element found at mid
        {
            loc = mid;
            break;
        }
        else
        if(ele < a[mid])
            end = mid-1;
    }
}
```



```
        else
            beg = mid+1;
    }
}

void Search::display()
{
    if(loc == -1)
        cout<<ele<<" Element does not exist...!!!";
    else
        cout<<ele<<" Found at Location:"<<loc+1;
}

void main( )
{
    Search s;
    clrscr();
    s.readdata( );
    s.bsearch();
    s.display( );
    getch();
}
```

OUTPUT 1:

```
Enter the size of the array:
5
Enter the array elements in sorted order:
12    23    39    47    57
Enter the element to search:
39
39 Found at Location:3
```

OUTPUT 2:

```
Enter the size of the array:
4
Enter the array elements in sorted order:
5     8     14    17
Enter the element to search:
22
22 Element does not exist...!!!
```



PROGRAM 6:

Write a C++ program to create a class with data members principal, time and rate. Create a member function to accept data values, to compute simple interest and to display the result.

```
#include<iostream.h>
#include<conio.h>

class SimpleInterest
{
    private:
        float principal,rate,time,si;           //Data Members
    public:
        void readdata( );
        void compute( );                       //Member Functions Declaration
        void display( );
};

void SimpleInterest::readdata( )              //Member Function Definition
{
    cout<<"Enter the Principal, Rate and Time"<<endl;
    cin>>principal>>rate>>time;
}

void SimpleInterest::compute( )              //Member Function Definition
{
    si=(principal * time * rate)/100;
}

void SimpleInterest::display( )              //Member Function Definition
{
    cout<<"Principal = "<<principal<<endl;
    cout<<"Time = "<<time<<endl;
    cout<<"Rate = "<<rate<<endl;
    cout<<"Simple Interest = "<<si<<endl;
}

void main( )
{
    SimpleInterest si;
    clrscr( );
    si.readdata( );
    si.compute( );
    si.display( );
}
```



```
    getch();  
}
```

OUTPUT 1:

```
Enter the Principal, Rate and Time  
120000  
12.75  
6.5  
Principal = 120000  
Time = 6.5  
Rate = 12.75  
Simple Interest = 99450
```

OUTPUT 2:

```
Enter the Principal, Rate and Time  
10000  
12  
2  
Principal = 10000  
Time = 2  
Rate = 12  
Simple Interest = 2400
```



PROGRAM 7:

Write a C++ program to create a class with data members a, b, c and member functions to input data, compute the discriminant based on the following conditions and print the roots.

- If discriminant = 0, print the roots are equal and their value.
- If discriminant > 0, print the real roots and their values.
- If discriminant < 0, print the roots are imaginary and exit the program.

```
#include<iostream.h>
#include<conio.h>
#include<math.h>
class Quadratic
{
    private:
        int a, b, c;
        float disc, x, x1, x2;
    public:
        void readdata( );
        void compute( );
        void display( );
};

void Quadratic::readdata( )
{
    cout<<"Enter the values for a, b, c (Co-efficeient)"<<endl;
    cin>>a>>b>>c;
}

void Quadratic::compute( )
{
    disc = b*b-4*a*c;
}

void Quadratic::display( )
{
    compute( );
    if(disc == 0)
    {
        cout<<"Equal Roots..."<<endl;
        x=-b/(2*a);
        cout<<"Root is...."<<x;
    }
    else if(disc>0)
    {
```



```
        cout<<"Real and Distinct Roots..."<<endl;
        x1=(-b+sqrt(disc))/(2*a);
        x2=(-b-sqrt(disc))/(2*a);
        cout<<"Root 1 is "<<x1<<endl;
        cout<<"Root 2 is "<<x2<<endl;
    }
    else
        cout<<"Imaginary Roots..."<<endl;
}

void main( )
{
    Quadratic q;
    clrscr( );
    q.readdata( );
    q.display( );
    getch( );
}
```

OUTPUT 1:

```
Enter the values for a, b, c (Co-efficeient)
2      -7      6
Real and Distinct Roots...
Root 1 is 2
Root 2 is 1.5
```

OUTPUT 2:

```
Enter the values for a, b, c (Co-efficeient)
1      2      1
Equal Roots...
Root is...-1
```

OUTPUT 3:

```
Enter the values for a, b, c (Co-efficeient)
1      2      5
Imaginary Roots...
```



PROGRAM 8:

Write a C++ program to find the area of square/ rectangle/ triangle using function overloading.

```
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
#include<math.h>

class Funcoverload
{
    public:
        float area(float a)                //To compute area of square
        {
            return a*a;
        }

        float area(float l,float b)        //To compute area of rectangle
        {
            return l*b;
        }

        float area(float s1,float s2,float s3)    //To compute area of triangle
        {
            float s=(s1+s2+s3)/2;
            return sqrt(s*(s-s1)*(s-s2)*(s-s3));
        }
};

void main()
{
    float s1,s2,s3;
    int choice;
    Funcoverload f;
    clrscr();
    while(1)
    {
        cout<<"Program demonstrates Function Overloaded...!!!"<<endl;
        cout<<"1.To find area of square"<<endl;
        cout<<"2.To find area of rectangle"<<endl;
        cout<<"3.To find the area of triangle"<<endl;
        cout<<"4.Exit"<<endl;
        cout<<"Enter your Choice"<<endl;
        cin>>choice;
```



```

switch(choice)
{
    case 1: cout<<"Enter the input for square"<<endl;
            cin>>s1;
            cout<<"Area of Square= "<<f.area(s1)<<endl;
            break;
    case 2: cout<<"Enter the input for rectangle"<<endl;
            cin>>s1>>s2;
            cout<<"Area of Rectangle= "<<f.area(s1,s2)<<endl;
            break;
    case 3: cout<<"Enter the input for triangle"<<endl;
            cin>>s1>>s2>>s3;
            cout<<"Area of Triangle= "<<f.area(s1,s2,s3)<<endl;
            break;
    case 4: cout<<"End of Program....."<<endl;
            getch();
            exit(1);
    default:cout<<"Invallid Choice....!!!"<<endl;
}
getch();
}
}

```

OUTPUT:

```

Program demonstrates Function Overloaded...!!!
1.To find area of square
2.To find area of rectangle
3.To find the area of triangle
4.Exit
Enter your Choice
1
Enter the input for square
4
Area of Square= 16
Program demonstrates Function Overloaded...!!!
1.To find area of square
2.To find area of rectangle
3.To find the area of triangle
4.Exit
Enter your Choice
2
Enter the input for rectangle
5 6
Area of Rectangle= 30

```

```

Program demonstrates Function Overloaded...!!!
1.To find area of square
2.To find area of rectangle
3.To find the area of triangle
4.Exit
Enter your Choice
3
Enter the input for triangle
3 4 5
Area of Triangle= 6
Program demonstrates Function Overloaded...!!!
1.To find area of square
2.To find area of rectangle
3.To find the area of triangle
4.Exit
Enter your Choice
4
End of Program....

```



PROGRAM 9:

Write a C++ program to find cube of a number using inline function.

```
#include<iostream.h>
#include<conio.h>

inline int cube(int a)                                //Inline function definition
{
    return a*a*a;
}

void main()                                          //Main Function
{
    int n;
    clrscr();
    cout<<"Enter the input number"<<endl;
    cin>>n;
    cout<<"Cube of"<<" = "<<cube(n);                //Inline function call
    getch();
}
```

OUTPUT 1:

```
Enter the input number
3
Cube of = 27
```

OUTPUT 2:

```
Enter the input number
5
Cube of = 125
```



PROGRAM 10:

Write a C++ program to find sum of the series $1 + x + x^2 + x^3 + \dots x^n$ using constructors.

```
#include<iostream.h>
#include<conio.h>
#include<math.h>
class Series
{
    private:
        int sum, x, n;
    public:
        Series(int y, int m)                // Parameterized Constructor
        {
            sum = 1;
            x = y;
            n = m;
        }
        int sumseries( );
};

int Series::sumseries( )
{
    for(int i=1;i<=n;i++)
        sum=sum+pow(x,i);
    return sum;
}

void main()
{
    int x,n;
    clrscr();
    cout<<"Enter the value for Base(X)="<<endl;
    cin>>x;
    cout<<"Enter the value for Power(n)"<<endl;
```



```
cin>>n;
Series s1(x,n); // Calling Parameterized Constructor
Series s2 = s1; // Copy Constructor
cout<<"Sum of Series using Parameterised Constructor:";
cout<<s1.sumseries()<<endl;
cout<<"Sum of Series using Copy Constructor:";
cout<<s2.sumseries( );
getch();
}
```

OUTPUT 1:

```
Enter the value for Base(X)=
2
Enter the value for Power(n)
4
Sum of Series using Parameterised Constructor:31
Sum of Series using Copy Constructor:31
```

OUTPUT 2:

```
Enter the value for Base(X)=
3
Enter the value for Power(n)
5
Sum of Series using Parameterised Constructor:364
Sum of Series using Copy Constructor:364_
```



PROGRAM 11:

Create a base class containing the data member roll number and name. Also create a member function to read and display the data using the concept of single level inheritance. Create a derived class that contains marks of two subjects and total marks as the data members.

```
#include<iostream.h>
#include<conio.h>

class Student                                // Base Class
{
    private:
        long rollnumber;
        char name[20];
    public:
        void readdata( )
        {
            cout<<"Enter the Roll Number: ";
            cin>>rollnumber;
            cout<<"Enter the Student Name:";
            cin>>name;
        }
        void display( )
        {
            cout<<"\nRoll Number      :"<<rollnumber<<endl;
            cout<<"Student Name:"<<name<<endl;
        }
};

class Report : public Student                // Derived class
{
    private:
        int marks1, marks2, total;
    public:
        void readmarks( )
        {
            cout<<"\nEnter Subject 1 Marks: ";
            cin>>marks1;
            cout<<"Enter Subject 2 Marks: ";
            cin>>marks2;
        }
        void compute( )
        {
            total = marks1 + marks2;
        }
};
```




```
        cout<<endl<<"Total Marks : "<<total;
    }
};

void main( )
{
    Report R;                // Create an object R to process Student data
    clrscr();
    R.readdata();
    R.display();
    R.readmarks( );
    R.compute();
    getch();
}
```

OUTPUT 1:

```
Enter the Roll Number: 243850
Enter the Student Name:Keerthi

Roll Number      :243850
Student Name     :Keerthi

Enter Subject 1 Marks: 89
Enter Subject 2 Marks: 92

Total Marks   : 181_
```

OUTPUT 2:

```
Enter the Roll Number: 123456
Enter the Student Name: Akash

Roll Number      :123456
Student Name     :Akash

Enter Subject 1 Marks: 65
Enter Subject 2 Marks: 78

Total Marks   : 143
```



PROGRAM 12:

Create a class containing the following data members Register_No, Name and Fees. Also create a member function to read and display the data using the concept of pointers to objects.

```
#include<iostream.h>
#include<conio.h>

class Student
{
    private:
        long regno;
        char name[20];
        float fees;
    public:
        void readdata( );
        void display( );
};

void Student::readdata( )
{
    cout<<"Enter the Register Number:"<<endl;
    cin>>regno;
    cout<<"Enter the Student Name:"<<endl;
    cin>>name;
    cout<<"Enter the Fees:"<<endl;
    cin>>fees;
}

void Student::display( )
{
    cout<<"Register Number      : "<<regno<<endl;
    cout<<"Student Name   : "<<name<<endl;
    cout<<"Fees          : "<<fees<<endl;
}

void main( )
{
    Student *S;                // Create a pointer to point Student object
    clrscr( );
    S->readdata( );            // Access Student data member using a pointer
    S->display( );              // Display data using a pointer
    getch( );
}
```



OUTPUT 1:

```
Enter the Register Number:
243850
Enter the Student Name:
Keerthi
Enter the Fees:
14050
Register Number : 243850
Student Name    : Keerthi
Fees           : 14050
```

OUTPUT 2:

```
Enter the Register Number:
12345
Enter the Student Name:
Akash
Enter the Fees:
25000
Register Number : 12345
Student Name    : Akash
Fees           : 25000
```



PROGRAM 13:

Write a C++ program to perform push items into the stack.

```
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
#define MAX 3
class Stack
{
    private:
    int s[MAX], top;
    public:
        Stack()                // Constructor to initialize TOP pointer
        {
            top = -1;
        }
        void push(int);        // Member Function Declaration
        void display( );
};
void Stack::push(int item)
{
    if(top == MAX-1)
        cout<<"Stack is Full....Overflow!!!"<<endl;
    else
    {
        top++;
        s[top]=item;
    }
}
void Stack::display( )
{
    if(top == -1)
        cout<<"Empty Stack!!!"<<endl;
    else
    {
        for(int i=0; i<=top; i++)
            cout<<endl<<s[i];
        cout<<"-->top element"<<endl;
    }
    getch( );
}
void main( )
{
    Stack s;
    int choice, ele;
```



```

clrscr();
while(1)
{
    cout<<"Stack Push Operation Menu"<<endl;
    cout<<"1. PUSH"<<endl;
    cout<<"2. DISPLAY"<<endl;
    cout<<"3. EXIT"<<endl;
    cout<<"Enter your Choice"<<endl;
    cin>>choice;
    switch(choice)
    {
        case 1: cout<<"Push Operation"<<endl;
                cout<<"Enter the value of element:"<<endl;
                cin>>ele;
                s.push(ele);
                break;
        case 2: cout<<"Stack elements are:"<<endl;
                s.display( );
                break;
        case 3: cout<<"End of Stack Operation"<<endl;
                getch( );
                exit(1);
        default:cout<<"Invalid choice.....!!!"<<endl;
    }
    getch();
}
}

```

OUTPUT:

```

Stack Push Operation Menu
1. PUSH
2. DISPLAY
3. EXIT
Enter your Choice
2
Stack elements are:
Empty Stack!!!
Stack Push Operation Menu
1. PUSH
2. DISPLAY
3. EXIT
Enter your Choice
1
Push Operation
Enter the value of element:
10
Stack Push Operation Menu
1. PUSH
2. DISPLAY
3. EXIT
Enter your Choice
1

```

```

Push Operation
Enter the value of element:
20
Stack Push Operation Menu
1. PUSH
2. DISPLAY
3. EXIT
Enter your Choice
1
Push Operation
Enter the value of element:
30
Stack Push Operation Menu
1. PUSH
2. DISPLAY
3. EXIT
Enter your Choice
1
Push Operation
Enter the value of element:
40
Stack is Full....Overflow!!!

```

```

Stack Push Operation Menu
1. PUSH
2. DISPLAY
3. EXIT
Enter your Choice
2
Stack elements are:
10
20
30-->top element
Stack Push Operation Menu
1. PUSH
2. DISPLAY
3. EXIT
Enter your Choice
3
End of Stack Operation

```



PROGRAM 14:

Write a C++ program to perform pop items into the stack.

```
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
#define MAX 3
class Stack
{
    private:
        int s[MAX], top;
    public:
        Stack()                // Constructor to initialize TOP pointer
        {
            top = -1;
        }
        void push(int);
        int pop();              // Member Functions Declaration
        void display( );
};

void Stack::push(int item)
{
    if(top == MAX-1)
        cout<<"Stack is Full....Overflow!!!"<<endl;
    else
    {
        top++;
        s[top]=item;
    }
}

int Stack::pop()
{
    int item;
    if(top == -1)
        cout<<"Stack Empty!!!...Can't POP"<<endl;
    else
    {
        item = s[top];
        top--;
    }
    return item;
}
```



```

void Stack::display( )
{
    if(top == -1)
        cout<<"Stack Empty!!!"<<endl;
    else
    {
        for(int i=0; i<=top; i++)
            cout<<endl<<s[i];
        cout<<"-->top element"<<endl;
    }
}

void main( )
{
    Stack s;
    int choice, ele;
    clrscr( );
    while(1)
    {
        cout<<"\n Stack Push & Pop Operation Menu"<<endl;
        cout<<"1.PUSH"<<endl;
        cout<<"2.POP"<<endl;
        cout<<"3.DISPLAY"<<endl;
        cout<<"4.EXIT"<<endl;
        cout<<"Enter your Choice"<<endl;
        cin>>choice;
        switch(choice)
        {
            case 1: cout<<"Push Operation"<<endl;
                    cout<<"enter the value of an element"<<endl;
                    cin>>ele;
                    s.push(ele);
                    break;
            case 2: cout<<"Pop Operation"<<endl;
                    cout<<"Popped Element is: "<<s.pop( );
                    break;
            case 3: cout<<"Stack elements are:"<<endl;
                    s.display( );
                    break;
            case 4: cout<<"End of the Stack Operetion"<<endl;
                    getch( );
                    exit(1);
            default:cout<<"Invalid Choice...!!!"<<endl;
                    break;
        }
        getch();
    }
}

```



}

OUTPUT:

<pre>Stack Push & Pop Operation Menu 1.PUSH 2.POP 3.DISPLAY 4.EXIT Enter your Choice 2 Pop Operation Popped Element is: Stack Empty!!!...Can't POP Stack Push & Pop Operation Menu 1.PUSH 2.POP 3.DISPLAY 4.EXIT Enter your Choice 3 Stack elements are: Stack Empty!!!</pre>	<pre>Stack Push & Pop Operation Menu 1.PUSH 2.POP 3.DISPLAY 4.EXIT Enter your Choice 1 Push Operation enter the value of an element 10 Stack Push & Pop Operation Menu 1.PUSH 2.POP 3.DISPLAY 4.EXIT Enter your Choice 1 Push Operation enter the value of an element 20</pre>	<pre>Stack Push & Pop Operation Menu 1.PUSH 2.POP 3.DISPLAY 4.EXIT Enter your Choice 1 Push Operation enter the value of an element 30 Stack Push & Pop Operation Menu 1.PUSH 2.POP 3.DISPLAY 4.EXIT Enter your Choice 1 Push Operation enter the value of an element 40 Stack is Full...Overflow!!!</pre>
--	---	---

<pre>Stack Push & Pop Operation Menu 1.PUSH 2.POP 3.DISPLAY 4.EXIT Enter your Choice 3 Stack elements are: 10 20 30-->top element Stack Push & Pop Operation Menu 1.PUSH 2.POP 3.DISPLAY 4.EXIT Enter your Choice 2 Pop Operation Popped Element is: 30_</pre>	<pre>Stack Push & Pop Operation Menu 1.PUSH 2.POP 3.DISPLAY 4.EXIT Enter your Choice 3 Stack elements are: 10 20-->top element Stack Push & Pop Operation Menu 1.PUSH 2.POP 3.DISPLAY 4.EXIT Enter your Choice 4 End of the Stack Operation</pre>
--	---



PROGRAM 15:

Write a C++ program to perform Enqueue and Dequeue.

```
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
#define MAX 3
class Queue
{
    private:
        int q[MAX],front,rear;
    public:
        Queue()                // Constructor to initialize FRONT and REAR pointer
        {
            front = -1;
            rear = -1;
        }
        void enqueue(int);
        int dequeue();        // Member Functions Declaration
        void display();
};
void Queue::enqueue(int item)
{
    if(rear == MAX-1)
    {
        cout<<"Queue is full.....Overflow!!!"<<endl;
        getch();
        exit(0);
    }
    if(front == -1)
    {
        front = 0;
        rear = 0;
    }
    else
        rear++;
    q[rear] = item;
    cout<<"Item Inserted: "<<item<<endl;
}

int Queue::dequeue()
{
    int item;
    if(front == -1)
    {
```



```
        cout<<"Queue is Empty....Underflow!!!"<<endl;
        //getch();
        //exit(1);
    }
    item = q[front];
    if(front == rear)
    {
        front = -1;
        rear = -1;
    }
    else
        front++;
    return item;
}

void Queue::display()
{
    if(front == -1)
        cout<<"Queue is Empty!!!"<<endl;
    else
        for(int i=front; i<=rear; i++)
            cout<<q[i]<<endl;
}

void main()
{
    int ele, choice;
    Queue q;
    clrscr();
    while(1)
    {
        cout<<"\nQueue Operation Menu"<<endl;
        cout<<"1.Adding Element"<<endl;
        cout<<"2.Deleting Element"<<endl;
        cout<<"3.Display"<<endl;
        cout<<"4.Exit"<<endl;
        cout<<"Enter your Choice"<<endl;
        cin>>choice;
        switch(choice)
        {
            case 1: cout<<"Enter the element to be inserted"<<endl;
                    cin>>ele;
                    q.enqueue(ele);
                    break;
            case 2: cout<<"Deleted Item = "<<q.dequeue();
                    break;
```



```

        case 3: cout<<"The Queue Contents:"<<endl;
                q.display( );
                break;
        case 4: cout<<"End of Queue Operation"<<endl;
                getch();
                exit(1);
        default:cout<<"Invalid Choice...!!!"<<endl;
    }
    getch();
}
}

```

OUTPUT:

```

Queue Operation Menu
1.Adding Element
2.Deleting Element
3.Display
4.Exit
Enter your Choice
2
Deleted Item =
Queue is Empty...Underflow!!!

Queue Operation Menu
1.Adding Element
2.Deleting Element
3.Display
4.Exit
Enter your Choice
3
The Queue Contents:
Queue is Empty!!!

```

```

Queue Operation Menu
1.Adding Element
2.Deleting Element
3.Display
4.Exit
Enter your Choice
1
Enter the element to be inserted
10
Item Inserted: 10

Queue Operation Menu
1.Adding Element
2.Deleting Element
3.Display
4.Exit
Enter your Choice
1
Enter the element to be inserted
20
Item Inserted: 20

```

```

Queue Operation Menu
1.Adding Element
2.Deleting Element
3.Display
4.Exit
Enter your Choice
1
Enter the element to be inserted
30
Item Inserted: 30

Queue Operation Menu
1.Adding Element
2.Deleting Element
3.Display
4.Exit
Enter your Choice
1
Enter the element to be inserted
40
Queue is full.....Overflow!!!

```

```

Queue Operation Menu
1.Adding Element
2.Deleting Element
3.Display
4.Exit
Enter your Choice
3
The Queue Contents:
10
20
30

```

```

Queue Operation Menu
1.Adding Element
2.Deleting Element
3.Display
4.Exit
Enter your Choice
2
Deleted Item = 10

```

```

Queue Operation Menu
1.Adding Element
2.Deleting Element
3.Display
4.Exit
Enter your Choice
2
Deleted Item = 20
Queue Operation Menu
1.Adding Element
2.Deleting Element
3.Display
4.Exit
Enter your Choice
3
The Queue Contents:
30

```



SECTION - B

*STRUCTURED QUERY
LANGUAGE (SQL)*

PROGRAM 1:

Generate the electricity bill for one customer.

Create a table for house hold Electricity bill with the following fields.

Field Name	Type
RR_NO	VARCHAR2(10)
CUS_NAME	VARCHAR2(15)
BILLING_DATE	DATE
UNITS	NUMBER(4)

Insert 10 records into the table.

1. Check the structure of table and note your observation.
2. Add two fields to the table.
 - a. BILL_AMT NUMBER(6,2)
 - b. DUE_DATE DATE
3. Compute the bill amount for each customer as per the following rules.
 - a. MIN_AMT Rs. 50
 - b. First 100 units Rs 4.50/Unit
 - c. >100 units Rs. 5.50/Unit
4. Compute due date as BILLING_DATE + 15 Days
5. List all the bills generated.

Solution:

First we have to create the table **EBILL** using **CREATE TABLE** command.

```
SQL> CREATE TABLE EBILL
2  (
3  RR_NO  VARCHAR2(10),
4  CUS_NAME VARCHAR(15),
5  BILLING_DATE DATE,
6  UNITS  NUMBER(4)
7  );
```

Table created.

Insert 10 records into the table using INSERT commands

```
SQL> INSERT INTO EBILL VALUES ('EH 1003', 'ARUN KUMAR', '12-MAR-16',98);
SQL> INSERT INTO EBILL VALUES ('EH 2005', 'NAVEEN', '14-MAR-16',108);
SQL> INSERT INTO EBILL VALUES ('EH 2007','VARUN', '18-FEB-16',157);
SQL> INSERT INTO EBILL VALUES ('EH 3009', 'DAVID', '11-APR-16',77);
SQL> INSERT INTO EBILL VALUES ('EH 3010', 'JHON', '01-MAR-16',89);
SQL> INSERT INTO EBILL VALUES ('EH 3013', 'AKSHAY', '02-FEB-16',68);
SQL> INSERT INTO EBILL VALUES ('EH 1010', 'CHANDRU', '12-MAR-16',108);
SQL> INSERT INTO EBILL VALUES ('EH 1008', 'GHANAVI', '12-MAR-16',132);
SQL> INSERT INTO EBILL VALUES ('EH 2105', 'DRUVA', '12-MAR-16',87);
```



```
SQL> INSERT INTO EBILL VALUES ('EH 3041', 'SHREYA', '12-MAR-16', 127);
```

```
SQL> SELECT * FROM EBILL;
```

RR_NO	CUS_NAME	BILLING_D	UNITS
EH 1003	ARUN KUMAR	12-MAR-16	98
EH 2005	NAVEEN	14-MAR-16	108
EH 2007	VARUN	18-FEB-16	157
EH 3009	DAVID	11-APR-16	77
EH 3010	JHON	01-MAR-16	89
EH 3013	AKSHAY	02-FEB-16	68
EH 1010	CHANDRU	12-MAR-16	108
EH 1008	GHANAVI	12-MAR-16	132
EH 2105	DRUVA	12-MAR-16	87
EH 3041	SHREYA	12-MAR-16	127

10 rows selected.

1. Check the structure of table and note your observation.

```
SQL> DESC EBILL;
```

Name	Null?	Type
RR_NO		VARCHAR2(10)
CUS_NAME		VARCHAR2(15)
BILLING_DATE		DATE
UNITS		NUMBER(4)

2. Add two fields to the table.

- BILL_AMT NUMBER(6,2)
- DUE_DATE DATE

```
SQL> ALTER TABLE EBILL ADD(BILL_AMT NUMBER(6,2));
```

Table altered.

```
SQL> ALTER TABLE EBILL ADD(DUE_DATE DATE);
```

Table altered.

3. Compute the bill amount for each customer as per the following rules.

- MIN_AMT Rs. 50
- First 100 units Rs 4.50/Unit
- >100 units Rs. 5.50/Unit

COMMAND 1:

```
SQL> UPDATE EBILL SET BILL_AMT=100 + UNITS *4.25 WHERE UNITS <=100;
```

5 rows updated.

COMMAND 2:

```
SQL> UPDATE EBILL SET BILL_AMT=100 + 100 *4.25 + (UNITS -100) *5 WHERE UNITS >100;
```

5 rows updated.



4. Compute due date as BILLING_DATE + 15 Days

```
SQL> UPDATE EBILL SET DUE_DATE = BILLING_DATE + 15;
```

10 rows updated.

5. List all the bills generated.

```
SQL> SELECT * FROM EBILL;
```

RR_NO	CUS_NAME	BILLING_D	UNITS	BILL_AMT	DUE_DATE
EH 1003	ARUN KUMAR	12-MAR-16	98	516.5	27-MAR-16
EH 2005	NAVEEN	14-MAR-16	108	565	29-MAR-16
EH 2007	VARUN	18-FEB-16	157	810	04-MAR-16
EH 3009	DAVID	11-APR-16	77	427.25	26-APR-16
EH 3010	JHON	01-MAR-16	89	478.25	16-MAR-16
EH 3013	AKSHAY	02-FEB-16	68	389	17-FEB-16
EH 1010	CHANDRU	12-MAR-16	108	565	27-MAR-16
EH 1008	GHANAUI	12-MAR-16	132	685	27-MAR-16
EH 2105	DRUVA	12-MAR-16	87	469.75	27-MAR-16
EH 3041	SHREYA	12-MAR-16	127	660	27-MAR-16

10 rows selected.



PROGRAM 2:

Create a student database and compute the results.

Create a table for class of students with the following fields.

Field Name	Type
ID_NO	NUMBER(4)
S_NAME	VARCHAR2(15)
SUB1	NUMBER(3)
SUB2	NUMBER(3)
SUB3	NUMBER(3)
SUB4	NUMBER(3)
SUB5	NUMBER(3)
SUB6	NUMBER(3)

1. Add records into the table for 10 students for Student ID, Student Name and marks in 6 subjects using INSERT command.
2. Display the description of the fields in the table using DESC command.
3. Alter the table and calculate TOTAL and PERC_MARKS.
4. Compute the RESULT as “PASS” or “FAIL” by checking if the student has scored more than 35 marks in each subject.
5. List the contents of the table.
6. Retrieve all the records of the table.
7. Retrieve only ID_NO and S_NAME of all the students.
8. List the students who have result as “PASS”.
9. List the students who have result as “FAIL”.
10. Count the number of students who have passed.
11. Count the number of students who have failed.
12. List the students who have percentage greater than 60.
13. Sort the table according to the order of ID_NO.

Solution:

First we have to create the table **CLASS** using **CREATE TABLE** command.

```
SQL> CREATE TABLE CLASS
2  (
3  ID_NO  NUMBER(4),
4  S_NAME VARCHAR2(15),
5  SUB1   NUMBER(3),
6  SUB2   NUMBER(3),
7  SUB3   NUMBER(3),
8  SUB4   NUMBER(3),
9  SUB5   NUMBER(3),
10 SUB6   NUMBER(3)
11 );
```

Table created.

1. Add records into the table for 10 students for Student ID, Student Name and marks in 6 subjects using INSERT command.

```
SQL> INSERT INTO CLASS VALUES (1401, 'PAWAN', 56, 36, 56, 78, 44, 67);
```

```
SQL> INSERT INTO CLASS VALUES (1411, 'RAJESH', 100,100,96,100,100,100);
```




```
SQL>INSERT INTO CLASS VALUES (1412, 'KARAN', 60,30,45,45,36,49);
SQL>INSERT INTO CLASS VALUES (1403, 'SACHIN', 56,60,72,57,78,67);
SQL>INSERT INTO CLASS VALUES (1410, 'PRAKASH', 96,99,97,90,78,100);
SQL>INSERT INTO CLASS VALUES (1402, 'POOJA', 30,45,39,20,33,56);
SQL>INSERT INTO CLASS VALUES (1405, 'ASHWINI', 79,65,79,70,89,88);
SQL>INSERT INTO CLASS VALUES (1406, 'PRAJWAL', 100,90,100,89,90,100);
SQL>INSERT INTO CLASS VALUES (1404, 'BALU', 35,30,78,23,44,70);
SQL>INSERT INTO CLASS VALUES (1407, 'ESHWAR', 100,100,100,98,99,100);
```

2. Display the description of the fields in the table using DESC command.

```
SQL> DESC CLASS;
```

Name	Null?	Type
ID_NO		NUMBER(4)
S_NAME		VARCHAR2(15)
SUB1		NUMBER(3)
SUB2		NUMBER(3)
SUB3		NUMBER(3)
SUB4		NUMBER(3)
SUB5		NUMBER(3)
SUB6		NUMBER(3)

3. Alter the table and calculate TOTAL and PERC_MARKS.

```
SQL> ALTER TABLE CLASS ADD
  2 (TOTAL NUMBER(3), PERC_MARKS NUMBER(6,2), RESULT VARCHAR2(10));
```

Table altered.

```
SQL> UPDATE CLASS SET TOTAL = SUB1+SUB2+SUB3+SUB4+SUB5+SUB6;
```

10 rows updated.

```
SQL> UPDATE CLASS SET PERC_MARKS = TOTAL/6;
```

10 rows updated.

4. Compute the RESULT as “PASS” or “FAIL” by checking if the student has scored more than 35 marks in each subject.

```
SQL> UPDATE CLASS SET RESULT = 'PASS'
  2 WHERE (SUB1>=35 AND SUB2>=35 AND SUB3>=35 AND SUB4>=35 AND SUB5>=35 AND SUB6>=35);
```

7 rows updated.

```
SQL>
SQL> UPDATE CLASS SET RESULT = 'FAIL'
  2 WHERE (SUB1<35 OR SUB2<35 OR SUB3<35 OR SUB4<35 OR SUB5<35 OR SUB6<35);
```

3 rows updated.

5. List the contents of the table.

6. Retrieve all the records of the table.



```
SQL> SELECT * FROM CLASS;
```

ID_NO	S_NAME	SUB1	SUB2	SUB3	SUB4	SUB5	SUB6	TOTAL	PERC_MARKS	RESULT
1401	PAWAN	56	36	56	78	44	67	337	56.17	PASS
1411	RAJESH	100	100	96	100	100	100	596	99.33	PASS
1412	KARAN	60	30	45	45	36	49	265	44.17	FAIL
1403	SACHIN	56	60	72	57	78	67	390	65	PASS
1410	PRAKASH	96	99	97	90	78	100	560	93.33	PASS
1402	POOJA	30	45	39	20	33	56	223	37.17	FAIL
1405	ASHWINI	79	65	79	70	89	88	470	78.33	PASS
1406	PRAJWAL	100	90	100	89	90	100	569	94.83	PASS
1404	BALU	35	30	78	23	44	70	280	46.67	FAIL
1407	ESHWAR	100	100	100	98	99	100	597	99.5	PASS

10 rows selected.

7. Retrieve only ID_NO and S_NAME of all the students.

```
SQL> SELECT ID_NO, S_NAME FROM CLASS;
```

ID_NO	S_NAME
1401	PAWAN
1411	RAJESH
1412	KARAN
1403	SACHIN
1410	PRAKASH
1402	POOJA
1405	ASHWINI
1406	PRAJWAL
1404	BALU
1407	ESHWAR

10 rows selected.

8. List the students who have result as "PASS".

```
SQL> SELECT * FROM CLASS WHERE RESULT='PASS';
```

ID_NO	S_NAME	SUB1	SUB2	SUB3	SUB4	SUB5	SUB6	TOTAL	PERC_MARKS	RESULT
1401	PAWAN	56	36	56	78	44	67	337	56.17	PASS
1411	RAJESH	100	100	96	100	100	100	596	99.33	PASS
1403	SACHIN	56	60	72	57	78	67	390	65	PASS
1410	PRAKASH	96	99	97	90	78	100	560	93.33	PASS
1405	ASHWINI	79	65	79	70	89	88	470	78.33	PASS
1406	PRAJWAL	100	90	100	89	90	100	569	94.83	PASS
1407	ESHWAR	100	100	100	98	99	100	597	99.5	PASS

7 rows selected.

9. List the students who have result as "FAIL".

```
SQL> SELECT * FROM CLASS WHERE RESULT='FAIL';
```

ID_NO	S_NAME	SUB1	SUB2	SUB3	SUB4	SUB5	SUB6	TOTAL	PERC_MARKS	RESULT
1412	KARAN	60	30	45	45	36	49	265	44.17	FAIL
1402	POOJA	30	45	39	20	33	56	223	37.17	FAIL
1404	BALU	35	30	78	23	44	70	280	46.67	FAIL

10. Count the number of students who have passed.

```
SQL> SELECT COUNT(*) FROM CLASS WHERE RESULT = 'PASS';
```

COUNT(*)
7



11. Count the number of students who have failed.

```
SQL> SELECT COUNT(*) FROM CLASS WHERE RESULT='FAIL';
```

```

COUNT(*)
-----
          3

```

12. List the students who have percentage greater than 60.

```
SQL> SELECT * FROM CLASS WHERE PERC_MARKS>60;
```

ID_NO	S_NAME	SUB1	SUB2	SUB3	SUB4	SUB5	SUB6	TOTAL	PERC_MARKS	RESULT
1411	RAJESH	100	100	96	100	100	100	596	99.33	PASS
1403	SACHIN	56	60	72	57	78	67	390	65	PASS
1410	PRAKASH	96	99	97	90	78	100	560	93.33	PASS
1405	ASHWINI	79	65	79	70	89	88	470	78.33	PASS
1406	PRAJWAL	100	90	100	89	90	100	569	94.83	PASS
1407	ESHWAR	100	100	100	98	99	100	597	99.5	PASS

6 rows selected.

13. Sort the table according to the order of ID_NO.

```
SQL> SELECT * FROM CLASS ORDER BY ID_NO;
```

ID_NO	S_NAME	SUB1	SUB2	SUB3	SUB4	SUB5	SUB6	TOTAL	PERC_MARKS	RESULT
1401	PAWAN	56	36	56	78	44	67	337	56.17	PASS
1402	POOJA	30	45	39	20	33	56	223	37.17	FAIL
1403	SACHIN	56	60	72	57	78	67	390	65	PASS
1404	BALU	35	30	78	23	44	70	280	46.67	FAIL
1405	ASHWINI	79	65	79	70	89	88	470	78.33	PASS
1406	PRAJWAL	100	90	100	89	90	100	569	94.83	PASS
1407	ESHWAR	100	100	100	98	99	100	597	99.5	PASS
1410	PRAKASH	96	99	97	90	78	100	560	93.33	PASS
1411	RAJESH	100	100	96	100	100	100	596	99.33	PASS
1412	KARAN	60	30	45	45	36	49	265	44.17	FAIL

10 rows selected.



PROGRAM 3:

Generate the Employee details and compute the salary based on the department.
Create the following table EMPLOYEE.

Field Name	Type
EMP_ID	NUMBER(4)
DEPT_ID	NUMBER(2)
EMP_NAME	VARCHAR2(10)
EMP_SALARY	NUMBER(5)

Create another table DEPARTMENT.

Field Name	Type
DEPT_ID	NUMBER(2)
DEPT_NAME	VARCHAR2(10)
SUPERVISOR	VARCHAR2(10)

Assume the DEPARTMENT names as Purchase (Id-01), Accounts (Id-02), Sales (Id-03), and Apprentice (Id-04)

Enter 10 rows of data for table EMPLOYEE and 4 rows of data for DEPARTMENT table.

Write the SQL statements for the following:

1. Find the names of all employees who work for the Accounts department.
2. How many employees work for Accounts department?
3. What are the Minimum, Maximum and Average salary of employees working for Accounts department?
4. List the employees working for particular supervisor.
5. Retrieve the department names for each department where only one employee works.
6. Increase the salary of all employees in the sales department by 15%.
7. Add a new Column to the table EMPLOYEE called BONUS NUMBER (5) and compute 5% of the salary to the said field.
8. Delete all the rows for the employee in the Apprentice department.

Solution:

First we have to create two tables, **EMPLOYEE** and **DEAPRTMENT**.

<pre>SQL> CREATE TABLE EMPLOYEE 2 (3 EMP_ID NUMBER(4), 4 DEPT_ID NUMBER(2), 5 EMP_NAME VARCHAR2(10), 6 EMP_SALARY NUMBER(5) 7);</pre> <p>Table created.</p>	<pre>SQL> CREATE TABLE DEPARTMENT 2 (3 DEPT_ID NUMBER(2), 4 DEPT_NAME VARCHAR2(10), 5 SUPERVISOR VARCHAR2(10) 6);</pre> <p>Table created.</p>
--	--

To Insert 10 records into the table EMPLOYEE using INSERT INTO command.

```
SQL> INSERT INTO EMPLOYEE VALUES (101, 01, 'ARUN', 15000);
SQL> INSERT INTO EMPLOYEE VALUES (104, 02, 'MOHAN', 20000);
SQL> INSERT INTO EMPLOYEE VALUES (105, 03, 'SUMAN', 22000);
```



```
SQL> INSERT INTO EMPLOYEE VALUES (106, 02, 'SUSHMA', 18000);
SQL> INSERT INTO EMPLOYEE VALUES (109, 01, 'KUSHI', 22300);
SQL> INSERT INTO EMPLOYEE VALUES (110, 02, 'VIDHYA', 15000);
SQL> INSERT INTO EMPLOYEE VALUES (102, 02, 'KAVYA', 21300);
SQL> INSERT INTO EMPLOYEE VALUES (107, 03, 'AKASH', 18200);
SQL> INSERT INTO EMPLOYEE VALUES (108, 04, 'NAWAZ', 12000);
SQL> INSERT INTO EMPLOYEE VALUES (103, 02, 'DEEPAK', 24000);
```

To insert 4 records into the table DEPARTMENT using the INSERT INTO command.

```
SQL>INSERT INTO DEPARTMENT VALUES (01, 'PURCHASE', 'KRISHNA');
SQL>INSERT INTO DEPARTMENT VALUES (02, 'ACCOUNTS', 'TANVEER');
SQL>INSERT INTO DEPARTMENT VALUES (03, 'SALES', 'SURYA');
SQL>INSERT INTO DEPARTMENT VALUES (04, 'APPRENTICE', 'HARSHA');
```

1. Find the names of all employees who work for the Accounts department.

```
SQL> SELECT * FROM EMPLOYEE WHERE DEPT_ID=
2 (SELECT DEPT_ID FROM DEPARTMENT
3 WHERE DEPT_NAME='ACCOUNTS');
```

EMP_ID	DEPT_ID	EMP_NAME	EMP_SALARY
104	2	MOHAN	20000
106	2	SUSHMA	18000
110	2	VIDHYA	15000
102	2	KAVYA	21300
103	2	DEEPAK	24000

2. How many employees work for Accounts department?

```
SQL> SELECT COUNT(*) FROM EMPLOYEE WHERE DEPT_ID =
2 (SELECT DEPT_ID FROM DEPARTMENT
3 WHERE DEPT_NAME='ACCOUNTS');
```

COUNT(*)
5

3. What are the Minimum, Maximum and Average salary of employees working for Accounts department?

```
SQL> SELECT MIN(EMP_SALARY),
2 MAX(EMP_SALARY),
3 AVG(EMP_SALARY)
4 FROM EMPLOYEE WHERE DEPT_ID =
5 (SELECT DEPT_ID FROM DEPARTMENT
6 WHERE DEPT_NAME='ACCOUNTS');
```

MIN(EMP_SALARY)	MAX(EMP_SALARY)	AUG(EMP_SALARY)
15000	24000	19660



4. List the employees working for particular supervisor.

```
SQL> SELECT * FROM EMPLOYEE WHERE DEPT_ID =
  2 (SELECT DEPT_ID FROM DEPARTMENT
  3 WHERE SUPERVISOR='SURYA');
```

EMP_ID	DEPT_ID	EMP_NAME	EMP_SALARY
105	3	SUMAN	22000
107	3	AKASH	18200

5. Retrieve the department names for each department where only one employee works.

```
SQL> SELECT DEPT_NAME FROM DEPARTMENT
  2 WHERE DEPT_ID IN (SELECT DEPT_ID FROM EMPLOYEE
  3 GROUP BY DEPT_ID HAVING COUNT(*)=1);
```

```
DEPT_NAME
-----
APPRENTICE
```

6. Increase the salary of all employees in the sales department by 15%.

```
SQL> UPDATE EMPLOYEE
  2 SET EMP_SALARY = EMP_SALARY + 15 * EMP_SALARY/100
  3 WHERE DEPT_ID = (SELECT DEPT_ID FROM DEPARTMENT
  4 WHERE DEPT_NAME='SALES');
```

2 rows updated.

7. Add a new Column to the table EMPLOYEE called BONUS NUMBER (5) and compute 5% of the salary to the said field.

```
SQL> ALTER TABLE EMPLOYEE ADD(BONUS NUMBER(5));
```

Table altered.

```
SQL>
SQL> UPDATE EMPLOYEE
  2 SET BONUS = 5 * EMP_SALARY/100;
```

10 rows updated.

```
SQL> SELECT * FROM EMPLOYEE;
```

EMP_ID	DEPT_ID	EMP_NAME	EMP_SALARY	BONUS
101	1	ARUN	15000	750
104	2	MOHAN	20000	1000
105	3	SUMAN	25300	1265
106	2	SUSHMA	18000	900
109	1	KUSHI	22300	1115
110	2	VIDHYA	15000	750
102	2	KAUYA	21300	1065
107	3	AKASH	20930	1047
108	4	NAWAZ	12000	600
103	2	DEEPAK	24000	1200

10 rows selected.



8. Delete all the rows for the employee in the Apprentice department.

```
SQL> DELETE FROM EMPLOYEE
2  WHERE DEPT_ID = (SELECT DEPT_ID FROM DEPARTMENT
3  WHERE DEPT_NAME='APPRENTICE');
```

1 row deleted.



SECTION - C

ADVANCED HTML

PROGRAM 1:

Write a HTML program to create a CLASS Time Table.

```
<HTML>
<HEAD>
  <TITLE> CLASS TIME TABLE </TITLE>
</HEAD>

<BODY TEXT=DARKBLUE COLOR=WHITE>
<CENTER> <H3> M D R PU SCIENCE COLLEGE </H3>
<CENTER> <H4> TIME TABLE 2016-17 </H4>

<TABLE BORDER=10 BORDERCOLOR=RED BGCOLOR=CORNSILK CELLSPACING=2
CELLPADDING=15>
<CAPTION> <B> II PUC PCMCs </B> <CAPTION>

<TR BGCOLOR=PEACHPUFF>
  <TD ROWSPAN=2 ALIGN=CENTER> <B> DAY </B> </TD>
  <TD COLSPAN=8 ALIGN=CENTER> <B> TIMINGS </B></TD>
</TR>

<TR BGCOLOR=RED>
  <TH> 9.20 - 10.20 </TH>
  <TH> 10.20 - 11.20 </TH>
  <TH> 11.20 - 11.30 </TH>
  <TH> 11.30 - 12.30 </TH>
  <TH> 12.30 - 1.30 </TH>
  <TH> 1.30 - 2.30 </TH>
  <TH> 2.30 - 3.15 </TH>
  <TH> 3.15 - 4.00 </TH>
</TR>
<TR>
  <TD> MONDAY </TD>
  <TD> MATHS </TD>
  <TD> PHYSICS </TD>
```



```

<TD ROWSPAN=6 ALIGN="CENTER">SHORT BREAK</TD>
<TD> CHEMISTRY </TD>
<TD> COMP SCI </TD>
<TD ROWSPAN=6 ALIGN=CENTER>LUNCH BREAK</TD>
<TD COLSPAN=2ALIGN=CENTER><.....COMP SCI LAB.....> </TD>
</TR>
<TR>
<TD>TUESDAY</TD>
<TD> PHYSICS </TD>
<TD> MATHS </TD>
<TD> CHEMISTRY </TD>
<TD> ENGLISH </TD>
<TD COLSPAN=2 ALIGN=CENTER><.....PHYSICS LAB.....></TD>
</TR>
<TR>
<TD> WEDNESDAY </TD>
<TD> COMP SCI </TD>
<TD> PHYSICS </TD>
<TD> MATHS </TD>
<TD> KANNADA </TD>
<TD COLSPAN=2 ALIGN=CENTER><.....CHEMISTRY LAB....></TD>
</TR>
<TR>
<TD> THURSDAY </TD>
<TD> MATHS </TD>
<TD> KANNADA </TD>
<TD> PHYSICS </TD>
<TD> ENGLISH </TD>
<TD COLSPAN=2 ALIGN=CENTER><.....COMP SCI LAB.....> </TD>
</TR>
<TR>
<TD>FRIDAY </TD>
<TD>ENGLISH </TD>
<TD>MATHS </TD>
<TD>PHYSICS </TD>

```

```

<TD>COMP SCI </TD>
<TD>CHEMISTRY </TD>
<TD>KANNADA </TD>
</TR>
<TR>
<TD> SATURDAY </TD>
<TD> MATHS </TD>
<TD> ENGLISH </TD>
<TD> CHEMISTRY </TD>
<TD> COMP SCI </TD>
<TD COLSPAN=2 ALIGN=CENTER><.....SPECIAL CLASS.....> </TD>
</TR>
</TABLE>
</CENTER>
</BODY>
</HTML>
    
```

OUTPUT:

M D R PU SCIENCE COLLEGE

TIME TABLE 2016-17

II PUC PCMCs

DAY	TIMINGS								
	9.20 - 10.20	10.20 - 11.20	11.20 - 11.30	11.30 - 12.30	12.30 - 1.30	1.30 - 2.30	2.30 - 3.15	3.15 - 4.00	
MONDAY	MATHS	PHYSICS	SHORT BREAK	CHEMISTRY	COMP SCI	LUNCH BREAK	<.....COMP SCI LAB.....>		
TUESDAY	PHYSICS	MATHS		CHEMISTRY	ENGLISH		<..... PHYSICS LAB.....>		
WEDNESDAY	COMP SCI	PHYSICS		MATHS	KANNADA		<.....CHEMISTRY LAB.....>		
THURSDAY	MATHS	KANNADA		PHYSICS	ENGLISH		<.....COMP SCI LAB.....>		
FRIDAY	ENGLISH	MATHS		PHYSICS	COMP SCI		CHEMISTRY	KANNADA	
SATURDAY	MATHS	ENGLISH		CHEMISTRY	COMP SCI		<.....SPECIAL CLASS.....>		



PROGRAM 2:**Create an HTML program with Table and Form.**

```
<HTML>
<HEAD>
  <TITLE> ONLINE APPLICATION </TITLE>
</HEAD>

<BODY>
<FORM NAME="APPFORMPUC" METHOD="POST" ACTION="IPUC_SEND.PHP">
<H3 ALIGN=CENTER> FIRST PUC APPLICATION FORM </H3>
<TABLE CELLSPACING=5 CELLPADDING=5% ALIGN=CENTER>
<TR>
  <TD ALIGN=LEFT>STUDENT NAME: </TD>
  <TD><INPUT TYPE="TEXT" NAME="STUNAME"></TD>
</TR>
<TR>
  <TD ALIGN=LEFT>FATHER NAME: </TD>
  <TD><INPUT TYPE="TEXT" NAME="FATNAME"></TD>
</TR>
<TR>
  <TD ALIGN=LEFT>FATHER OCCUPATION: </TD>
  <TD><INPUT TYPE="TEXT" NAME="FATOCC"></TD>
</TR>
<TR>
  <TD ALIGN=LEFT>DATE OF BIRTH: </TD>
  <TD><INPUT TYPE="TEXT" NAME="DOB"></TD>
</TR>
<TR>
  <TD ALIGN=LEFT>CONTACT NUMBER: </TD>
  <TD><INPUT TYPE="TEXT" NAME="CONTACT"></TD>
</TR>
<TR>
  <TD ALIGN=LEFT> EMAIL ID: </TD>
  <TD><INPUT TYPE="TEXT" NAME="EMAIL"></TD>
</TR>
<TR>
  <TD ALIGN=LEFT>UPLOAD PHOTO: </TD>
  <TD><INPUT TYPE=FILE NAME="PHOTO"></TD>
</TR>
<TR>
  <TD ALIGN=LEFT>GENDER: </TD>
  <TD><INPUT TYPE=RADIO NAME=GEN VALUE="M">MALE
  <INPUT TYPE=RADIO NAME=GEN VALUE="F">FEMALE</TD>
</TR>
```



```

<TR>
  <TD ALGIN=LEFT> CATEGORY:</TD>
  <TD ALGIN=LEFT><INPUT TYPE="TEXT" NAME="CATEGORY">
  <SELECT NAME="DROPDOWN" >
    <OPTION VALUE=1>GM</OPTION>
    <OPTION VALUE=2>SC</OPTION>
    <OPTION VALUE=3>ST</OPTION>
    <OPTION VALUE=4>C1</OPTION>
    <OPTION VALUE=5>2A</OPTION>
    <OPTION VALUE=6>2B</OPTION>
    <OPTION VALUE=7>3A</OPTION>
    <OPTION VALUE=8>3B</OPTION>
  </SELECT>
</TD>
</TR>
<TR>
  <TD ALIGN=LEFT>Indicate the Board PASSED: </TD>
  <TD ALGIN=LEFT><INPUT TYPE="TEXT" NAME="QUALIFICATION" >
  <SELECT NAME="DROPDOWN" SIZE=4 ID=QUALI>
    <OPTION VALUE=1>SSLC</OPTION>
    <OPTION VALUE=2>CBSE</OPTION>
    <OPTION VALUE=3>ICSE</OPTION>
    <OPTION VALUE=4>OTHER STATE</OPTION>
  </SELECT>
</TD>
</TR>
<TR>
  <TD ALGIN=LEFT> STUDENT ADDRESS :</TD>
  <TD> <TEXTAREA ROWS=2 COLS=15 NAME=ADD></TEXTAREA>
  <P> Enter the Contact Address with Pin Code </P>
</TD>
</TR>
<TR>
  <TD ALGIN=LEFT>SUBJECT CHOOSEN: </TD>
  <TD ALGIN=LEFT>
    <INPUT TYPE=CHECKBOX NAME=LANG1 >KANNADA
    <INPUT TYPE=CHECKBOX NAME=LANG2 >ENGLISH
    <INPUT TYPE=CHECKBOX NAME=SUB1 >PHYSICS
    <INPUT TYPE=CHECKBOX NAME=SUB2 >CHEMISTRY
    <INPUT TYPE=CHECKBOX NAME=SUB3 >MATHS
    <INPUT TYPE=CHECKBOX NAME=SUB4 >BIOLOGY
    <INPUT TYPE=CHECKBOX NAME=SUB5 >COMP SCI
    <INPUT TYPE=CHECKBOX NAME=SUB56>ELECTRONICS
  </TD>
</TR>

```



```

<TR>
  <TD><INPUT TYPE="SUBMIT" VALUE="SUBMIT THE FORM"> </TD>
  <TD><INPUT TYPE="RESET" VALUE="RESET THE FORM"></TD>
</TR>
</TABLE>
</FORM>
</BODY>
</HTML>

```

OUTPUT:

FIRST PUC APPLICATION FORM

STUDENT NAME:

FATHER NAME:

FATHER OCCUPATION:

DATE OF BIRTH:

CONTACT NUMBER:

EMAIL ID:

UPLOAD PHOTO: No file chosen

GENDER: MALE FEMALE

CATEGORY:

Indicate the Board PASSED:

STUDENT ADDRESS:

Enter the Contact Address with Pin Code

SUBJECT CHOSEN: KANNADA ENGLISH PHYSICS CHEMISTRY MATHS BIOLOGY COMP SCI ELECTRONICS

