

**KammavariSangham (R) - 1952**  
**K.S.Group of Institutions**  
**KS INSTITUTE OF TECHNOLOGY**  
#14, Raghuvanahalli, Kanakapura Main Road, Bengaluru-560062

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**  
**I/II SEMESTER**

**18CSL17/27 –C PROGRAMMING AND PROBLEM-  
SOLVING LABORATORY**

# **K S INSTITUTE OF TECHNOLOGY**

## **VISION**

**“To impart quality technical education with ethical values, employable skills and research to achieve excellence”**

## **MISSION**

- **To attract and retain highly qualified, experienced & committed faculty.**
- **To create relevant infrastructure**
- **Network with industry & premier institutions to encourage emergence of new ideas by providing research & development facilities to strive for academic excellence**
- **To inculcate the professional & ethical values among young students with employable skills & knowledge acquired to transform the society**

## **DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

### **VISION**

**To create competent professionals in Computer Science and Engineering with adequate skills to drive the IT industry**

### **MISSION**

- **Impart sound technical knowledge and quest for continuous learning.**
- **To equip students to furnish Computer Applications for the society through experiential learning and research with professional ethics.**
- **Encourage team work through inter-disciplinary project and evolve as leaders with social concerns.**

# DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

## Program Educational Objectives (PEO's)

**PEO1:** Excel in professional career by acquiring knowledge in cutting edge Technology and contribute to the society as an excellent employee or as an entrepreneur in the field of Computer Science & Engineering.

**PEO2:** Continuously enhance their knowledge on par with the development in IT industry and pursue higher studies in computer science & engineering.

**PEO3 :** Exhibit professionalism, cultural awareness, team work, ethics, and effective communication skills with their knowledge in solving social and environmental problems by applying computer technology.

## Program Specific Outcomes (PSO's)

**PSO1 :** Ability to understand, analyze problems and implement solutions in Programming languages, as well to apply concepts in core areas of Computer Science in association with professional bodies and clubs.

**PSO2 :** Ability to use computational skills and apply software knowledge to develop effective solutions and data to address real world challenges.

## **PROGRAM OUTCOMES (PO's)**

**PO1: Engineering Knowledge:** Apply knowledge of mathematics and science, with fundamentals of Computer Science & Engineering to be able to solve complex engineering problems related to CSE.

**PO2: Problem Analysis:** Identify, Formulate, review research literature and analyze complex engineering problems related to CSE and reaching substantiated conclusions using first principles of mathematics, natural sciences and engineering sciences

**PO3: Design/Development of solutions:** Design solutions for complex engineering problems related to CSE and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety and the cultural societal and environmental considerations

**PO4: Conduct Investigations of Complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

**PO5: Modern Tool Usage:** Create, Select and apply appropriate techniques, resources and modern engineering and IT tools including prediction and modeling to computer science related complex engineering activities with an understanding of the limitations

**PO6: The Engineer and Society:** Apply Reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the CSE professional engineering practice

**PO7: Environment and Sustainability:** Understand the impact of the CSE professional engineering solutions in societal and environmental contexts and demonstrate the knowledge of, and need for sustainable development

**PO8: Ethics:** Apply Ethical Principles and commit to professional ethics and responsibilities and norms of the engineering practice

**PO9: Individual and Team Work:** Function effectively as an individual and as a member or leader in diverse teams and in multidisciplinary Settings

**PO10: Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large such as able to comprehend and with write effective reports and design documentation, make effective presentations and give and receive clear instructions.

**PO11: Project Management and Finance:** Demonstrate knowledge and understanding of the engineering management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multi-disciplinary environments

**PO12: Life-Long Learning:** Recognize the need for and have the preparation and ability to engage in independent and life-long learning the broadest context

# **SYLLABUS**

<b>Semester</b>	<b>: I/II</b>	<b>CIE Marks</b>	<b>: 40</b>
<b>Course Code</b>	<b>: 18CPL17/27</b>	<b>SEE Marks</b>	<b>: 60</b>
<b>Teaching Hours/week (L:T:P)</b>	<b>: 0:0:2</b>	<b>Exam Hours</b>	<b>: 03</b>
<b>Credits : 01</b>			

## **Course Learning Objectives:**

This course (**18CPL17/27**) will enable students to:

- Write flowcharts, algorithms and programs.
- Familiarize the processes of debugging and execution.
- Implement basics of C programming language.
- Illustrate solutions to the laboratory programs.

## **Descriptions (if any):**

- The laboratory should be preceded or followed by a tutorial to explain the approach or algorithm being implemented or implemented for the problems given.
- Note that experiment 1 is mandatory and written in the journal.
- Questions related with experiment 1, need to be asked during viva-voce for all experiments.
- Every experiment should have algorithm and flowchart be written before writing the program.
- Code should be traced using minimum two test cases which should be recorded.
- It is preferred to implement using Linux and GCC.

## **Laboratory Programs:**

1. Familiarization with computer hardware and programming environment, concept of naming the program files, storing, compilation, execution and debugging, taking any simple C- code.

### **PART A**

2. Develop a program to solve simple computational problems using arithmetic expressions and use of each operator leading to simulation of a commercial calculator. (No built-in math function)
3. Develop a program to compute the roots of a quadratic equation by accepting the coefficients. Print appropriate messages.
4. Develop a program to find the reverse of a positive integer and check for palindrome or not. Display appropriate messages.

5. An electricity board charges the following rates for the use of electricity: for the first 200 units 80 paise per unit; for the next 100 units 90 paise per unit; beyond 300 units Rs 1 per unit. All users are charged a minimum of Rs. 100 as meter charge. If the total amount is more than Rs 400, then an additional surcharge of 15% of total amount is charged. Write a program to read the name of the user, number of units consumed and print out the charges.
6. Introduce 1D Array manipulation and implement Binary search.
7. Implement using functions to check whether the given number is prime and display appropriate messages. (No built-in math function)

### **PART B**

8. Develop a program to introduce 2D Array manipulation and implement Matrix multiplication and ensure the rules of multiplication are checked.
9. Develop a Program to compute  $\sin(x)$  using Taylor series approximation. Compare your result with the built-in Library function. Print both the results with appropriate messages.
10. Write functions to implement string operations such as compare, concatenate, string length. Convince the parameter passing techniques.
11. Develop a program to sort the given set of N numbers using Bubble sort.
12. Develop a program to find the square root of a given number N and execute for all possible inputs with appropriate messages. Note: Don't use library function  $\sqrt{n}$ .
13. Implement structures to read, write and compute average- marks and the students scoring above and below the average marks for a class of N students.
14. Develop a program using pointers to compute the sum, mean and standard deviation of all elements stored in an array of n real numbers.
15. Implement Recursive functions for Binary to Decimal Conversion.

### **Laboratory Outcomes:**

The student should be able to:

- Write algorithms, flowcharts and program for simple problems.
- Correct syntax and logical errors to execute a program.
- Write iterative and wherever possible recursive programs.
- Demonstrate use of functions, arrays, strings, structures and pointers in problem solving.

### **Conduct of Practical Examination:**

- All laboratory experiments, excluding the first, are to be included for practical examination.
- Experiment distribution
  - o For questions having only one part: Students are allowed to pick one experiment from the lot and are given equal opportunity.

<b>Subject Code: 18CPL17/27</b>	<b>Subject: C Programming Laboratory</b>
<b>Faculty: Manoj Kumar S</b>	<b>Academic Year:2020-2021</b>
<b>Class: I/II Semester</b>	<b>Department: Artificial Intelligence and Machine Learning</b>

### CO-PO Mapping:

<b>Course: C PROGRAMMING LABORATORY</b>			
<b>Type: Programming</b>		<b>Course Code:18CPL17/27</b>	
<b>No of Hours per week</b>			
Theory (Lecture Class)	Practical/Field Work/Allied Activities	Total/Week	Total teaching hours
0	2	3	40
<b>Marks</b>			
Internal Assessment	Examination	Total	Credits
40	60	100	1
<b><u>Aim/Objective of the Course:</u></b>			
<ol style="list-style-type: none"> <li>1. Write flowcharts, algorithms and programs.</li> <li>2. Familiarize the process the debugging and execution.</li> <li>3. Implement basis of C programming language.</li> <li>4. Illustrate solutions to the laboratory programs.</li> </ol>			
<b>Course Learning Outcomes:</b>			
After completing the course, the students will be able to,			
<b>CO1</b>	Illustrate the knowledge on various parts of a computer.		Understanding (K2)
<b>CO2</b>	Develop flowcharts and write algorithms for every C Programs.		Applying (K3)
<b>CO3</b>	Develop C problem solving skills.		Applying (K3)
<b>CO4</b>	Develop modular programming skills.		Applying (K3)
<b>CO5</b>	Analyze the tracing and debugging of a program.		Analyze (K4)

CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2
CO1	3	2	2	1	3	-	-	-	-	-	-	2	3	1
CO2	3	2	2	1	3	-	-	-	-	-	-	2	3	1
CO3	3	2	2	1	3	-	-	-	-	-	-	2	3	1
CO4	3	2	2	1	3	-	-	-	-	-	-	2	3	1
CO5	3	2	2	1	3	-	-	-	-	-	-	2	3	1
<b>18CP L17/27</b>	3	2	2	1	3							2	3	1

## **RUBRICS FOR LABORATORY EVALUATION**

	<b>Exceptional (10-9)</b>	<b>Acceptable (8-6)</b>	<b>Marginal (5-1)</b>	<b>Unsatisfactory (0)</b>
<b>Record (10)</b>	The Record is well written and clearly explains what the experiments are accomplishing	The record is written and useful in understanding	The record is vague only the reader can understand.	The record does not help the reader understand
<b>Program Logic (In the Observation book) (10)</b>	The program executes and meets all of the specifications.	The Program executes correctly with no errors and meets few of the specifications.	The program executes with few logical errors	Program does not execute
<b>Viva (10)</b>	Demonstrates deep knowledge; answer the questions with explanations and elaboration.	Adequate knowledge of most topics; answer the questions, but fails to elaborate.	Superficial knowledge of topic; only able to answer basic questions.	Not able to answer basic questions
<b>Internal Test (10)</b>	Internal test conducted for 10 marks. These 10 marks will be divided for write up, execution of program and viva.			

## Introduction to Hardware

### Laboratory Session-1

Write-up on Functional block diagram of Computer, CPU, Buses, Mother Board, Chip sets, Operating System & types of OS, Basics of Networking & Topology and NIC.

### Functional block diagram of Computer:

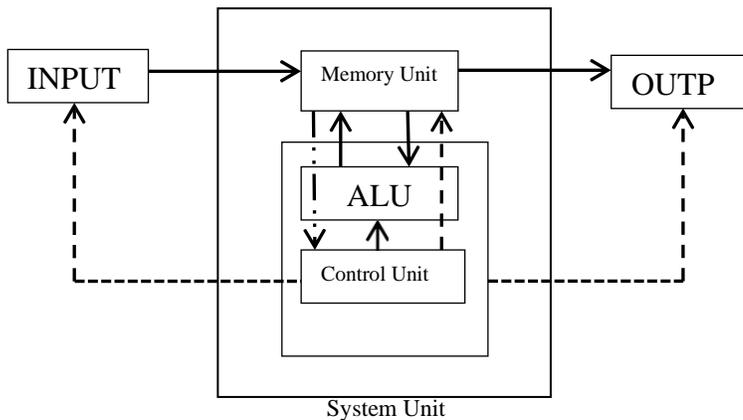
#### DESCRIPTION

A computer is an electronic device which is capable of receiving information (data) in a particular form and of performing a sequence of operations in accordance with a predetermined but variable set of procedural instructions (program) to produce a result in the form of information or signals.

A computer can process data, pictures, sound and graphics. They can solve highly complicated problems quickly and accurately.

A computer as shown in below figure basically performs five major operations or functions irrespective of their size and make. These are

- 1) It accepts data or instructions by way of input.
- 2) It stores data.
- 3) It can process data as required by the user.
- 4) It gives results in the form of output.
- 5) It controls all operations inside the computer.



**BLOCK DIAGRAM OF A COMPUTER**

- Data and results flow
- Control instructions to other units from control unit
- Instructions from memory unit to control unit



We discuss below each of these Computer operations

**1. Input:** computer receives data and instructions through the input unit. The input unit consists of one or more input devices.

Input devices include:

- Keyboard
- Mouse
- Joystick
- Scanner

Functions of input unit

- Accept the data and instructions from the outside world
- Convert it to a form that the computer can understand
- Supply the converted data to the computer system for further processing.



Keyboard



Joystick

**2. Storage:** The process of saving data and instructions permanently is known as storage. A Computer has two types of storage areas:

- Primary Storage
- Secondary Storage

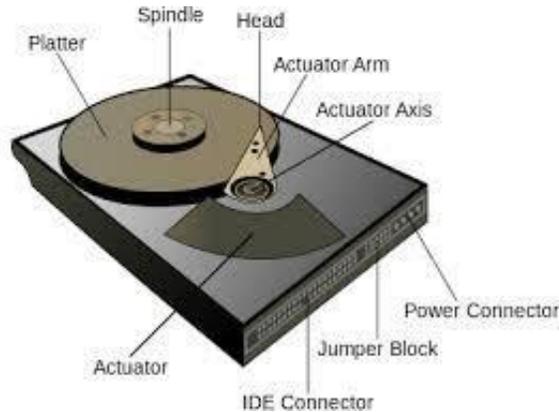
**Primary Storage:** Primary Storage is also known as main memory, is the storage area that is directly accessible by the CPU at very high speeds. Data has to be fed into the system before the actual processing starts. It is because the processing speed of Central Processing Unit (CPU) is so fast that the data has to be provided to CPU with the same speed. Therefore the data is first stored in the storage unit for faster access and processing.



Primary Storage (RAM)

This storage unit or the primary storage of the computer system is designed to do the above functionality. It provides space for storing data and instructions.

**Secondary Storage** : is also known as secondary memory or auxiliary memory. It basically overcomes all the drawbacks of the primary storage area. It is cheaper, non-volatile and used to permanently store data and programs that are not being currently executed by the CPU.



Secondary Storage (Hard Disk)

**3. Processing:** The process of performing operations on the data as per the instructions specified by the user (program) is called processing. Data and instructions are taken from the primary memory and transferred to the arithmetic and Logical Unit (ALU), which performs all sorts of calculations. The intermediate results of processing may be stored in the main memory, as they might be required again. When the processing completes, the final result is then transferred to the main memory.

**4. Output:** This is the process of producing results from the data for getting useful information. Similarly the output produced by the computer after processing must also be kept somewhere inside the computer before being given to you in human readable form. Again the output is also stored inside the computer for further processing.

Output devices include:



a) Monitors.

b) Printers.

c) Speakers

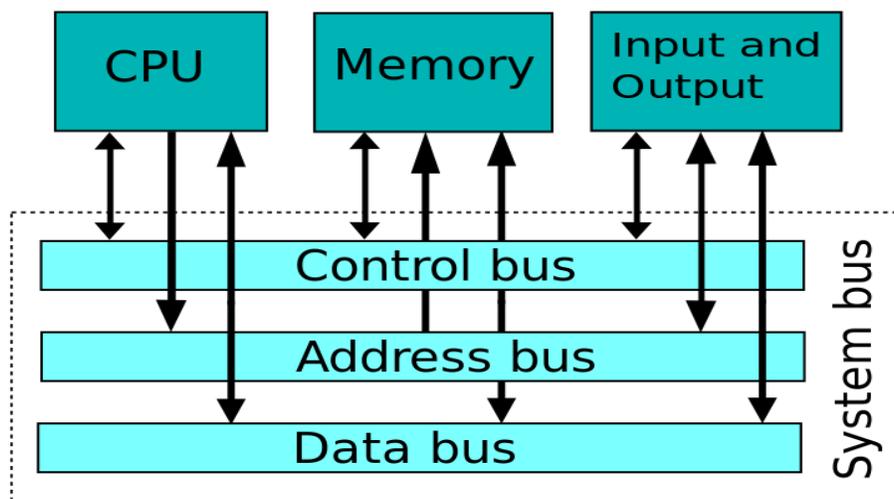
**5. Control unit:** The control unit (CU) is the central nervous system of the entire computer. It manages and controls all the components of the computer system. Controlling of all operations like input, processing and output are performed by control unit. It takes care of step by step processing of all operations inside the computer.

## Buses:

A bus is a collection of wires through which data is transmitted from one part of a computer to another. It connects physical components such as cables, printed circuits, CPU, memory, peripherals etc., for sharing of information and communication with one another.

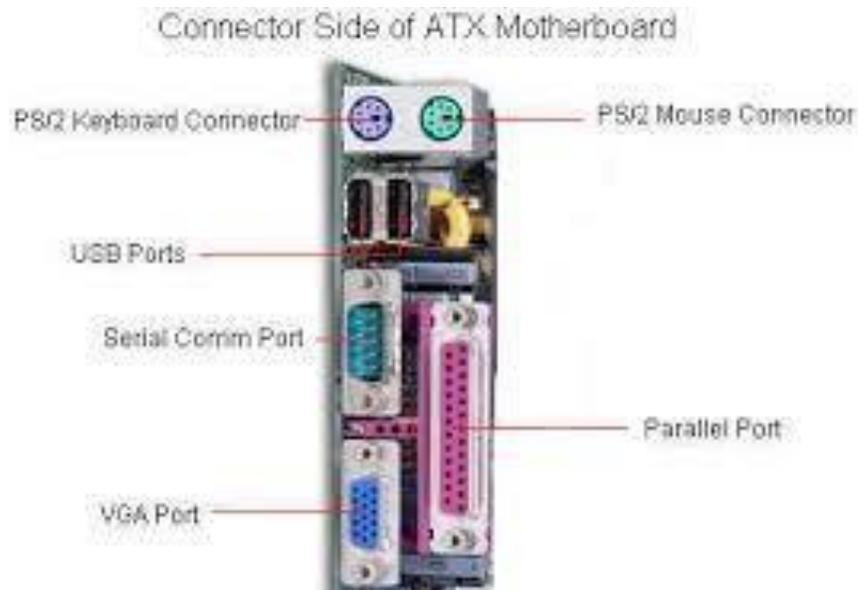
Types of buses:

1. Data bus: It is used to transfer the data between processor, memory and I/O devices. It is bidirectional in nature.
2. Address bus: It is used to transfer the addresses of data and instructions stored in memory. It is unidirectional in nature.
3. Control bus: It is used to transfer the control signals between CPU, memory and I/O devices. It is unidirectional or bidirectional in nature.

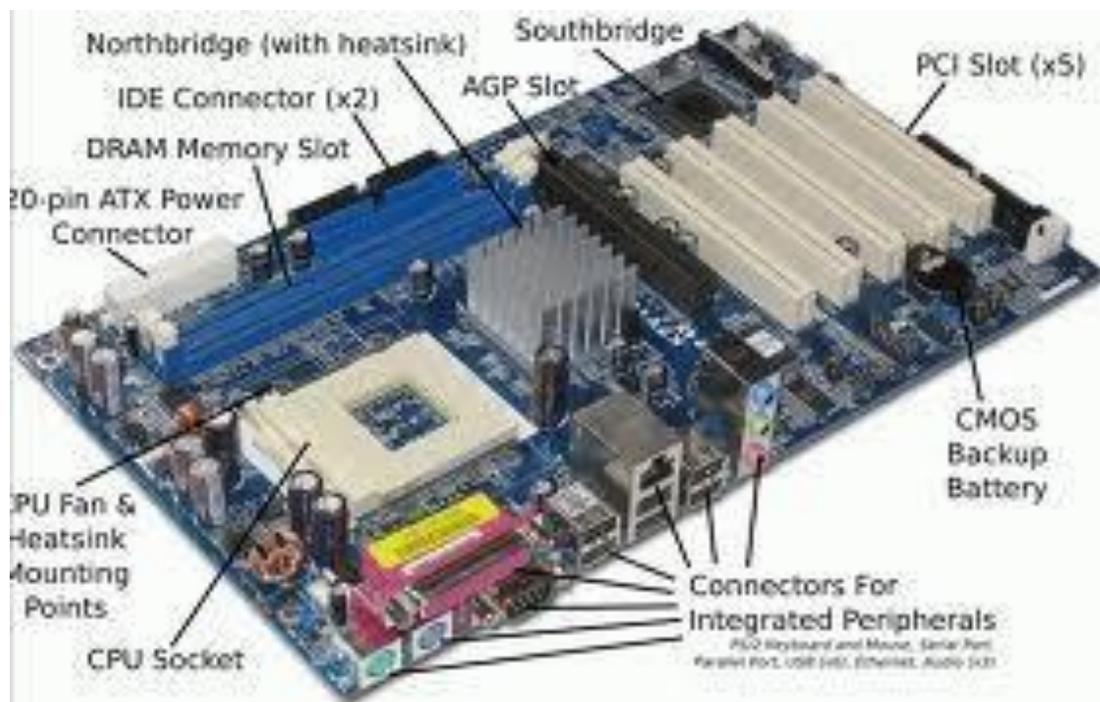


Types of Buses

**MOTHER BOARD:** The motherboard serves to connect all of the parts of a computer together. The CPU, memory, hard drives, optical drives, video card, sound card and other ports and expansion cards all connect to the motherboard directly or via cables.



Connector side of Motherboard



MOTHER BOARD

## Chip sets:

**Chip:** A small piece of semiconducting material (usually silicon) on which an integrated circuit is embedded. A typical chip is less than square inches and can contain millions of

electronic components (transistors). Computers consist of many chips placed on electronic boards called *printed circuit boards*.

There are different types of chips. For example, CPU chips (also called *microprocessors*) contain an entire processing unit, whereas memory chips contain blank memory.

### **Chip set:**

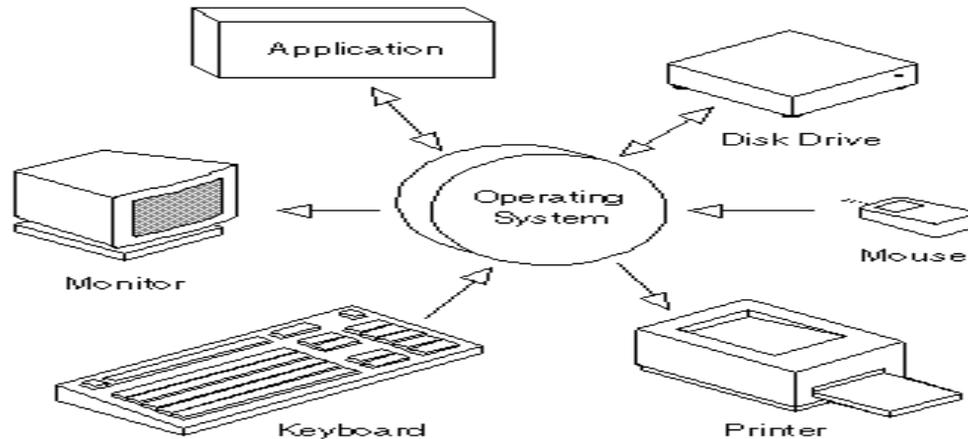
A number of integrated circuits designed to perform one or more related functions. For example, one chipset may provide the basic functions of a modem while another provides the CPU functions for a computer. Newer chipsets generally include functions provided by two or more older chipsets. In some cases, older chipsets that required two or more physical chips can be replaced with a chipset on one chip.



## **Operating System and Types of Operating System:**

The operating system is system software that controls and supervises the hardware components of a computer system and it provides services to computer users. Every general-purpose computer must have an operating system to run other programs. Operating systems perform basic tasks, such as recognizing input from the keyboard, sending output to the display screen, keeping

track of files and directories on the disk, and controlling peripheral devices such as disk drives and printers.



Operating System controls the hardware components of a computer system

The four general kinds of operating systems (OS) are categorized as:

- **Real-time OS:** This kind of OS controls machinery, industrial equipment and scientific instruments. Its main purpose is to ensure that an operation executes in exactly the same way, in the same amount of time, every time it happens.
- **Multuser:** A multiuser OS lets many people do many things, all at the same time. It has to balance the needs of each user and keep them separate, so that they don't interfere with each other.
- **Single user, single task:** This kind of OS is designed so that a computer executes a user's tasks one at a time, such as with early personal digital assistants.
- **Single-user, multitasking:** Most PCs use this kind of OS, such as Windows or Mac OS. It lets a single user do many things at once.

Operating systems provide a software platform on top of which other programs, called *application programs*, can run. The application programs must be written to run on top of a particular operating system. Your choice of operating system, therefore, determines to a great extent the applications you can run. For PCs, the most popular operating systems are DOS, OS/2, and Windows, but others are available, such as Linux.

As a user, you normally interact with the operating system through a set of commands. For example, the DOS operating system contains commands such as COPY and RENAME for copying files and changing the names of files, respectively. The commands are accepted and executed by a part of the operating system called the command processor or command line interpreter. Graphical user interfaces allow you to enter commands by pointing and clicking at objects that appear on the screen.

## Basics of Networking:

A network is a group of two or more computer systems linked together. A network is a set of technologies including hardware, software and media that can be used to connect computers together, enabling them to communicate, exchange information and share resources in real time.

### Benefits of a Network:

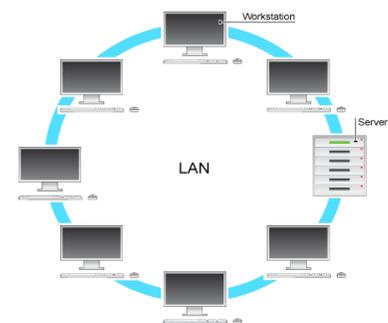
- Allows simultaneous access to critical programs and data.
- Allows people to share peripheral devices, such as printers and scanners.
- Streamlines personal communication with email.
- Makes the backup process easier.

### Network Setup: There are 3 types of networks

- LAN (Local Area Network)
- WAN (Wide Area Network)
- MAN (Metropolitan Area Network)

### Local Area Network (LANs)

A network of computers located relatively near each other and connected by a cable. A LAN is a data communication system consisting of several devices such as computers and printers. A LAN can consist of just two or three PCs connected together to share resources, or it can include hundreds of computers of different kinds. Any network that exists within a single building, or even a group of adjacent buildings, is considered a LAN.



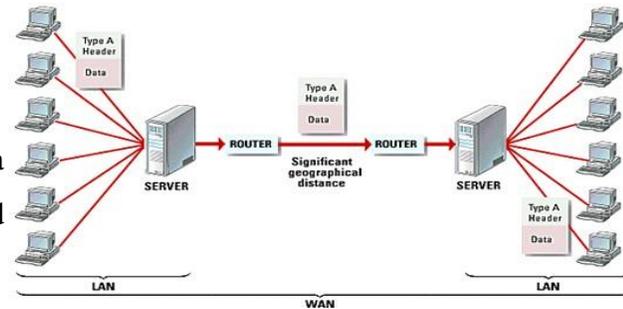
Local Area Network

For example, two departments located on the same floor of a building may have their own separate LANs, but if the departments need to share data, then they can create a link between the two LANs.

### **Wide Area Network (WANs)**

Two or more LANs connected together, generally across a wide geographical area using high-speed or dedicated telephone line. For example, a company may have its corporate headquarters and manufacturing plant in one city

and its marketing office in another. Each site needs resources, data, and programs locally, but it also needs to share data with the other sites. To accomplish this feat of data communication, the company can attach devices that connect over public utilities to create a WAN. These remote LANs are connected through a telecommunication network or via the internet through an Internet Service Provider (ISP) that contracts with the telecommunication networks to gain access to the Internet's backbone.



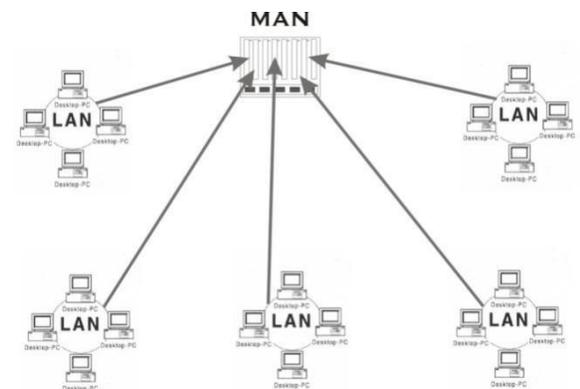
Wide Area Network

### **Metropolitan Area Networks (MANs)**

The Metropolitan Area Network (MAN) is a large-scale network that connects multiple corporate LANs together.

MANs usually

are not owned by a single organization; their communication devices and equipment are usually maintained by a group or single network provider that sells its networking services to corporate customers. Similar to a WAN network but is confined to a single city or metropolitan area.



Metropolitan Area

**Topology** - The physical layout of the cables that connect the nodes of the network.

### **Network Topologies:**

- Bus topology
- Star topology
- Ring topology
- Mesh topology

- Hybrid topology

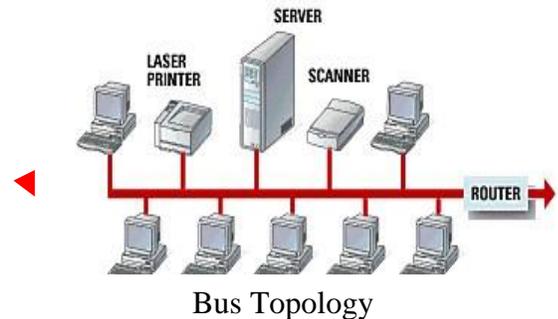
**Bus topology** - In this network structure, a single cable runs in a building or campus. All the nodes (terminals/computers) are connected to this single cable. It is suitable for LAN.

Advantages:

- Failure of one node will not affect the whole network
- Well suited for quick setup
- Easy to install and expand
- High rate of data transmission as compared to star and ring topology

Disadvantages :

- A cable break can disable the entire network
- Troubleshooting is very difficult
- Only a single message can travel at a time



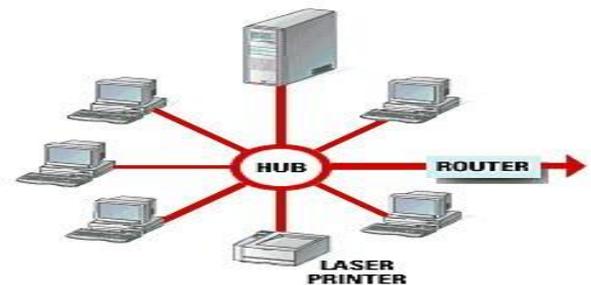
**Star topology** - In this network structure, all the computers are connected with a centralized system called server. The central computer is also called a hub. To transmit information from one node to another node, it should be transmitted through a central hub. The central hub manages and controls all the functions of network.

Advantages :

- Easy to install and expand
- Addition or deletion of a node is easier
- Failure of one node will not affect the entire network
- Well suited for quick setup
- Easier to debug network problems through a hub

Disadvantages :

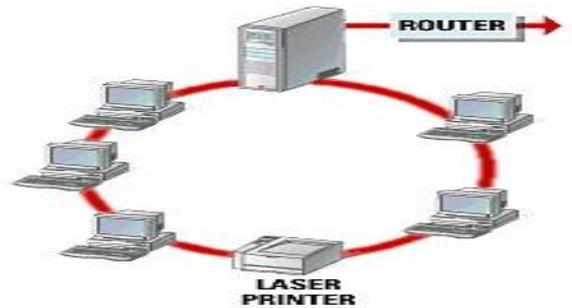
- Failure of the hub will affect the whole network
- Cost of hub is expensive



**Ring topology** - In this network structure, all the computers are connected to each other in the form of a ring. It connects the nodes of the network in a circular chain in which each node is connected to the next.

Advantages :

- All the nodes have equal chance to transfer the data
- These are easily extensible
- It can span longer distance than other types of networks



Ring Topology

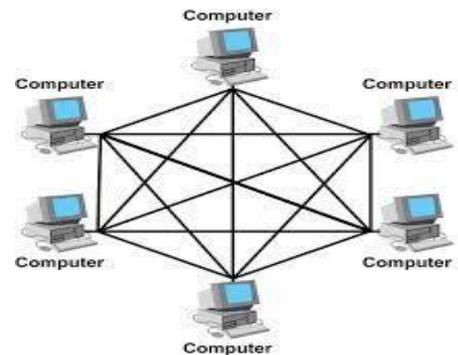
Disadvantages:

- Difficult to install
- Difficult to troubleshoot
- Adding or removing computer can disturb the entire network

**Mesh Topology :** In this network structure, all the computers and network devices are interconnected with one another like a mesh. Every node has a connection to every other node in the network.

Advantages :

- Failure of a single node will not affect the entire network
- Data transfer rate is very fast because all the nodes are connected to each other

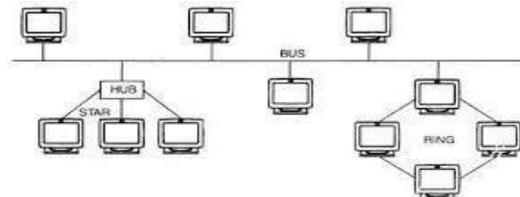


Ring Topology

Disadvantages :

- Installation and re configuration is very difficult

**Hybrid Topology :** This topology uses a combination of any two or more topologies in such a that the resulting network does not exhibit one of the standard topologies( Ex. bus, star,ring, etc.)



Hybrid Topology

### Network Interface Cards (NIC) :

A network interface card is used to connect a machine to an computer network. This card provides an interface to the media. This may be either using an external transceiver (as shown) or through an internal integrated transceiver mounted on the network interface card PCB. The card usually also contains the protocol control

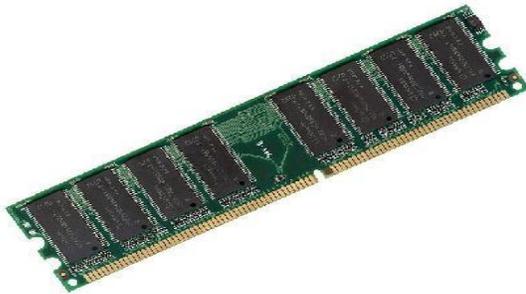


Network Interface Card

firmware and Ethernet Controller needed to support the Medium Access Control (MAC) data link protocol used by Ethernet.

Write-up on RAM, SDRAM, FLASH memory, Hard disks, Optical media, CD-ROM/R/RW, DVDs, Flash drives, Keyboard, Mouse, Printers and Plotters.

**Random Access Memory (RAM) :**Random Access Memory is a volatile storage area within the computer that is typically used to store data temporarily. The information stored in the RAM is basically loaded from the computer's hard disk, and includes data related to the operating system and applications that are currently being executed by the processor.



RAM fixed on the motherboard

### Types of RAM

There are two different types of RAM:

- DRAM (Dynamic Random Access Memory)
- SRAM (Static Random Access Memory).

### **Static RAM**

In RAM electrical signals are sent to individual memory locations on the RAM chip for future reading by the processor. If RAM is volatile that means that electricity must be supplied to the chip at all times in order to keep the information. Static RAM keeps this information in a series of transistors which remember the electrical signal given to them with just a standard power signal. Because of the complexity of this type of RAM it is usually more expensive, but there is less chance of information loss. The other great benefit of this type of RAM is its speed. Because of its higher cost but greater speed, the most common usage for this type of RAM is in something called the L2 cache. This is RAM that is kept in the processor itself for near instantaneous access to the memory, and it doesn't have to be large so the cost is minimal.

### **Dynamic RAM**

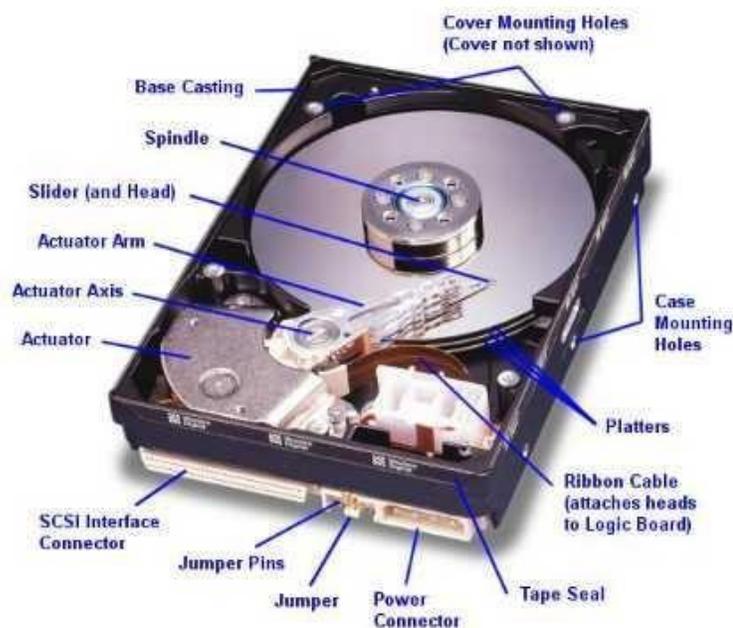
Dynamic RAM is different from static RAM in that it needs to be refreshed. It uses small capacitors to hold information, and these capacitors will drain out if not refreshed (electricity reapplied) about every 15 nanoseconds. This slight loss in speed is made up for by the reduced cost of this type of RAM. It is the most common form of RAM used in computers today. The most used type is DDR2. DDR2 is the successor to DDR SDRAM (Double Data Rate synchronous Dynamic RAM). What DDR did was to increasing the timing accuracy of RAM synchronizing with the processor so that greater speeds could be achieved. DDR2 increased on this principle and double the speed again. The current cutting edge computer RAM is DDR3. This managed to double even the speed of DDR2. Unfortunately none of the DDR RAMs are compatible with each other and can't replace each other in a computer unless specifically designed for it.

**Synchronous Dynamic Random Access Memory (SDRAM):** It is an improvement standard DRAM because it retrieves data alternatively between two sets of memory. This eliminates the delay caused when one bank of memory address is shut down while another is prepared for reading. It is also called synchronous DRAM because the memory synchronous with the clock speed that the computer's CPU bus speeds optimized for. The faster the bus speed, the faster the SDRAM can be. SDRAM speed is measured in Megahertz.

## Flash Memory

Flash memory is a type of Electrically Erasable Programmable Read Only Memory (EEPROM). The name comes from how the memory is designed. A section of memory cells can be erased in a single action or in a "flash". Flash memory cards are used for digital cameras, cellular phones, networking hardware and PC cards.

**Hard disks:** A hard disk is part of a unit, often called a "disk drive," "hard drive," or "hard disk drive," that stores and provides relatively quick access to large amounts of data on an electromagnetically charged surface or set of surfaces. Today's computers typically come with a hard disk that contains several billion bytes (gigabytes) of storage.



Hard disk

## Optical Media:

Optical media - such as the compact disk (CD) - are storage media that hold content in digital form and that are written and read by a laser; these media include all the various CD and DVD variations, as well as optical jukeboxes and auto-changers. Ex: CD-ROM, DVD etc.



**CD-ROM/R/RW:**

Stands for "Compact Disc Read-Only Memory." A CD-ROM is a CD that can be read by a computer with an optical drive. The "ROM" part of the term means the data on the disc is "read-only," or cannot be altered or erased. Because of this feature and their large capacity, CD-ROMs are a great media format for retail software. The first CD-ROMs could hold about 600 MB of data, but now they can hold up to 700 MB. CD-ROMs share the same technology as audio CDs, but they are formatted differently, allowing them to store many types of data.



- **CD-R:** A blank, *recordable* CD on which you can record information *once*.
- **CD-ROM:** A ex-CD-R that now has information on it. The ROM stands for Read-Only Memory, meaning any CD player can read (play) the information on the CD. But no machine in the world can erase or change the contents of the CD-ROM. It's contents are permanent.

**DVD:** (*digital versatile disc or digital video disc*)

*It is a* A type of optical disk technology similar to the CD-ROM. A DVD holds a minimum of 4.7 GB of data, enough for a full-length movie. DVDs are commonly used as a medium for digital representation of movies and other multimedia presentations that combine sound with graphics.

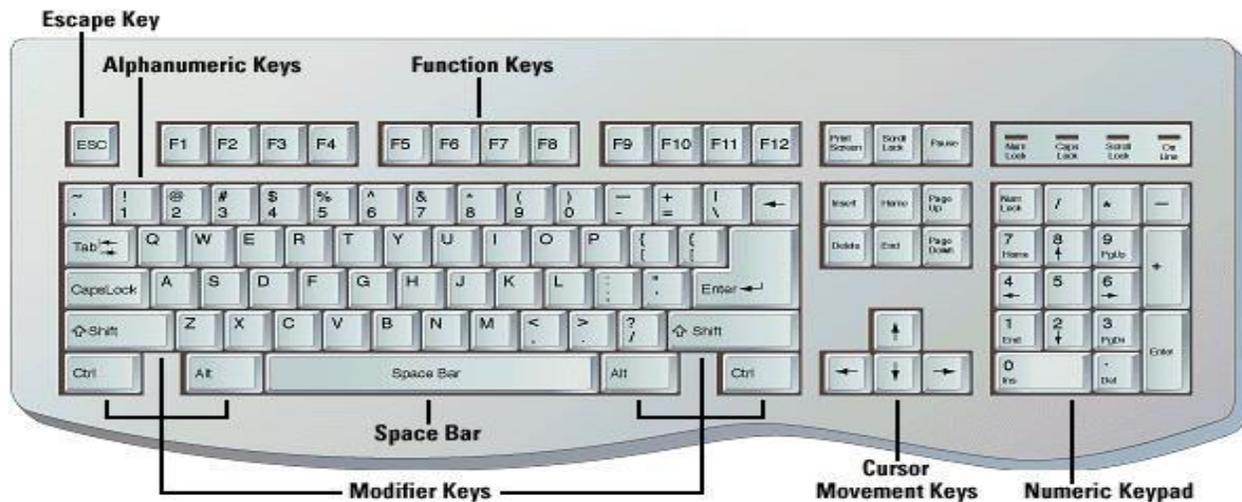
The DVD specification supports disks with capacities of from 4.7GB to 17GB and access rates of 600KBps to 1.3 MBPS. One of the best features of DVD drives is that they are backward-compatible with CD-ROMs, meaning they can play old CD-ROMs, CD-I disks, and video CDs, as well as new DVD-ROMs. Newer DVD players can also read CD-R disks.

**Flash drives:**

USB flash drives are removable, rewritable and physically much smaller drives weighing even less than 30g. A flash drive consists of a small printed circuit board carrying the circuit elements and a USB connector.



**Keyboard:** A keyboard is a primary input device used in all computers. Keyboard has a group of switches resembling the keys on the ordinary typewriter machine. Normally keyboard has around 101 keys. The keyboard includes key that allows us to type letters, numbers and various special symbols such as \*,/,[,% etc.



**Mouse:** The mouse is the key input device to be used in a Graphical User Interface (GUI). The users can use the mouse to handle the cursor pointer easily on the screen to perform various functions like opening a program or file. With mouse, the users no longer need to memorize commands, which was earlier a necessity when working with text-based command line environment such as MS-DOS.



**Printers:** The printer is an output device, which is used to get hard copy of the text displayed on the screen. The printer is an external optional device that is connected to the computer system using cables. The printer driver software is required to make the printer working. The performance of a printer is measured in terms of Dots Per Inch (DPI) and Pages Per Minute (PPM) produced by the printer.

### Types of printers

1. Impact Printers : Impact printers are those printers in which a physical contact is established between the print head, ribbon(cartridge) and paper. Ex. Dot Matrix Printer
2. Non-Impact Printers : No physical contact is established between the print head, ribbon and paper. Ex. Inkjet printer and laser printer.



Dot matrix Printer



Inkjet Printer



Laser Printer

**Plotters:** A plotter is similar to printer that produces hard-copy output with high-quality color graphics. Plotters are generally more expensive than printers, ranging from about \$1000 to \$75000.



### Problem Solving Techniques:

There are three approaches to solve a given problem :

- Algorithm
- Flowchart
- Pseudo code

**Algorithm:** An algorithm is a step-by-step to be followed in solving a problem. It provides a scheme to solve a particular problem in finite number of unambiguous steps.

### Features of an algorithm :

- **Sequence** : Each step of the algorithm is executed in the specified order
- **Decision** : Decision statements are used when the outcome of the process depends on some condition
- **Repetition** : Executing one or more steps for a number of times

**Example : To compute Sum of two numbers**

**Algorithm: Sum\_Two\_numbers**

Step 1: [Initialize]

Start

Step 2: [Input two numbers]

Read number1, number2

Step 3: [Compute sum of two numbers]

Sum  $\leftarrow$  number1 + number2

Step4: [Display the sum]

Print Sum

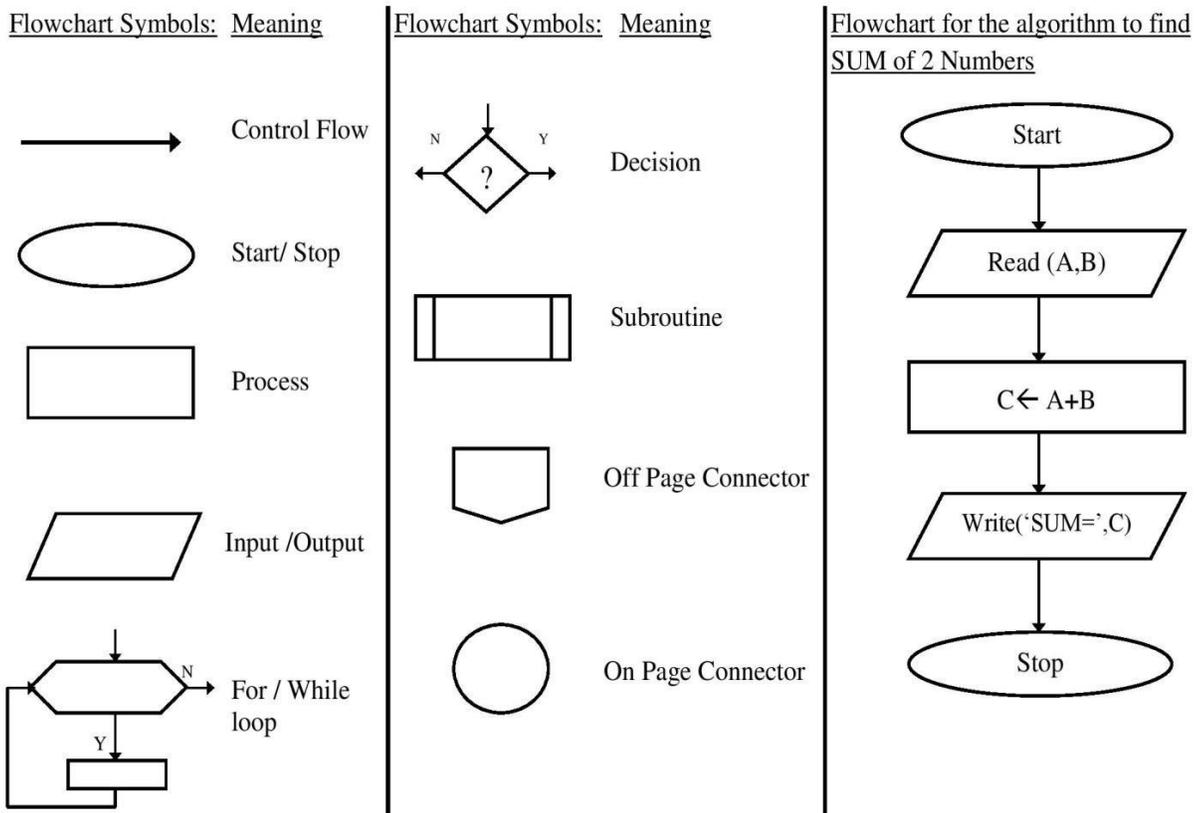
Step5: [Finished]

Stop

**Flowcharts :** A flowchart is a graphical or symbolic representation of an algorithm. They are basically used to design and develop complex programs to help the users to visualize the logic of the program so that they can gain a better understanding of the program and find flaws, bottlenecks, and other less-obvious features within it. Basically, a flowchart depicts the “flow” of a program. The following table shows the symbols used in flowchart along with its descriptions.

➤ What is Flowchart?

A Flowchart is a pictorial representation of an algorithm.



**Pseudo code:** It is a form of structured English that describes algorithms. Pseudocode is a compact and informal high-level description of an algorithm that uses the structural conventions of a programming language.

E.g.: To compute sum of two numbers

Input number1, number2

Sum=number1+number2

Print Sum

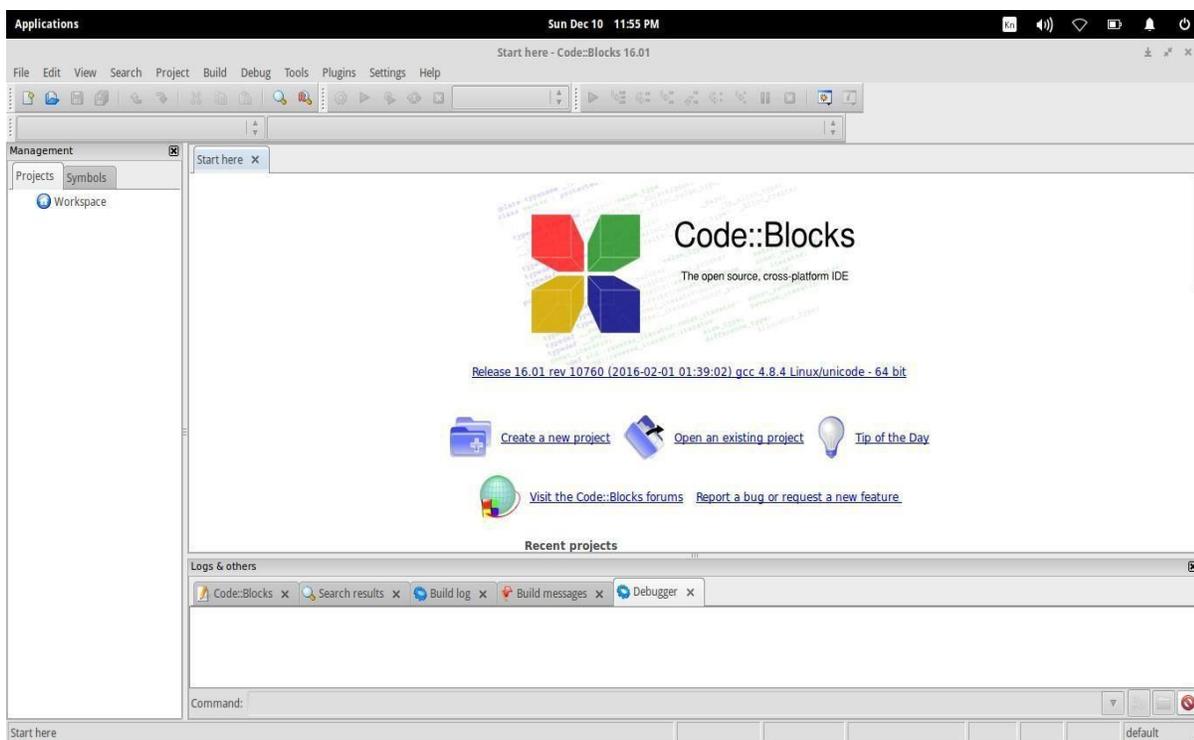
## Laboratory Session-2

### PART-A

**Familiarization with computer hardware and programming environment, concept of naming the program files, storing, compilation, execution and debugging. Taking any simple C- code.**

**Purpose:** This program demonstrates naming and saving program files, compilation, execution and debugging of source file.

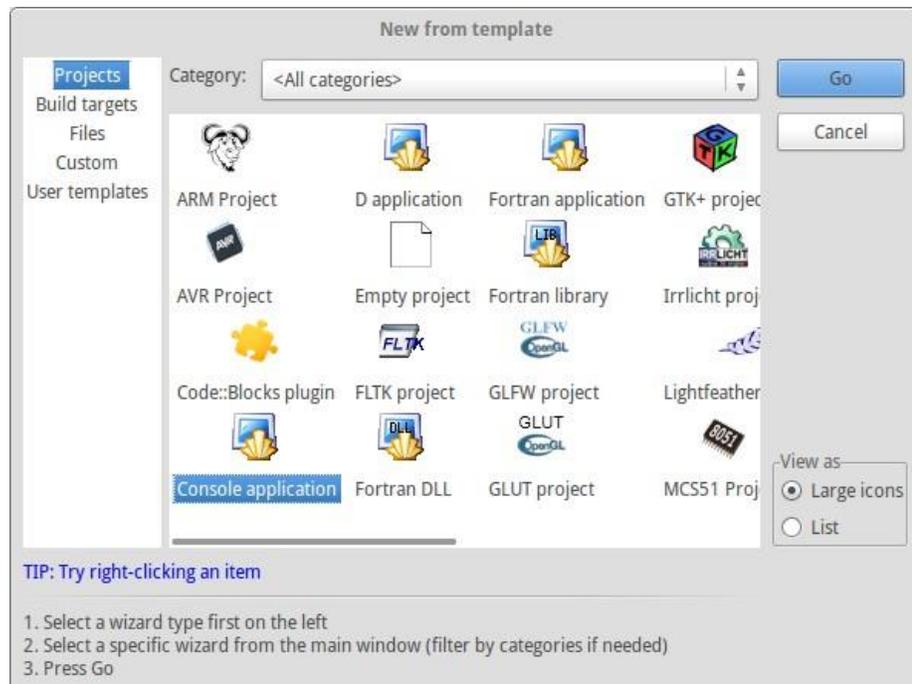
To launch Code::Blocks IDE, Click on You will get a window as shown below



Code::Blocks creates what is called a **Workspace** to keep track of the project you are working on. It is possible for you to be working on multiple projects within your workspace. A project is a collection of one or more source (as well as header) files. Source files are the files that contain the source code for your program. If you are developing a C program, you are writing C source code (.c files).

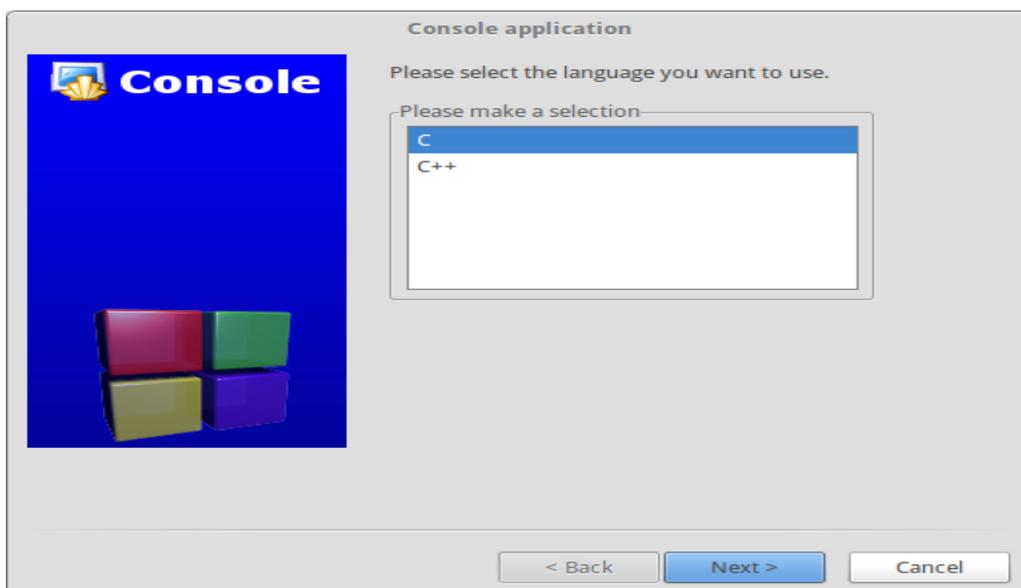
First start a new Project by clicking on **Create a new project**

**OR** To create a project, click on the **File** pull-down menu, open **New** and then **Project**. You get this pop-up window

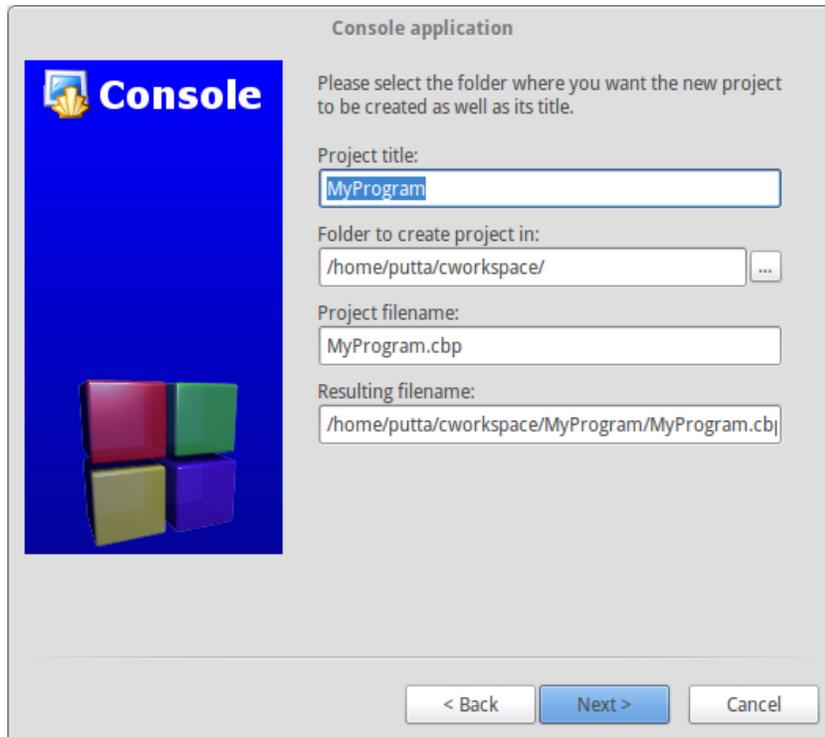


Choose Console application

Then in the next window select the programming language C (and not C++)



In the next step give the Project Title and specify the folder where you want to save your project. (**Note** : Dont use any special characters or whitespaces for project title and folder names)



**Console application**

Please select the folder where you want the new project to be created as well as its title.

Project title:  
MyProgram

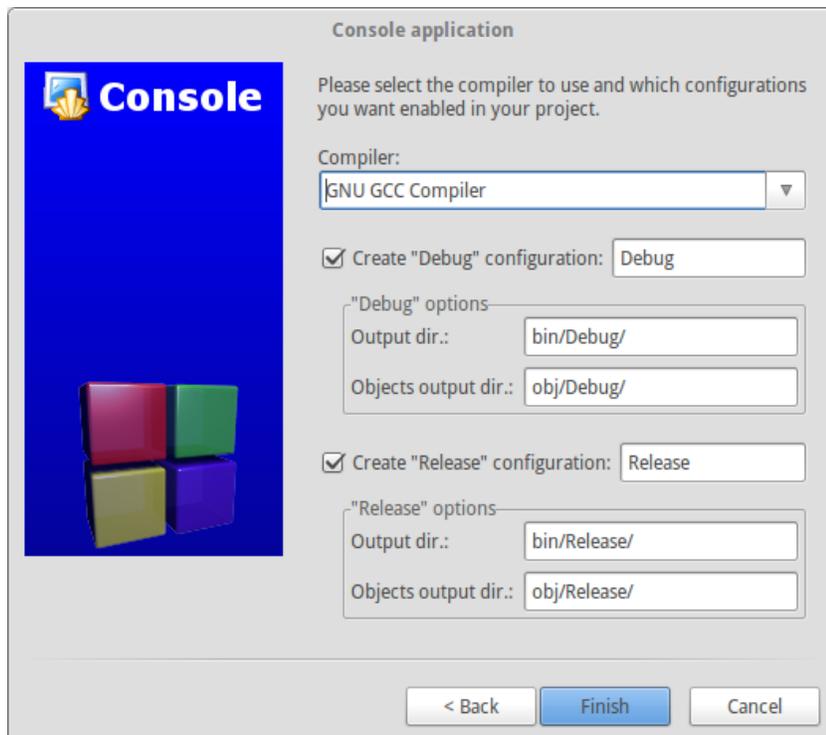
Folder to create project in:  
/home/putta/cworkspace/ ...

Project filename:  
MyProgram.cbp

Resulting filename:  
/home/putta/cworkspace/MyProgram/MyProgram.cbp

< Back   Next >   Cancel

Finally you will be prompted to choose the compiler. Just choose the default options here (Dont change the options). You should be using GNU GCC Compiler. Click Finish to create the new project.



**Console application**

Please select the compiler to use and which configurations you want enabled in your project.

Compiler:  
GNU GCC Compiler

Create "Debug" configuration: Debug

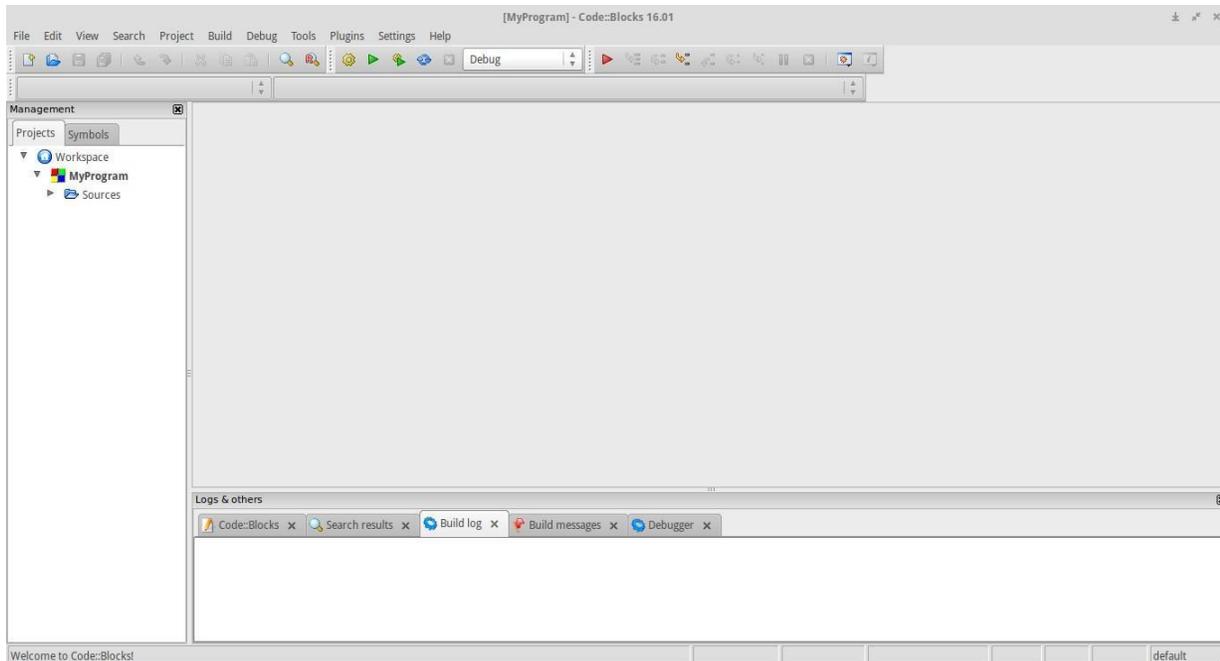
"Debug" options  
Output dir.: bin/Debug/  
Objects output dir.: obj/Debug/

Create "Release" configuration: Release

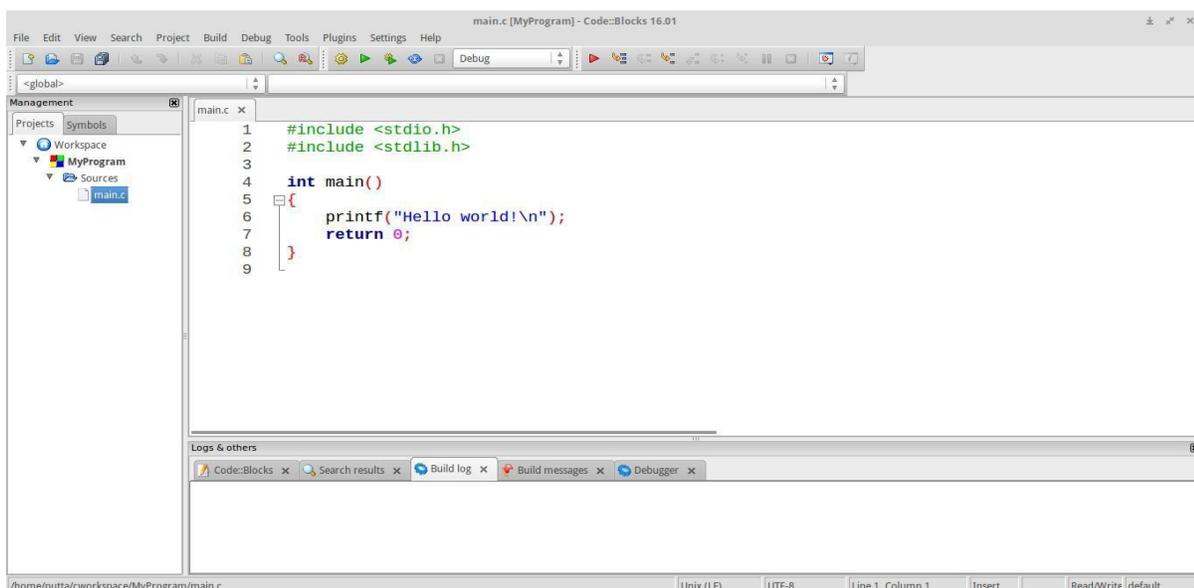
"Release" options  
Output dir.: bin/Release/  
Objects output dir.: obj/Release/

< Back   Finish   Cancel

The system will then return to the [MyProgram] window and you are ready to write your program. In the Management area of the screen (Shift-F2 toggles the Management display), you will see the files that are part of the project in the Projects tab. To see the source files, click on the triangle symbol to expand the Workspace and its subdirectories.

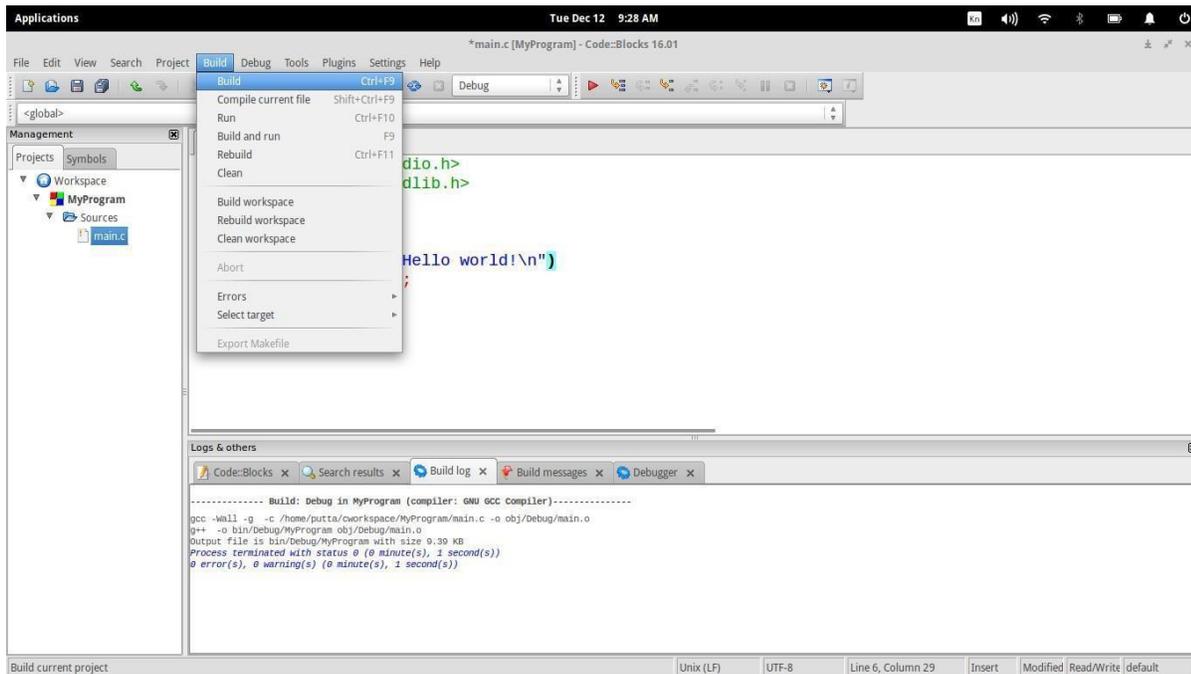


Under Sources, there is a file called **main.c**, which is automatically created for you when you build a console application. Click on **main.c**.



main.c contains a simple Hello World program which you can edit later to solve a programming problem. Now let us see how to compile and execute this main.c program. Just to understand the process of debugging we knowingly introduce an error in the program by removing the semicolon after the printf statement. We will now compile the program (To compile a file means to take the instructions that you have written and translate it into machine code for the computer to understand).

Compile the project from the Build pull-down menu by clicking on **Build** option[**Ctrl+F9**].



The error messages are shown in the Build messages window at the bottom. Let us now try to understand these error messages.

```
==== Build: Debug in MyProgram (compiler: GNU GCC Compiler) ====
main.c In function main: main.c 7 error: expected ; before return
main.c 8 warning: control reaches end of non-void function [-Wreturn-type]
==== Build failed: 1 error(s), 1 warning(s) (0 minute(s), 1 second(s)) ====
```

```

main.c [MyProgram] - Code::Blocks 16.01
File Edit View Search Project Build Debug Tools Plugins Settings Help
<global>
main(): int
Management
Projects Symbols
Workspace
MyProgram
Sources
main.c
main.c x
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main()
5 {
6     printf("Hello world!\\n")
7     return 0;
8 }
9
Logs & others
Code::Blocks x Search results x Build log x Build messages x Debugger x
File Line Message
==== Build: Debug in MyProgram (compiler: GNU GCC compiler) ====
/home/putta/cwork... In function 'main':
/home/putta/cwork... 7 error: expected ';' before 'return'
/home/putta/cwork... 8 warning: control reaches end of non-void function [-Wreturn-type]
==== Build Failed: 1 error(s), 1 warning(s) (0 minute(s), 1 second(s)) ====
/home/putta/cworkspace/MyProgram/main.c Unix (LF) UTF-8 Line 7, Column 1 Insert Read/Write default

```

The error messages show the errors in syntax (and not the logical errors). It also indicates the line number and the type of error. Here in this example the error says that before the return statement in line no 7 a semicolon is missing. The next message is a warning message which has resulted because of the previous error.

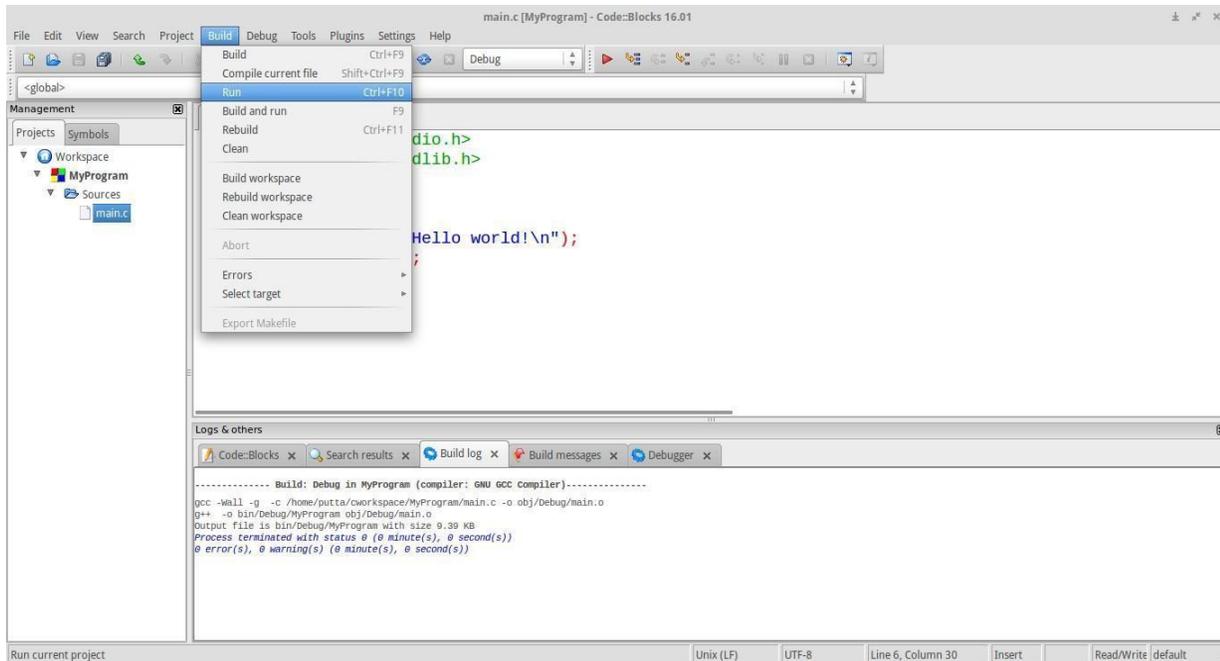
Now go to line number 6 and add a semicolon at the end. Now build your project again.

```

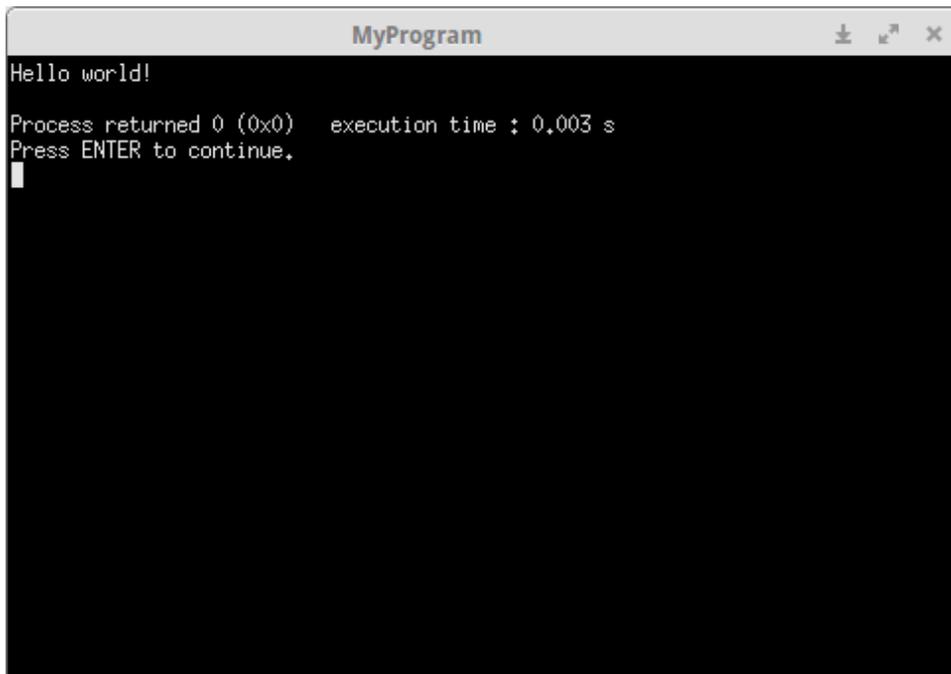
main.c [MyProgram] - Code::Blocks 16.01
File Edit View Search Project Build Debug Tools Plugins Settings Help
<global>
main(): int
Management
Projects Symbols
Workspace
MyProgram
Sources
main.c
main.c x
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main()
5 {
6     printf("Hello world!\\n");
7     return 0;
8 }
9
Logs & others
Code::Blocks x Search results x Build log x Build messages x Debugger x
..... Build: Debug in MyProgram (compiler: GNU GCC compiler).....
gcc -Wall -g -c /home/putta/cworkspace/MyProgram/main.c -o obj/Debug/main.o
g++ -o bin/Debug/MyProgram obj/Debug/main.o
Output file is bin/Debug/MyProgram with size 9.39 KB
Process terminated with status 0 (0 minute(s), 0 second(s))
0 error(s), 0 warning(s) (0 minute(s), 0 second(s))
/home/putta/cworkspace/MyProgram/main.c Unix (LF) UTF-8 Line 6, Column 30 Insert Read/Write default

```

Now the build message window shows the following message. 0 error(s), 0 warning(s) (0 minute(s), 0 second(s)) This means that the errors and warnings have been successfully resolved. Now it is time to run the program. You can Execute the project from the Build pull-down menu by clicking on **Run** option[**Ctrl+F10**].



An output window pops displaying the output of the program. A greeting message Hello world! is printed on to the output console.





**COMPUTER PROGRAMMING LABORATORY**  
**[As Per Choice Based Credit System (CBCS) System]**  
**(Effective from the academic year 2018 -2019)**

**SEMESTER – I/II**

<b>Subject Code</b>	18CPL17/27	<b>CIE Marks</b>	40
<b>Number of Lecture Hours/Week</b>	2	<b>SEE Marks</b>	60
<b>Total Number of Lab Hours</b>	32	<b>Exam Hours</b>	3 Hrs

**Credits – 1**

**Course Learning Objectives :**

This course (18CPL17/27) will enable students to:

- Write flowcharts, algorithms and programs.
- Familiarize the processes of debugging and execution.
- Implement basics of C programming language.
- Illustrate solutions to the laboratory programs.

**Descriptions (if any):**

- The laboratory should be preceded or followed by a tutorial to explain the approach or algorithm being implemented or implemented for the problems given.
- Note that experiment 1 is mandatory and written in the journal.
- Questions related with experiment 1, need to be asked during viva-voce for all experiments.
- Every experiment should have algorithm and flowchart be written before writing the program.
- Code should be traced using minimum two test cases which should be recorded.
- It is preferred to implement using Linux and GCC.

**Laboratory Programs:**

1	Familiarization with programming environment, concept of naming the program files, storing, compilation, execution and debugging. Taking any simple C- code.
---	--

**PART - A**

2	Develop a program to solve simple computational problems using arithmetic expressions and use of each operator leading to simulation of a Commercial calculator. (No built-in math function)
3	Develop a program to compute the roots of a quadratic equation by accepting the coefficients. Print appropriate messages.
4	Develop a program to find the reverse of a positive integer and check for palindrome or not. Display appropriate messages.
5	An electricity board charges the following rates for the use of electricity: for the first 200 units 80 paise per unit: for the next 100 units 90 paise per unit: beyond 300 units Rs 1 per unit. All users are charged a minimum of Rs. 100 as meter charge. If the total amount is more than Rs. 400, then an additional surcharge of 15% of total amount is charged. Write a program to read the name of the user, number of units consumed and print out the charges.
6	Introduce 1D Array manipulation and implement Binary search.
7	Implement using functions to check whether the given number is prime and display appropriate messages. (No built-in math function)

**PART - B**

8	Develop a program to introduce 2D Array manipulation and implement Matrix multiplication and ensure the rules of multiplication are checked.
9	Develop a Program to compute Sin(x) using Taylor series approximation. Compare your result with the built-in Library function. Print both the results with appropriate messages.
10	Write functions to implement string operations such as compare, concatenate, string length. Convince the parameter passing techniques.

11	Develop a program to sort the given set of N numbers using Bubble sort.
12	Develop a program to find the square root of a given number N and execute for all possible inputs with appropriate messages. Note: Don't use library function sqrt(n).
13	Implement structures to read, write and compute average marks and the students scoring above and below the average marks for a class of N students.
14	Develop a program using pointers to compute the sum, mean and standard deviation of all elements stored in an array of n real numbers.
15	Implement recursive functions for Binary to Decimal Conversion.
<b>Laboratory Outcomes:</b> The student should be able to:	
<ul style="list-style-type: none"> <li>• Write algorithms, flowcharts and program for simple problems.</li> <li>• Correct syntax and logical errors to execute a program.</li> <li>• Write iterative and wherever possible recursive programs</li> <li>• Demonstrate use of functions, arrays, strings and structures in problem solving.</li> </ul>	
<b>Conduct of Practical Examination:</b>	
<ul style="list-style-type: none"> <li>• All laboratory experiments, excluding the first, are to be included for practical examination.</li> <li>• Experiment distribution: <ul style="list-style-type: none"> <li>○ For questions having only one part: Students are allowed to pick one experiment from the lot and are given equal opportunity.</li> <li>○ For questions having part A and B: Students are allowed to pick one experiment from part A and one experiment from part B and are given equal opportunity.</li> </ul> </li> <li>• Strictly follow the instructions as printed on the cover page of the answer script for breakup of marks.</li> <li>• Change of experiment is allowed only once and marks allotted for procedure part to be made zero.</li> <li>• Marks Distribution (Subjected to change in accordance with university regulations) <ul style="list-style-type: none"> <li>a). For questions having only one part – Procedure + Execution + Viva-Voce: 15+ 70 + 15 = 100 Marks</li> <li>b). For questions having part A and B <ul style="list-style-type: none"> <li>i. Part A – Procedure + Execution + Viva = 4 + 21 + 5 = 30 Marks</li> <li>ii. Part B – Procedure + Execution + Viva = 10 + 49 + 11 = 70 Marks</li> </ul> </li> </ul> </li> </ul>	

---

Program 2

---

**Develop a program to solve simple computational problems using arithmetic expressions and use of each operator leading to simulation of a commercial calculator. (No built-in math function)**

**ALGORITHM**

Step1. [Initialize] Start

Step2. [Input the values of two operands and operator]

read num1,op,num2

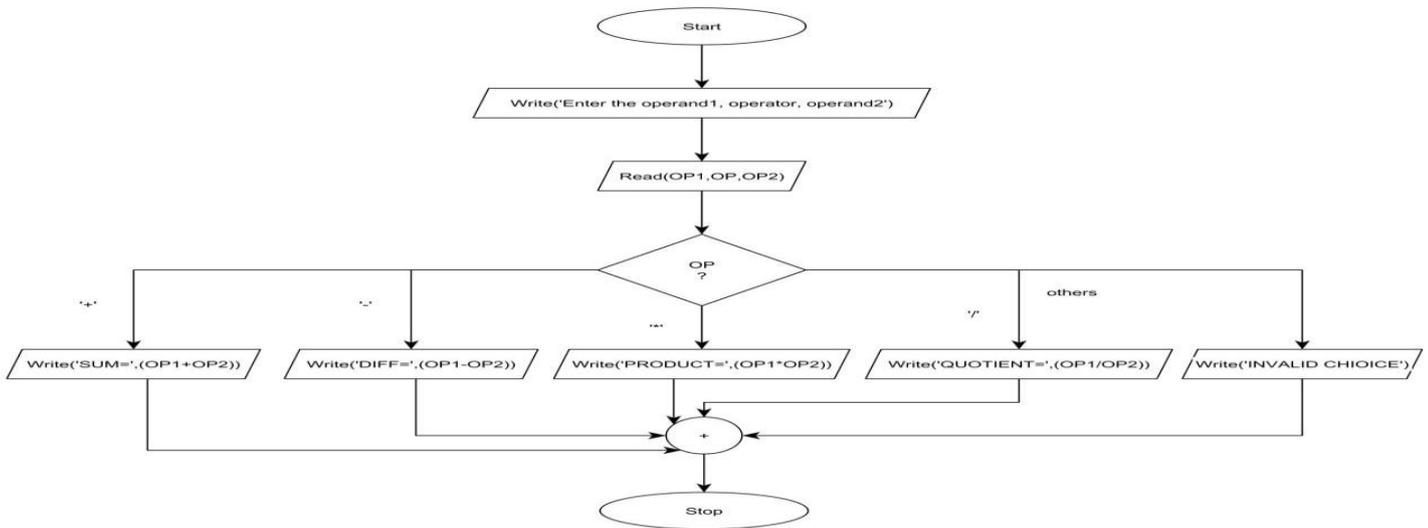
Step3. [Check the operator read] Switch (ch)

Case '+' then Print(sum: num1 + num2 )

Case '-' then Print(Sub: num1 -num2 )  
 Case '\*' then Print(Mul:num1 \* num2 )  
 Case '/' then Print(Quo: num1 / num2)  
 Otherwise Print "Invalid Choice" :

Step 4: [Finished] Stop

2. Flowchart for simulating simple commercial calculator



**/\* Program to design simple commercial calculator \*/**

```

#include <stdio.h> void
main()
{
int op1,op2;
char op;
printf("Enter Arithmetic expression\n");
scanf("%d%c%d",&op1,&op,&op2);
switch(op)
{
case '+': printf("sum=%d\n",(op1+op2));
break;
case '-': printf("difference=%d\n",(op1-op2));
break;
case '*': printf("product=%d\n",(op1*op2));
break;
case '/': printf("quotient=%d\n",(op1/op2));
break;
case '%': printf("remainder=%d\n",(op1%op2));
break;

default:
printf("Invalid Operator\n");
}
return 0;
}

```

Input-Output

**Sample Output :**

Enter Arithmetic expression 1+2  
Sum=3

Enter Arithmetic expression 1-2  
Subtraction=-1

Enter Arithmetic expression

1\*2  
product=2

Enter Arithmetic expression 1/2  
Quotient=0

Enter Arithmetic expression 1%2  
Remainder=1

Enter Arithmetic expression 1>2  
Invalid Operator

---

---

Program 3

---

---

**Develop a program to compute the roots of a quadratic equation by accepting the coefficients. Print appropriate messages.**

**ALGORITHM**

Step 1: [Start of the algorithm]

Start

Step 2: [Read the coefficients]

Read non zero coefficients a, b,

c Step 3: [calculate the discriminant]

$d \leftarrow b^2 - 4ac$

Step 4: [check if roots are real and equal]

if (d=0)

$x_1 \leftarrow -$

$b/(2a) \quad x_2 \leftarrow -b/$

$(2a)$

Print "Roots are equal"

Print x1,x2

Go to step 7

Step 5: [check if roots are real and

distinct] If(d>0)

$x_1 \leftarrow (-b + \sqrt{d}) / (2a)$

$x_2 \leftarrow (-b - \sqrt{d}) / (2a)$

Print "Roots are real and distinct"

Print  $x_1, x_2$  Go to step 7

Step 6: [check if roots are imaginary] If( $d < 0$ )

$$x_1 \leftarrow -b/(2*a)$$

$$x_2 \leftarrow \sqrt{\text{fabs}(d)/(2*a)}$$

Print “ The roots are complex”

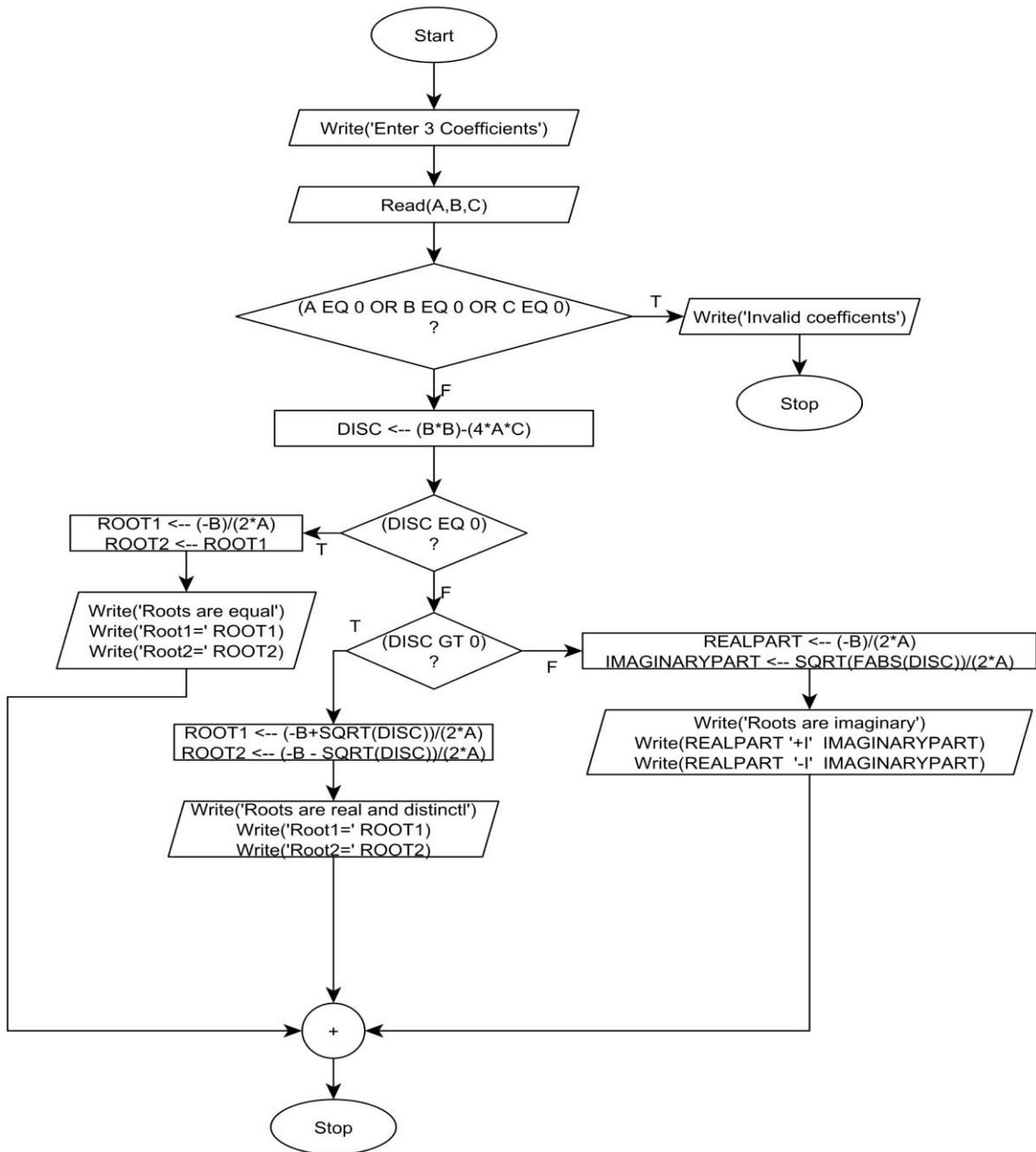
Print “Root1  $\leftarrow$  “,  $x_1 + ix_2$

Print “ Root2  $\leftarrow$  “,  $x_1 - ix_2$

Step 7: [terminate the algorithm]

Stop

### 3. Flowchart to find Roots of the Quadratic Equation



**/\* Program to find roots of the quadratic equation \*/**

```

#include<stdio.h>
#include<stdlib.h>
#include<math.h>

void main()
{
    float a,b,c,d,x1,x2;                                /*Declaration of variables */

    printf("Enter non zero coefficients a, b and c of a quadratic equation \n");
    scanf("%f%f%f",&a,&b,&c);                            /* Input the coefficients */

    if(a==0||b==0||c==0)

    {

    printf("invalid coefficients");
    exit(0);
    }
    d=b*b-4*a*c;                                        /* Computing discriminant */

    if(d==0)                                           /* Computing real and equal roots */
    {
        printf("Roots are
        equal\n"); x1=-b/(2*a);
        x2=-b/(2*a);
        printf("Root1=%f \t Root2=%f",x1,x2);
    }
    else if(d>0)                                       /* Computing real and distinct roots */
    {
        printf("Roots are real and distinct\n");
        x1=(-b+sqrt(d))/(2*a);
        x2=(-b-sqrt(d))/(2*a);
        printf("Root1=%f \t Root2=%f", x1,x2);
    }
    else                                             /* Computing complex roots */
    {
        printf("Roots are complex\n");
        x1=-b/(2*a);
        x2=sqrt(fabs(d))/(2*a);
        printf("Root1=%f+i%f \n",x1,x2);
        printf("Root2=%f-i%f \n",x1,x2);
    }
    }
}

```

---



---

Input-Output

---



---

### Sample Output 1:

```

Enter the three co-efficient:1 4 4
The roots are real and equal
root1=-2.0000 root2=-2.0000

```

**Sample Output 2:**

Enter the three co-efficient:1 5 6

The roots are real and distinct

root1=-2.0000 root2=-3.0000

**Sample Output 3:**

Enter the three co-efficient:2 3 4

The roots are imaginary

root1= -0.75+i 1.19 root2=-0.75 -i 1.19

**Sample Output 3:**

Enter the three co-efficient:2 3 0

Invalid coefficients

---

---

Program 4
-----------

---

---

**Develop a program to find the reverse of a positive integer and check for palindrome or not.**

**Display appropriate messages.**

**ALGORITHM**

Step 1:[start of the algorithm]

Start

Step 2:[read the value of n]

Read n

Step 3:[initialize the variable]

temp  $\leftarrow$  n

rev  $\leftarrow$  0

Step 4:[find the reverse number]

Repeat through step 4 while (n  $\neq$  0)

rem  $\leftarrow$  n % 10

rev  $\leftarrow$  (rev \* 10) + dig

n  $\leftarrow$  n / 10

Step 5: [display reversed number]

print rev

Step 6:[check the reverse number with original number]

if (rev == temp)

print "no is a palindrome"

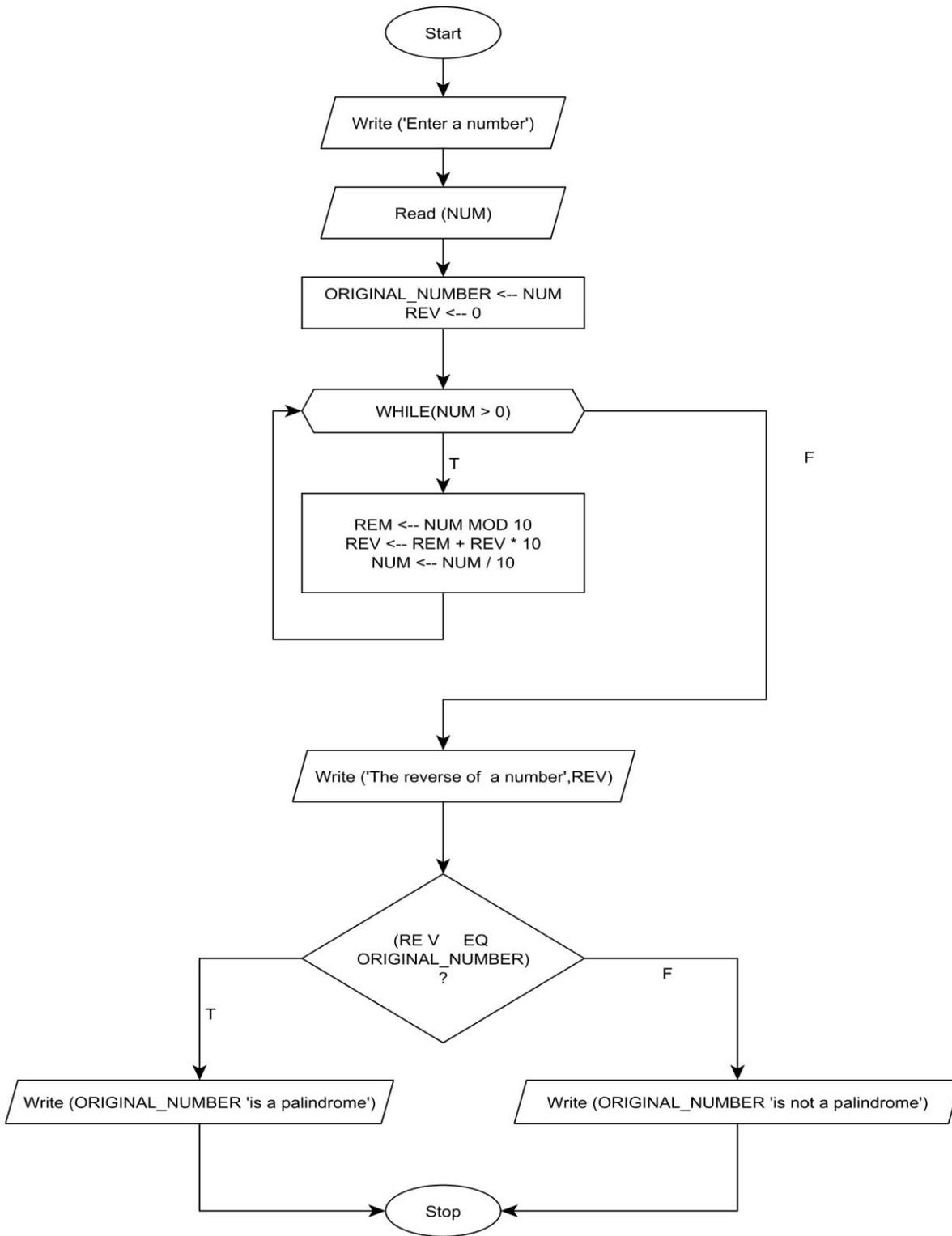
else

print "nois not a palindrome"

Step 7:[terminate the algorithm]

Stop

4. Flowchart to find Reverse of a Given Number and check its a Palindrome or not



```

/* Program to check whether the given number is palindrome or not */
#include<stdio.h>
void main()
{
    int n,temp,rev,rem; /* Declaration of the variables */

    printf(" Enter an integer\n");
    scanf("%d",&n); /* Accept the number */

    temp = n;
    rev = 0;

    /* Calculate the reverse of the number */
    while( n != 0 )
    {
        rem = n % 10;
        rev = rev * 10 + rem;
        n = n / 10;
    }
    /*print reversed number*/
    printf("The reversed number is %d\n",rev);

    /* Check for palindrome */
    if( rev == temp )
        printf(" %d is a palindrome",temp);
    else
        printf(" %d is not a palindrome",temp);
}

```

---



---

Input-Output
--------------

---



---

**Sample Output 1:**

Enter a number:5642  
5642 is not Palindrome

**Sample Output 2:**

Enter a number:1221  
1221 is Palindrome

---

---

Program 5
-----------

---

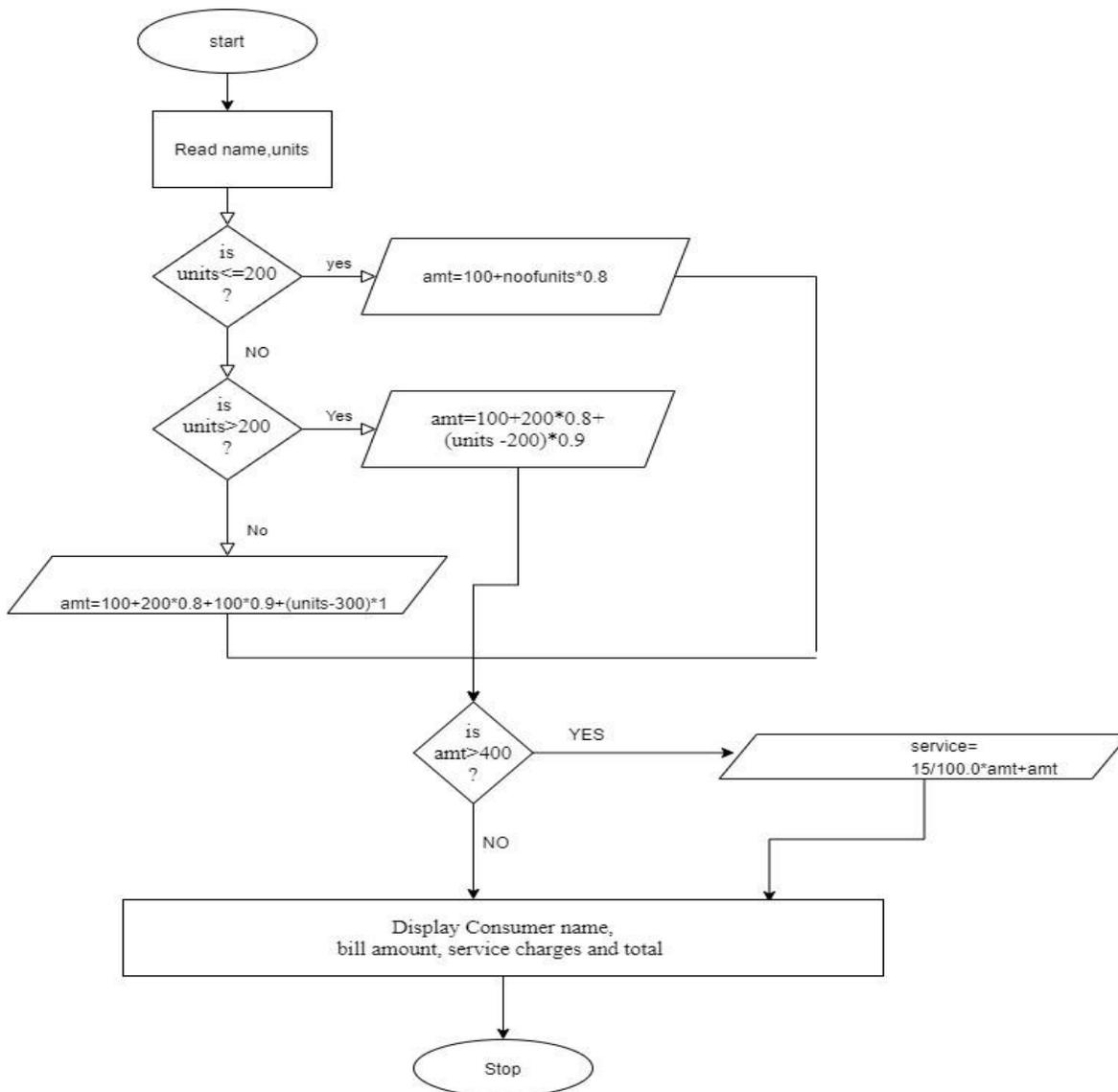
---

**An electricity board charges the following rates for the use of electricity: for the first 200 units 80 paise per unit; for the next 100 units 90 paise per unit; beyond 300 units Rs 1 per unit. All users are charged a minimum of Rs. 100 as meter charge. If the total amount is more than Rs 400, then an additional surcharge of 15% of total amount is charged. Write a program to read the name of the user, number of units consumed and print out the charges.**

**ALGORITHM**

Step1. [Initialize] Start  
Step2. [input the consumer name] Read name  
Step3. [input the number of units consumes] Read units  
Step4. [Calculate the Electricity bill amount]  
    If(units<=200) amt =100+noofunits\*0.8  
    else if((units>200) and (units<=300))  
        amt=100+200\*0.8+ (units -200)\*0.9  
    else  
        amt=100+200\*0.8+100\*0.9+(units-300)\*1  
Step 5.[Calculate the servicecharge]  
    if(amt>400)service= 15/100.0\*amt+amt  
Step6. [Output the result] Print “Consumer name, bill amount,  
    service charges and total ” ,name , amt, sur, total  
Step7. [Finished] Stop.

## Flow chart



```
/*program to compute electricity bill*/  
#include <stdio.h>
```

```

void main()
{
    char name[10];
    int unit,m=100;
    float charge;
    printf("Enter your name and unit Consumed:");
    scanf("%s%d",name,&unit);
    if(unit<=200)
    charge=unit*0.80+m;

    else if(unit>200 && unit<=300)
    {
    charge=(unit-200)*0.90+200*0.80+m;
    }
    else
    {
    charge=(unit-300)*1+200*0.80+100*0.90+m;
    }
    if(charge>=400)
    {
    charge=charge+charge*0.15;
    }
    printf("Name: %s\nCharge: %f",name,charge);
}

```

---

Input-Output

---

### Sample Output 1:

Enter the consumer name: JOHN  
 Enter number of units consume: 200  
 The consumer name = JOHN  
 The number of units consumed =200  
 Electricity Bill Amount = 260.00  
 Service charge=0.00  
 The Total Electricity Bill Amount: 260.00

### Sample Output 2:

Enter the consumer name: RAMESH  
 Enter number of units consume: 460  
 The consumer name = RAMESH  
 The number of units consumed =460  
 Electricity Bill Amount = 510.00  
 Service charge=61.50  
 The Total Electricity Bill Amount: 571.50

**Sample Output 3:**

Enter the consumer name: VINOD

Enter number of units consume: 0

The consumer name = VINOD

The number of units consumed =0

Electricity Bill Amount = 100.00

Service charge=0.00

The Total Electricity Bill Amount: 100.00

---

---

Program 6
-----------

---

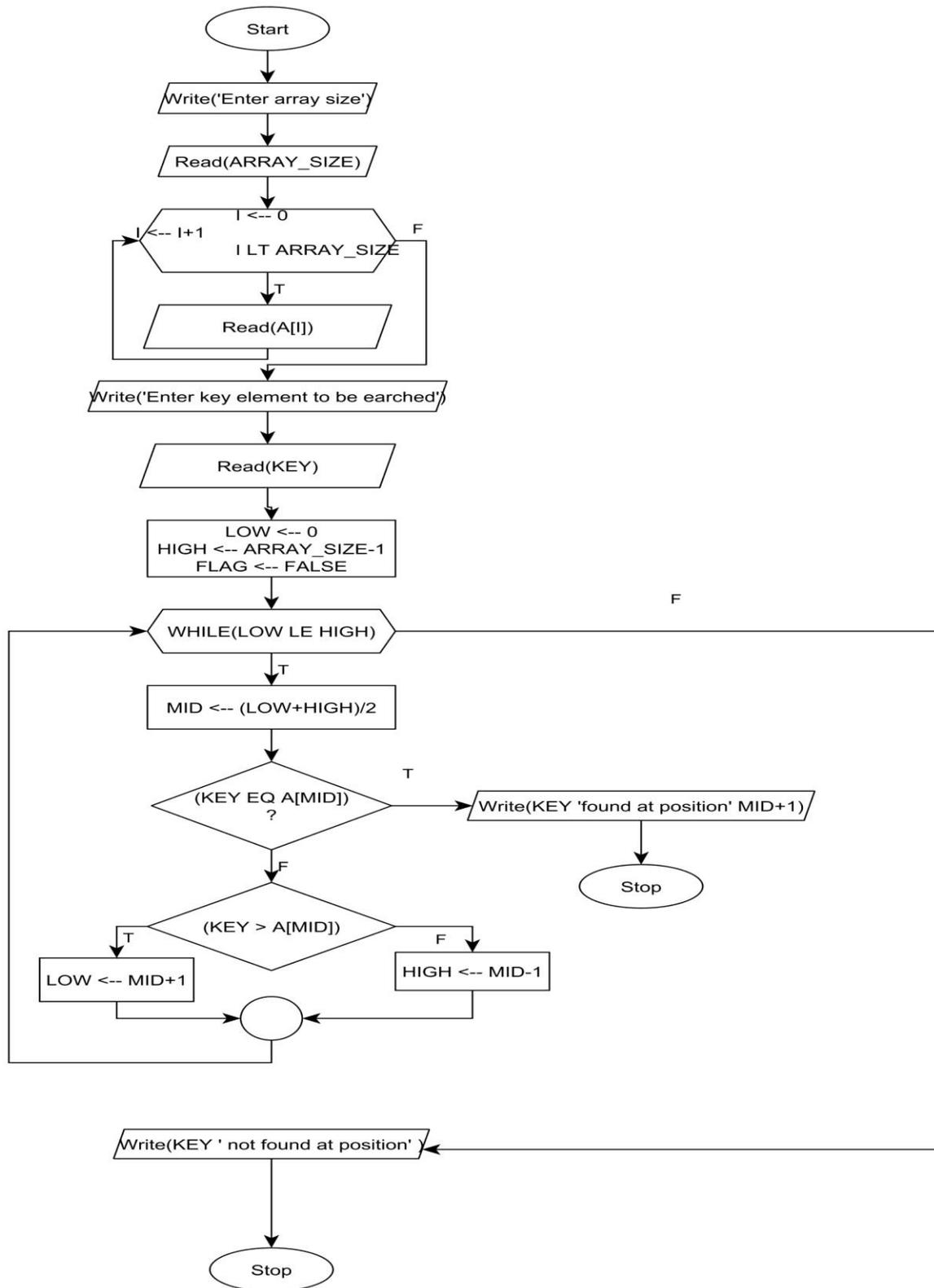
---

**Introduce 1D Array manipulation and implement Binary search.**

**ALGORITHM**

Step1. [Initialize] Start  
Step2. [input value for 'n'] Read n  
Step3. [Enter the n elements of single dimensional array in ascending order ]  
        for i=0 to n-1 read A  
Step4.[input a element to be search] read key  
Step5. [Initialize first, last and middle]assume low= 0, last = n -1 and mid= (low+high)/2  
Step6. [Implement binary search with n elements of an array]  
        while (low <= high)  
            begin while  
                if (mid== key) print "element found"  
            else if (mid>key)  
                high = mid -1  
            else  
                low = mid+ 1  
  
        end while  
Step7. [Output the result] print "element not found"  
Step8. [Finished] Stop.

6. Flowchart to find element using Binary Search Technique



```

/* Program to search an element in a list of elements using binary search */
#include<stdio.h>
#include<stdlib.h>
void main()
{
    int a[30],i,n,key,low,mid,high;
    printf("\n enter the no of elements");
    scanf("%d",&n);
    printf("\n enter the elements : ");
    for(i=0;i<n;i++){
        scanf("%d",&a[i]);
    }
    printf("\n enter the key element to be searched\n");
    scanf("%d",&key);
    low=0;
    high=n-1;
    while(low<=high)
    {
        mid=(low+high)/2;
        if(a[mid]==key)
        {
            printf("element %d is found at %d position :",key,mid+1);
            exit(0);
        }
        else if(a[mid]>key)
            high=mid-1;
        else
            low=mid+1;
    }
    printf("element not found\n");
}

```

## Input-Output

---

### Sample Output 1:

Enter number of elements in  
an array: 5

Enter 5 integers in ascending order

4

7

8

9

45

Enter value to find 9

9 found at location 4.

### Sample Output 2:

Enter number of elements in an array: 7

Enter 7 integers in ascending order

9

23

41

90

132

290

301

Enter value to find 10

Element not found

## Program 7

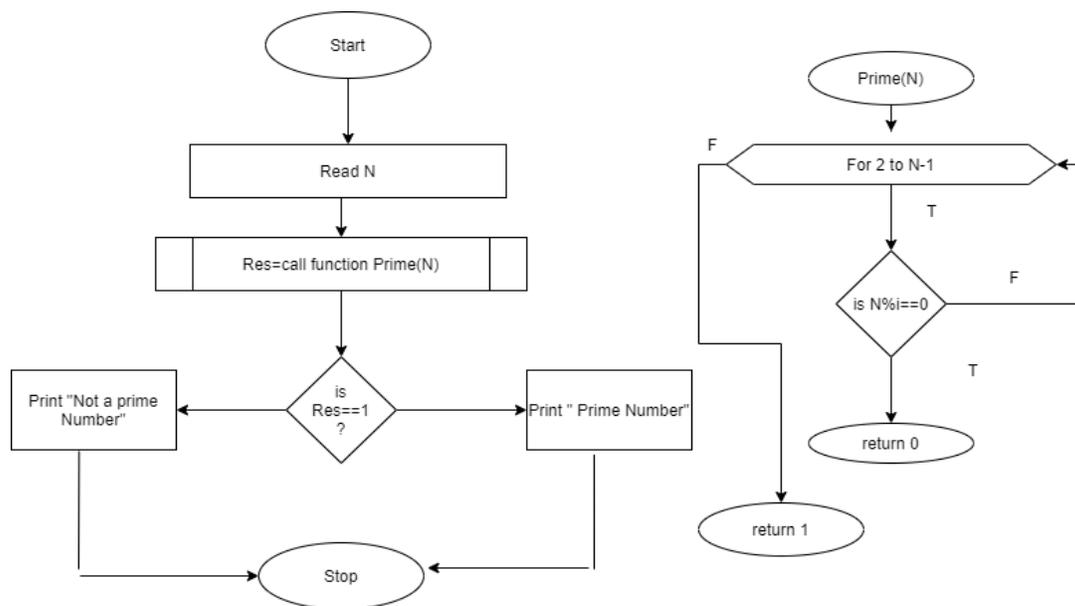
**Implement using functions to check whether the given number is prime and display appropriate messages. (No built-in math function)**

**Purpose:** This program demonstrates **USER DEFINED FUNCTIONS**.

**Procedure:** Input N array elements and to find whether element is present or not

**Input:** Array elements

**Expected Output:** Successful search or unsuccessful search



### ALGORITHM

Step 1. [Initialize] Start

Step 2. [Input an integer number]

Read num

Step 3.res =prime(num) [call function to find prime and not prime numbers]

Step 4. [Output] If (res = 1)

Print("The entered number is a prime")

Else

Print("The entered number is not prime")

Step 5. [Stop]

End

**Algorithm: prime(num)**

Step 1: Start

Step 2: Repeat  $i \leftarrow 2$  to  $n-1$

Step 2.1: if( $n \bmod i == 0$ ) then

return 0

```
End If  
Return 1
```

Step 3:STOP

**/\*program to find prime number\*/**

```
#include <stdio.h>  
int prime(int n);  
void main()  
{  
    int num,res=0;  
    printf("\nENTER A NUMBER: ");  
    scanf("%d",&num);  
    res=prime(num);  
    if(res==1)  
        printf("\n%d IS A PRIME NUMBER",num);  
    else  
        printf("\n%d IS NOT A PRIME NUMBER",num);  
}  
  
int prime(int n)  
{  
    int i;  
    for(i=2;i<=n-1;i++)  
    {  
        if(n%i==0)  
            return 0;  
    }  
    return 1;  
}
```

---

---

Input-Output
--------------

---

---

**Sample Output 1:**

Enter any positive Integer : 7  
The given number 7 is prime

**Sample Output 2:**

Enter any positive Integer : 100 The  
given number 100 is not prime

## PART-B

### Program 8

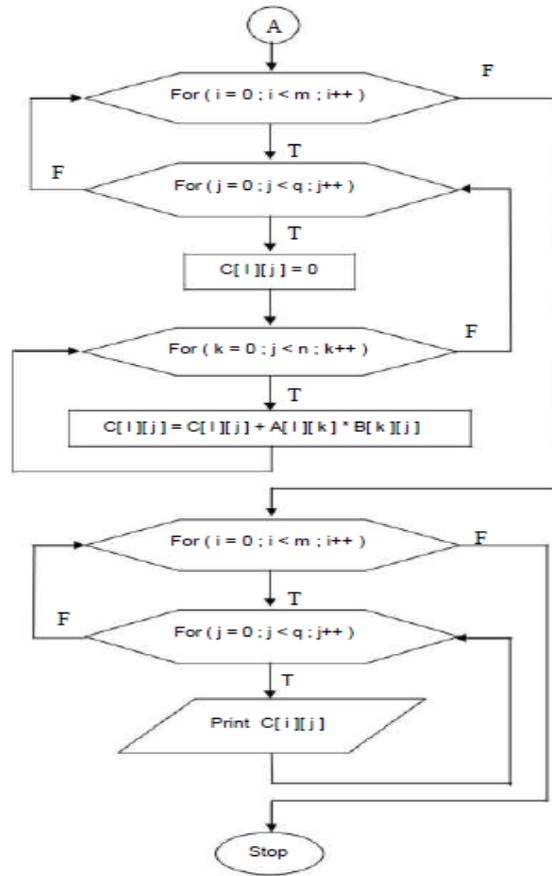
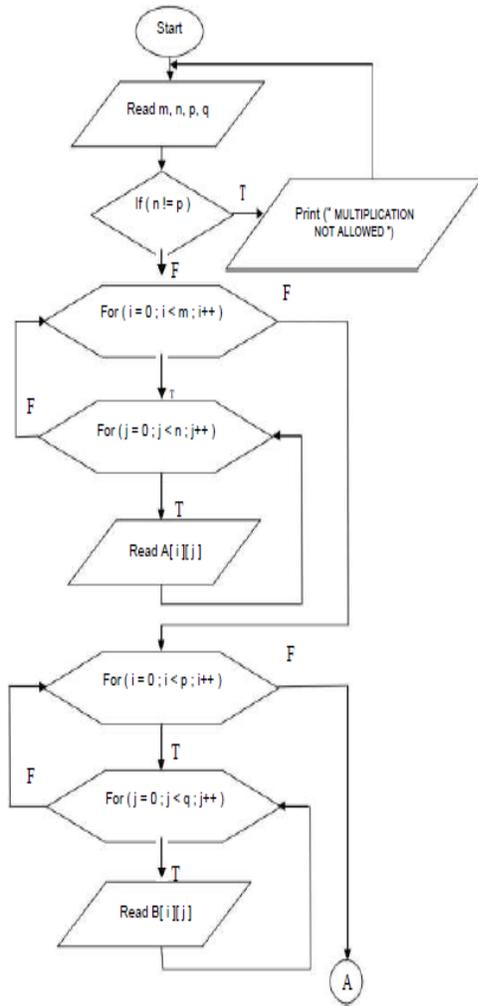
**Develop a program to introduce 2D Array manipulation and implement Matrix multiplication and ensure the rules of multiplication are checked.**

**Purpose:** This program demonstrates Two-Dimensional array.

**Procedure:** Input  $m*n$  and  $p*q$  size of 2 matrices elements to compute matrix multiplication.

### ALGORITHM

- Step 1: [Start of the algorithm]  
Start
- Step 2: [Read the order of the matrix A]  
Read  $m, n$
- Step 3: [Read the order of the matrix B]  
Read  $p, q$
- Step 4: [To check for compatibility condition of matrix multiplication]  
if  $(n \neq p)$   
print "Matrix multiplication not possible"  
go to step 11
- Step 5: [Read the elements of matrix A]  
Repeat through step 5 for  $i \leftarrow 0$  to  $m-1$   
Repeat for  $j \leftarrow 0$  to  $n-1$   
Read  $a[i][j]$
- Step 6: [Read the elements of matrix B]  
Repeat through step 6 for  $j \leftarrow 0$  to  $q-1$   
Repeat for  $i \leftarrow 0$  to  $p-1$   
Read  $b[i][j]$
- Step 7: [Calculate the product of two given matrices]  
Repeat through step 9 for  $i \leftarrow 0$  to  $m-1$   
Repeat for  $j \leftarrow 0$  to  $q-1$   
 $c[i][j] \leftarrow 0$   
Repeat for  $k \leftarrow 0$  to  $n-1$   
 $c[i][j] \leftarrow c[i][j] + a[i][k] * b[k][j]$
- Step 8: [Print the resultant matrix]  
Repeat step 10 for  $i \leftarrow 0$  to  $m-1$   
Repeat for  $j \leftarrow 0$  to  $n-1$   
Print  $c[i][j]$
- Step 9: [terminate the algorithm]  
Stop



```

/* Program to multiply two matrices */
#include<stdio.h>
#include<stdlib.h>
void main()
{
    int i,j,m,n,p,q,k,a[5][5],b[5][5],c[5][5];

    printf("Enter the order of Matrix A\n");
    scanf("%d%d",&m,&n);          /* Read the size of first matrix */
    printf("Enter the order of Matrix B\n");
    scanf("%d%d",&p,&q);          /* Read the size of second matrix */

    if(n!=p)                      /* Check whether multiplication is possible or not */
    {
        printf("Multiplication is not possible\n");
        exit(0);
    }
    printf("Enter the elements of matrix A\n");
    for(i=0;i<m;i++)
    {
        for(j=0;j<n;j++)
        {
            scanf("%d",&a[i][j]); /* Read the elements of matrix A in row major order*/
        }
    }
    printf("Enter the elements of matrix B\n");
    for(i=0;i<p;i++)
    {
        for(j=0;j<q;j++)
        {
            scanf("%d",&b[i][j]); /* Read the elements of matrix B in column major order */
        }
    }

    /* Multiply the matrices */
    for(i=0;i<m;i++)
    {
        for(j=0;j<q;j++)
        {
            c[i][j]=0;
            for(k=0;k<n;k++)
                c[i][j]=c[i][j]+a[i][k]*b[k][j];
        }
    }

    printf("Product matrix is\n");
    for(i=0;i<m;i++)

```

```

        {
            for(j=0;j<q;j++)
            {
                printf("%d\t",c[i][j]);        /* Display product matrix */
            }
            printf("\n");
        }
    } /*end of main*/

```

---

## Input-Output

---

### Sample Output 1:

Enter the size of the first matrix A: 2 2  
 Enter the size of the second matrix B: 2 2  
 Enter the elements of first matrix  
 1 2  
 3 4

Enter the elements of second matrix  
 2 3  
 4 5

Matrix A is  
 1 2  
 3 4

Matrix B is  
 2 3  
 4 5

The product of 2 matrices is  
 10 13  
 22 29

### Sample Output 2:

Enter the order of the matrix A: 2 3  
 Enter the order of the matrix B: 2 3  
 Multiplication is not possible

---

---

Program 9
-----------

---

---

**Develop a Program to compute Sin(x) using Taylor series approximation .Compare your result with the built- in Library function. Print both the results with appropriate messages.**

**ALGORITHM**

Step 1:[Start the algorithm]  
Start

Step 2:[Read the value of x]  
Read x

Step 3:[Initialize the variables]  
degree  $\leftarrow$  x;  
x  $\leftarrow$  x\*(3.142/180.0)

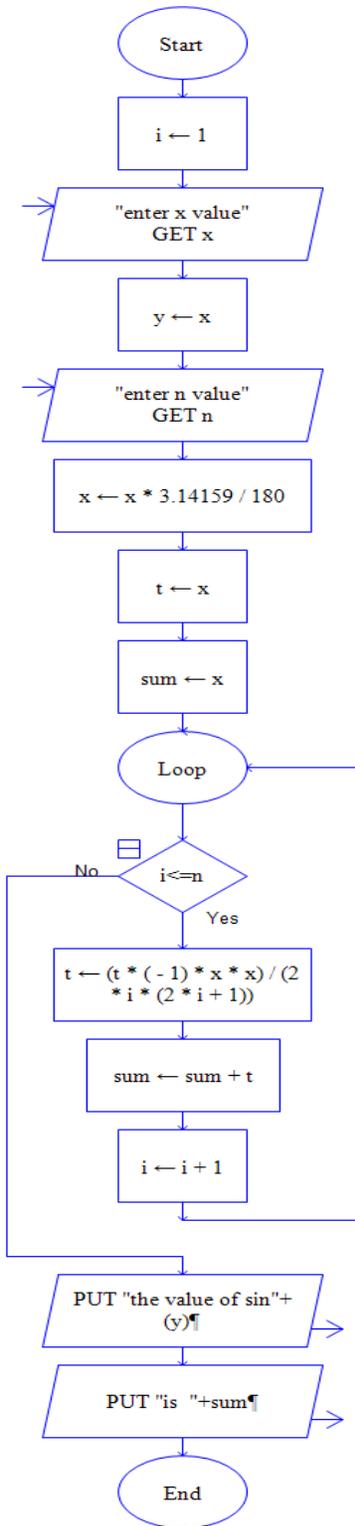
Step 4:[Read the value of no of terms]  
Read n

Step 5:[Initialize the variables]  
term  $\leftarrow$  x;  
sum  $\leftarrow$  t;

Step 6:[Repeat the following steps for i=1 to n terms]  
term  $\leftarrow$  term\*(-1)\*x\*x/2\*i\*(2\*i+1);  
sum=sum+term;

Step 7:Print the result  
sum of sine series=sum  
sum using library function=sin(x)

Step 8: Stop



**/\*Program to calculate sin(x) using Taylor series\*/**

```
#include<stdio.h>
#include<math.h>
void main()
{
    int i, n,degree;
    float x, sum,term;
    printf(" Enter the value for x : ");
    scanf("%f",&x);
    printf(" Enter the value for n : ");
    scanf("%d",&n);
    degree=x;
    x=x*3.14159/180;
    term=x;
    sum=term;

    /* Loop to calculate the value of Sine */
    for(i=1;i<=n;i++)
    {
        term=(term*(-1)*x*x)/(2*i*(2*i+1));
        sum=sum+term;
    }

    printf("sin(%f) using sine series=%f\n",degree,sum);
    printf("sin(%f) using library function=%f\n",degree,sin(x));
}
```

---

---

## Input-Output

---

---

### **Sample Output 1:**

Enter the value of degree 90

sin(90)= 1.000000 without using built in function sin(90)= 1.000000 using built in function

### **Sample Output 2:**

Enter the value of degree 45

sin(45)= 0.707179 without using built in function sin(45)= 0.707179 using built in function

---

---

Program 10
------------

---

---

**Write functions to implement string operations such as compare, concatenate, string length. Convince the parameter passing techniques.**

**ALGORITHM**

Step 1:       **Start**

Step 2:        [Prompt and Input]  
              Write("Enter string1")  
              Read(S1)  
              Write("Enter String2")  
              Read(S2)

Step 3:        [Update strings with NULL character, Call STRLEN and Output]  
              LENGTH ← STRLEN(S1)  
              S1[LENGTH-1] ← "\0"  
              Write(S1, "Length is", LENGTH)  
              LENGTH ← STRLEN(S2)  
              S2[LENGTH-1] ← "\0"  
              Write(S1, "Length is", LENGTH)

Step 4:        [Call STRCMP]  
              CMP ← STRCMP(S1, S2)  
              If (CMP EQ 0) Then  
              Write(S1, "Equals", S2)  
              Else If (CMP GT 0) Then  
              Write(S1, "Greater Than", S2) Else  
              Write(S1, "Lesser Than", S2)  
              End If

Step 5:        [Call STRCAT]  
              Write(„Before Concatenation“, S2)  
              STRCAT(S2, S1)  
              Write(„After Concatenation“, S2)

Step 6:        STOP

Sub-Algorithm: STRLEN(S)

This sub-Algorithm returns the length of the string S, S is a parameter of type array of characters, the variable COUNT is of type integer

Step 1 :        Start

Step 2 :        [Initialize] COUNT ← 0

Step 3 :        While (S[COUNT] NE NULL)  
              COUNT ← COUNT + 1  
              End While

Step 4 :        [Finished] Return COUNT

**Sub-Algorithm: STRCMP(S1, S2)**

This sub-Algorithm compares 2 strings S1 and S2 character by character and return the difference of the ASCII values of the compared characters of both strings. The variable COUNT is of type integer

Step 1 :                   Start

Step 2 :                   [Initialize] COUNT←0

Step 3 :                   [Process and Return]

          While(S1[COUNT] NE NULL)

          If(S1[COUNT] NE S2[COUNT]) Then

          Return(S1[COUNT] - S2[COUNT])

          Else

          COUNT←COUNT+1

          End If

          End While

          Return 0

**Sub-Algorithm: STRCAT(DEST, SRC)**

This sub-Algorithm adds SRC string to DEST string. DEST and SRC are parameters of type array of characters. Variables COUNT and I are of type integers.

Step 1 :                   Start

Step 2 :                   [Initialize] COUNT←0

          I←0

Step 3 :                   [Process and Return]

          While(DEST[COUNT] NE NULL)

          COUNT←COUNT+1

          End While

          While(SRC[I] NE NULL)

          DEST[COUNT]←SRC[I]

          COUNT←COUNT+1

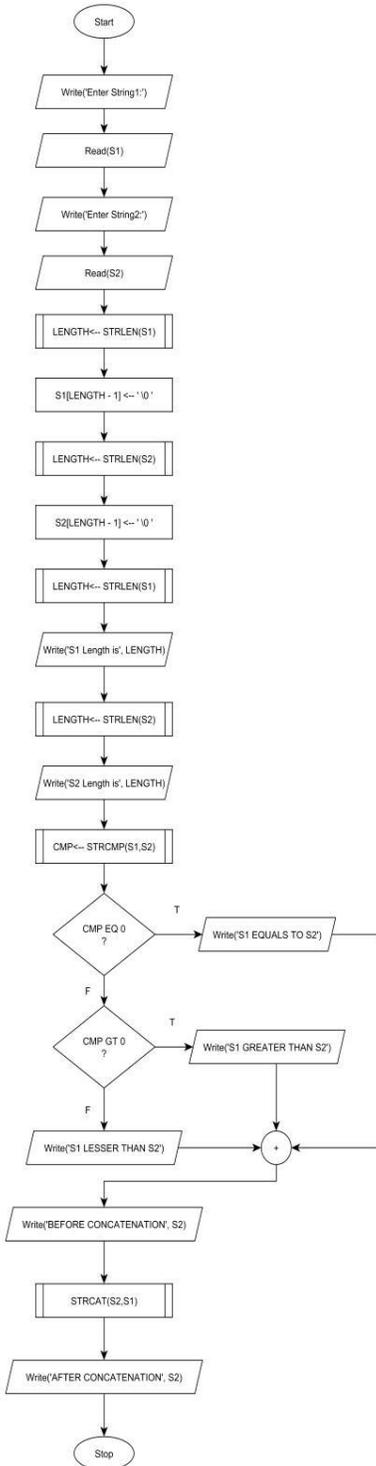
          I←I+1

          End While

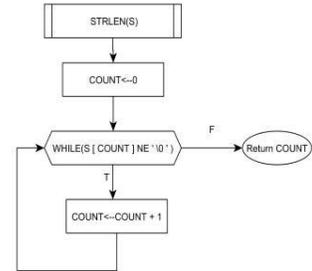
          DEST[COUNT]←“\0”

Step 4 :                   [Finished] Return

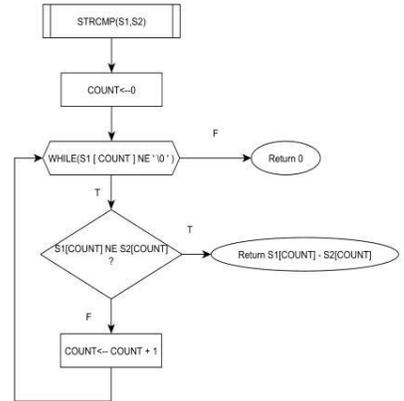
10. Flowchart for string operations like string length, string comparison and string concatenation using user defined functions



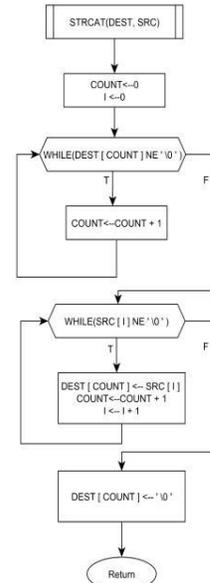
Function to find string length



Function to compare two strings



Function to concatenate two strings



```

#include<stdio.h>
int STRLEN(char S1[]);

int STRCMP(char S1[],char S2[]);

void STRCAT(char dest[],char src[]);

void main()
{
char s1[20],s2[20];
printf("enter the string1 \n");
scanf("%s",s1);
printf("enter the string2 \n");
scanf("%s",s2);
int length;
length=STRLEN(s1);
printf("%s length is %d \n",s1,length);
length=STRLEN(s2);
printf("%s length is %d \n",s2,length); int cmp;
cmp=STRCMP(s1,s2);
if(cmp==0)
printf("%s is equal to %s\n",s1,s2); else if(cmp>0)
printf("%s is greater than %s\n",s1,s2); else
printf("%s is lesser than %s\n",s1,s2); printf("before concatenation\n"); printf("%s\n",s2);
STRCAT(s2,s1);
printf("after concatenation\n");
printf("%s\n",s2);
return 0;
}

```

```
int STRLEN(char s[])
{
int count=0;
while(s[count]!='\0')
{
count++;
}
return count;
}
int STRCMP(char s1[],char s2[])
{
int count=0;
while(s1[count]!='\0')
{
if(s1[count]!=s2[count]) return (s1[count]-s2[count]);
count++;
}
return 0;
}
void STRCAT(char dest[],char src[])
{
int count=0,i=0;
while(dest[count]!='\0')
count++;
while(src[i]!='\0')
{
dest[count]=src[i];
count++;
}
```

```
i++;  
}  
dest[count]='\0';  
return;  
}
```

---

---

Program 11
------------

---

---

**Develop a program to sort the given set of N numbers using Bubble sort.**

**Purpose:** This program demonstrates **NESTED FOR** loop.

**Procedure:** To read an array of elements a[]. While iterating, compare each pair of adjacent items in every pass. If the former value is greater than the latter one, their positions are swapped. Over a number of passes, at most equal to the number of elements in the list, all of the values drift into their correct positions. Then print sorted array element.

**Input:** Number of elements-n  
An array of unsorted elements-a[]

**Expected Output:** An array of sorted elements-a[]

**ALGORITHM**

**Algorithm: Bubble sort**

Step 1: [Start of the algorithm]  
Start

Step 2: [Read the size of the array]  
read n

Step 3: [Read the array elements]  
Repeat for i=0 to n-1  
read a[i]

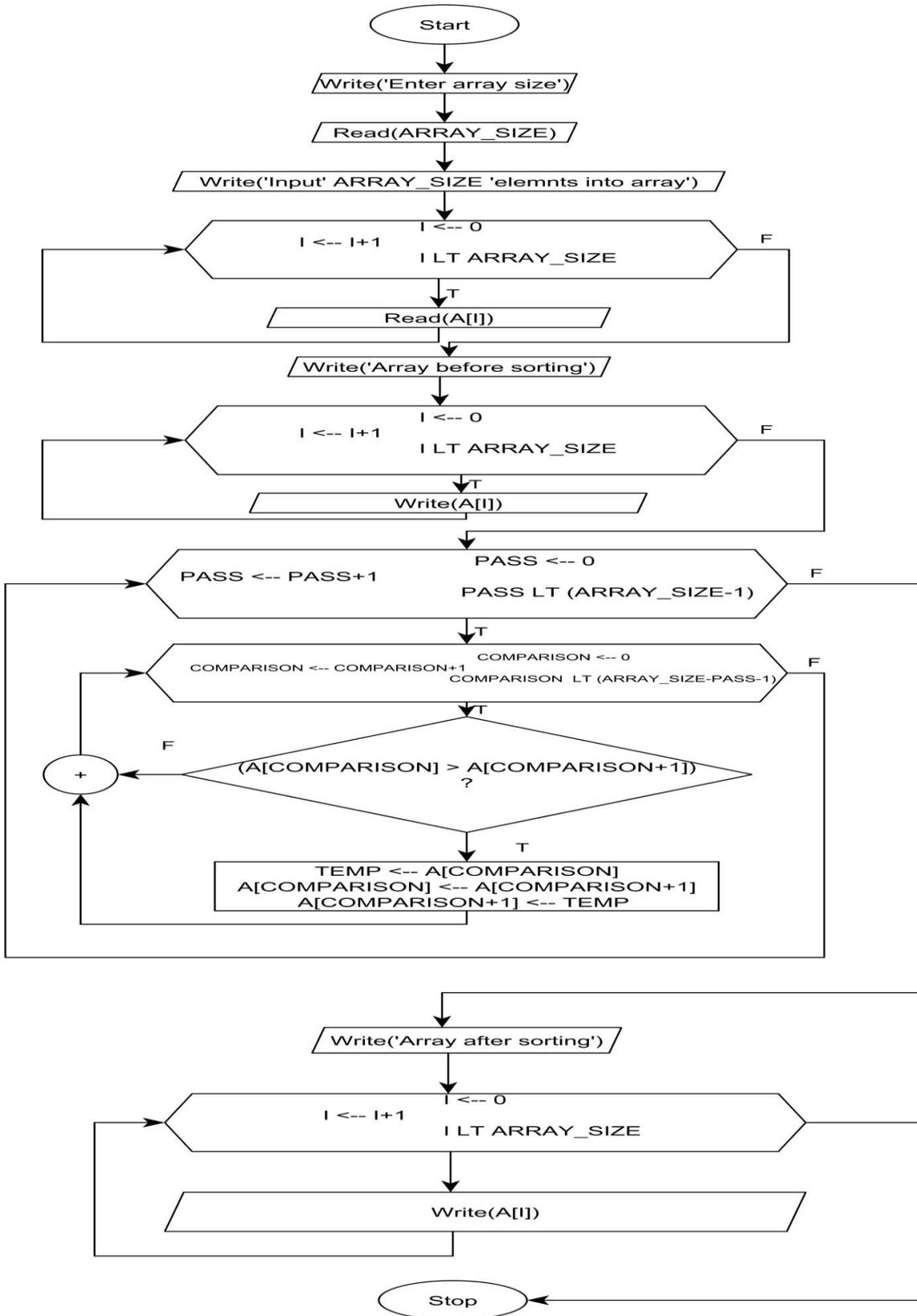
Step 4: [Print the given array]  
Repeat for i=0 to n-1  
print a[i]

Step 5: Repeat through step 5 for pass  $\leftarrow$  1 to n-1  
Repeat for comp  $\leftarrow$  0 to n-pass  
if(a[comp]>a[comp+1])  
temp  $\leftarrow$  a[comp]  
a[comp]  $\leftarrow$  a[comp+1]  
a[comp+1]  $\leftarrow$  temp

Step 6: [Print the sorted array]  
Repeat for i  $\leftarrow$  0 to n-1  
print a[i]

Step 7 :[terminate the algorithm]  
Stop

11. Flowchart to sort array elements in ascending using Bubble Sort Technique



```

/*Program to sort the given elements, using bubble sort*/
#include<stdio.h>
void main()
{
    int a[50],i,temp,n; /* Declaration of the variables */
    printf("Enter the value of n\n");
    scanf("%d",&n); /* Read the size of an array */

    printf("Enter the array elements\n");
    for(i=0;i<n;i++)
        scanf("%d",&a[i]); /* Read the elements */

    printf("The given array elements are \n");
    for(i=0;i<n;i++)
        printf("%d\t",a[i]);
    /* Perform Bubble Sort */
    for( int pass=1;pass<n;pass++)
    {
        for(int comp=0;comp<n-pass;comp++)
        {
            if(a[comp]>a[comp+1])
            {
                temp=a[comp];
                a[comp]=a[comp+1];
                a[comp+1]=temp;
            }
        }
    }

    printf("The array after sorting\n"); /* Display the array after sorting */
    for(i=0;i<n;i++)
        printf("%d\t",a[i]);
}

```

---

---

## Input-Output

---

---

### **Sample Output:**

Enter the size of an array: 5

Enter the 5 elements of an array:

5 4 3 2 1

The given array before sorting is: 5

4

3

2

1

the sorted array is:

1

2

3

4

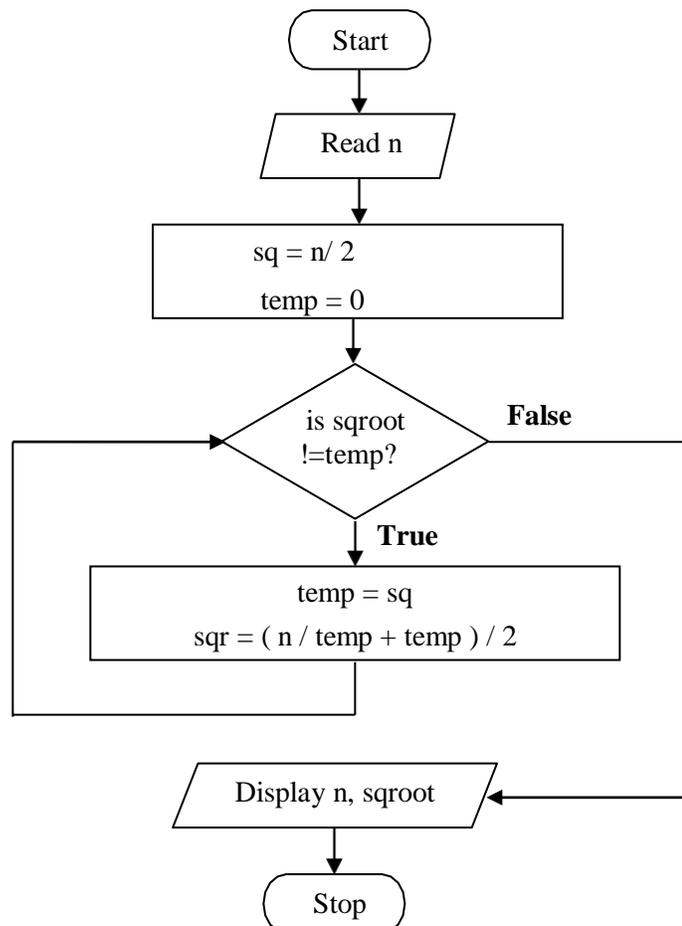
5

## Program 12

Develop a program to find the square root of a given number N and execute for all possible inputs with appropriate messages. Note: Don't use library function sqrt(n).

### ALGORITHM

- Step1. [Initialize] Start  
Step2. [input the number]  
    read n  
Step 3:[Initialize the variables]  
    term $\leftarrow$ 0;  
    sq $\leftarrow$ n/2;  
Step 4:[Repeat the following steps while (sq NE temp)]  
    temp $\leftarrow$ sq  
    sq $\leftarrow$ (n/temp+temp)/2  
    End While  
Step 7:Print the result  
    Square root of number is=sq  
    Square root using library function=sqrt(n)  
Step 8: [Finished] Stop



```
/*program to find the square root of a number*/
#include<stdio.h>
void main( )
{
    float sq,temp,n;
    printf("Enter any number\n");
    scanf("%f",&n);
    temp=0;
    sq=n/2;
    while(sq!=temp)
    {
        temp=sq;
        sq=(n/temp+temp)/2;
    }
    printf("The square root of '%f' is '%f'", n, sq);
    printf("the square root of '%f' using built in function is '%lf' ",n,sqrt(n));
}
```

---

Input-Output
--------------

**Sample Output 1:**

Enter a number to which square root to be found: 4  
Square root of a real number 4.00 is 2.000

**Sample Output 2:**

Enter a number to which square root to be found: 8.9  
Square root of a real number 8.9 is 2.893

**Sample Output 3:**

Enter a number to which square root to be found: -7  
Sorry Invalid Number

---

---

Program 13
------------

---

---

**Implement structures to read, write and compute average-marks and the students scoring above and below the average marks for a class of N students.**

**Algorithm**

Step1: Input number of students, Read n

Step 2: Initialize total = 0

Step 3: Input details of students i.e., name, usn and marks for all nstudents

For i=0 to n-1 do

    Read s[i].name, s[i].usn, s[i].marks

Step 4: Display details of students i.e., name, usn and marks for all nstudents

For i=0 to n-1 do

    Print s[i].name, s[i].usn, s[i].marks

End i for loop

Step5: For i=0 to n-1 do

    total = total + s[i].marks

    End i for loop

End i for loop

Step 6: Calculate average,  $avg = total / n$

Step 7: Display whether a student is below average or above average

For i=0 to n-1 do

    If ( s[i].marks > avg )

        Display "Student is above average"

    Otherwise

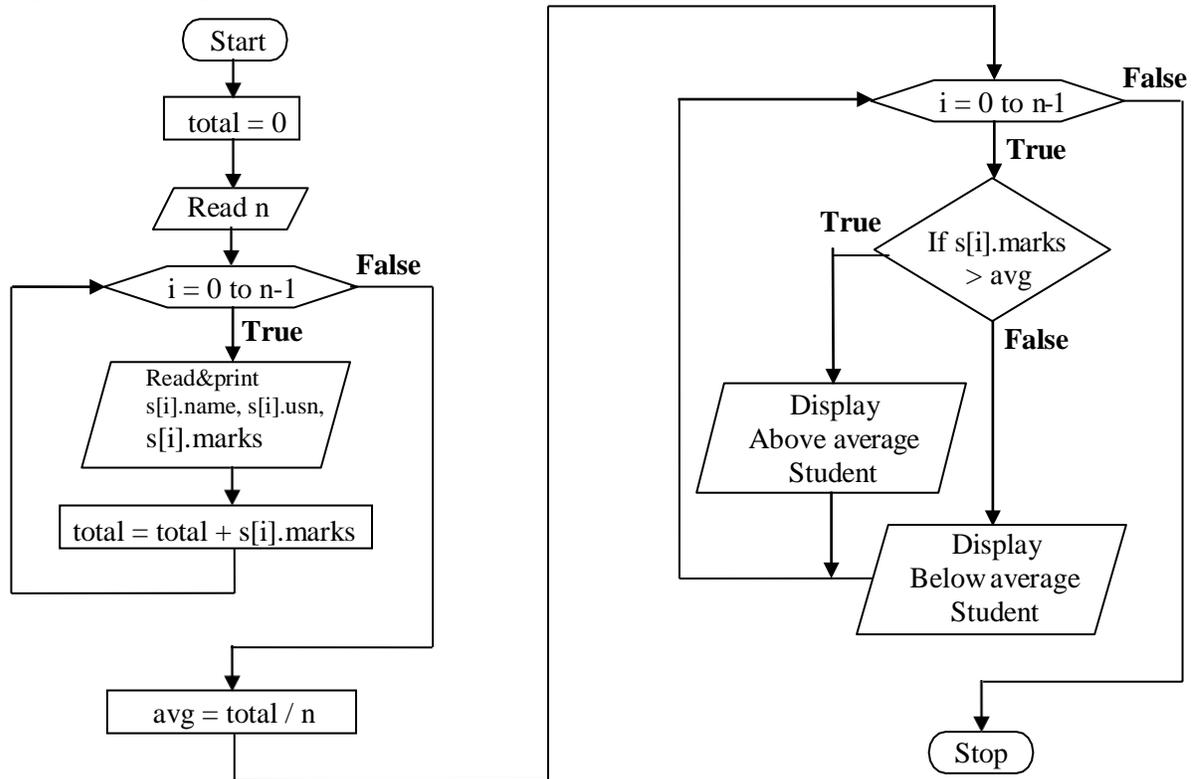
        Display "Student is below average"

    End i for loop

Step 8: Stop

## Flowchart

*/\*program to compute above and below the average marks\*/*



## Program

```
#include<stdio.h>
struct student
{
char name[20];
char usn[10];
int marks;
}s[10];

void main()
{
int i, n, total=0;
float avg = 0.0;
printf("\nEnter the number of students : ");
scanf("%d", &n);
for(i=0; i<n;i++)
{
printf("enter student %d details",i+1);
printf("Enter USN: ");
scanf("%s",s[i].usn);
printf("Enter name: ");
```

```

scanf("%s",s[i].name);
printf("Enter marks: ");
scanf("%d",&s[i].marks);
printf("\n");
}
printf("Displaying Information:\n\n");
// displaying information
for(i=0; i<n; i++)
{
printf("\nUSN: %s\n",s[i].usn);
printf("Name: %s\n ", s[i].name);
printf("Marks: %d",s[i].marks);
printf("\n");
}
for(i=0;i<n;i++)
{
total=total+s[i].marks;
}
avg = total / n;
printf("\nThe average marks for the class is : %f\n", avg);
for(i=0; i<n; i++)
{
if(s[i].marks > avg)
printf("\nThe student %s has scored above average\n",s[i].name);
else
printf("\nThe student %s has scored below average\n", s[i].name);
}
}
}

```

---



---

Input-Output

---



---

**Sample Output:**

**Output**

Enter the number of students : 3

Enter the details of Student 1

Name: Raju

Usn:4AD17CS036

Marks:67

Enter the details of Student 2

Name: Sahana

Usn:4AD17CS405

Marks:77

Enter the details of Student 3

Name:Manasa

Usn:4AD17CS025

Marks:83

The average marks for the class is : 75.67 The student

Raju has scored below average The student Sahana has

scored above average

The student Manasa has scored above average

---

---

Program 14
------------

---

---

**Develop a program using pointers to compute the sum, mean and standard deviation of all elements stored in an array of n real numbers.**

**ALGORITHM**

Step 1:Start

Step 2:Input the number of elements, Read n

Step 3:Initialize pointer so that it points to the beginning of the array

Step 4:Initialize variables, variance=0, sum=0

Step 5:Input array elements

    For i = 0 to n-1 do

        Read array element using pointer, Read a value to be stored at location (ptr + i) Calculate sum,  
        sum = sum + \*(ptr + i)

    End i for loop

Step 6:Calculate mean, mean = sum / n

Step 7:Calculate variance

    For i = 0 to n-1 do

        variance = variance + (ptr[i] - mean) \* (ptr[i] - mean)

    End i for loop

    variance = variance / n

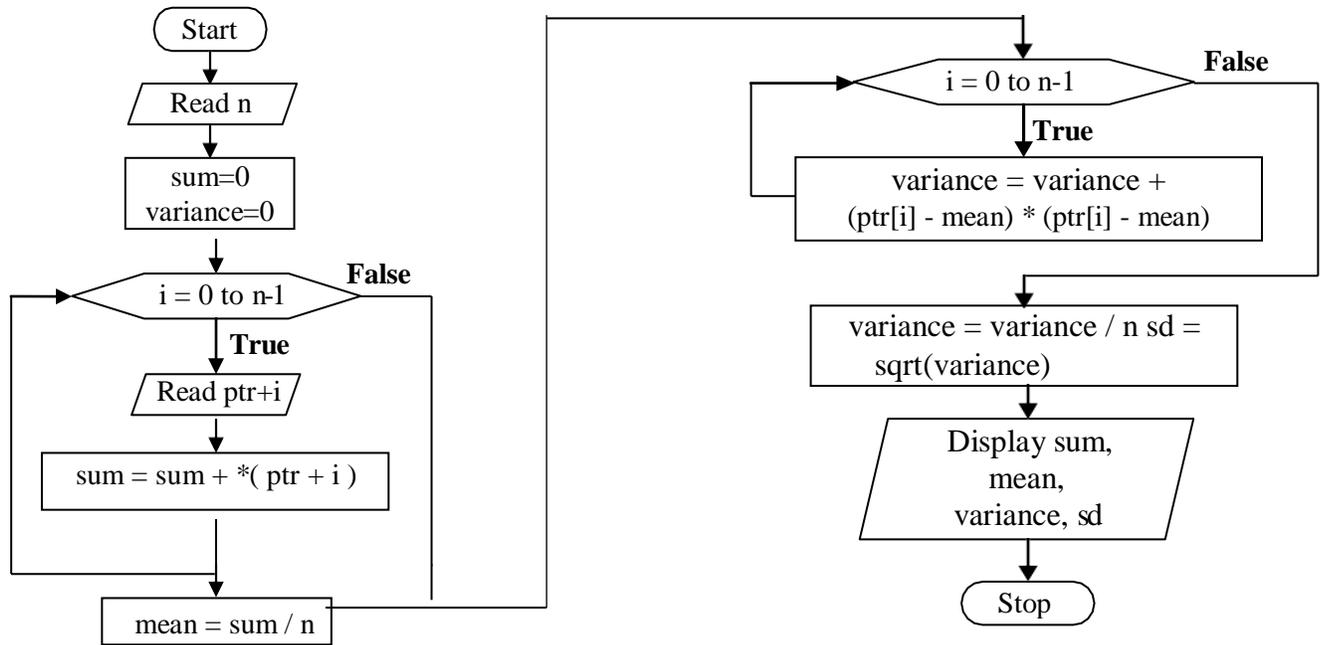
Step 8:Calculate standard deviation,

    sd = sqrt(variance)

Step 9: Display the values of sum, mean, variance and standard deviation.

Step 10:Stop

## Flowchart



**/\* Program to compute the sum, mean, and standard deviation of all elements \*/**

```

#include<stdio.h>
#include<math.h>
void main()
{
    int i, n;
    float mean = 0.0, variance = 0.0, sd = 0.0, num[100], sum=0.0; float *ptr;
    ptr = num;
    printf("\nEnter the number of Values : ");
    scanf("%d",&n);
    printf("\nEnter %d values\n", n);
    for(i=0; i<n; i++)
    {
        scanf("%f", ptr+i);
        sum += *(ptr+i);
    }
    mean = sum / n;
    for(i=0; i<n; i++)
    {
        variance += (ptr[i] - mean) * (ptr[i] - mean);
    }
}
  
```

```

        variance /= n;
        sd = sqrt(variance);
        printf("\nThe values entered are");
        for(i=0; i<n; i++)
        {
            printf("\n\t%f", ptr[i]);
        }
    printf("\n\tsum=%f\n\tMean = \t%f\n\tVariance = \t%f\n\tStandard Deviation = \t%f\n", sum,mean,
    variance, sd);

}

```

---

Input-Output

---

**Sample Output:**

Enter the number of Values : 4

Enter 4 values

1.1    2.2    3.3    4.4

The values entered are 1.1

2.2

3.3

4.4

Sum=11.0

Mean = 2.75

Variance = 1.5125

Standard Deviation = 1.22984

## Program 15

### Implement Recursive functions for Binary to Decimal Conversion.

**Purpose:** This program demonstrates **RECURSION**.

**Procedure:** Input binary number and call the recursive function convert for translating binary number to decimal number.

**Input:** A binary number bin.

**Expected Output:** Decimal number dec.

### ALGORITHM

Step1:Start

Step2:Input binary number, Read binval

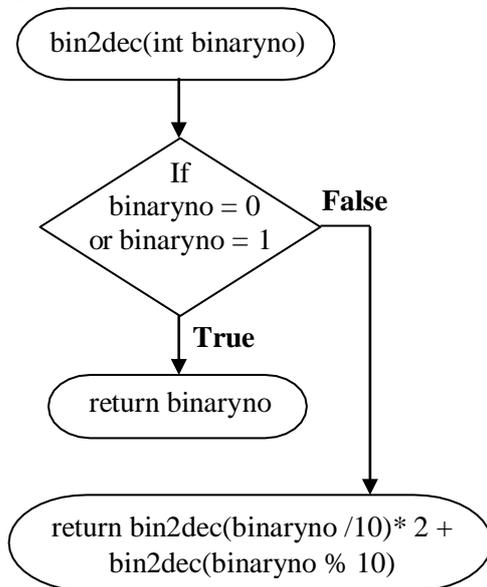
Step3:Call recursive function that converts binary no to decimal value

$decval = bin2dec(binval)$

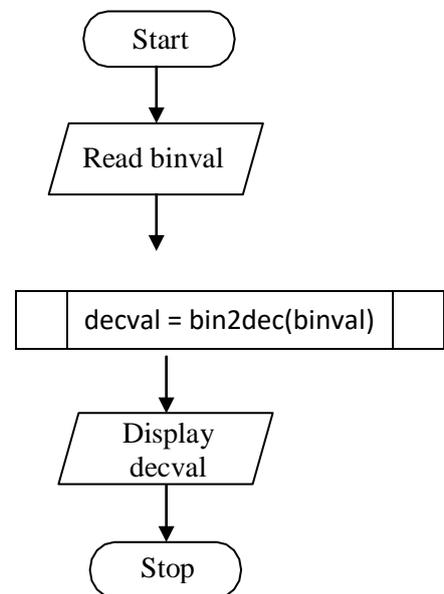
Step 4:Display decimal number, Display decval

Step5: Stop

### Flowchart for recursive function program



### Flowchart for main or driver program



```

/*program to convert binary to decimal*/
#include<stdio.h>
int bin2dec(int n);
void
main()
{
    int binval, decval;
    printf("\nEnter the binary
value : "); scanf("%d",
&binval);
    decval = bin2dec(binval);
    printf("\nDecimal equivalent of %d is %d\n", binval, decval);
}

int bin2dec(int binaryno)
{
    if(binaryno == 0 ||
        binaryno == 1)
        return binaryno;
    else
        return bin2dec(binaryno /10)* 2 + bin2dec(binaryno %10);
}

```

---



---

Input-Output
--------------

---



---

**Sample Output 1:**

Enter the binary value :  
11111 Decimal  
equivalent of 11111 is  
31

**Sample Output 2:**

Enter the binary value :  
1111111 Decimal  
equivalent of 1111111 is

127

**Sample Output 3:**

Enter the binary value :

11011 Decimal

equivalent of 11011 is

27