Partnering in Academic Excellence

Channabasaveshwara Institute of Technology
(Affiliated to VTU, Belgaum & Approved by AICTE, New Delhi)
(**NAAC Accredited & ISO 9001:2015 Certified Institution)**
NH 206 (B.H. Road), Gubbi, Tumkur – 572 216. Karnataka.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

LAB MANUAL

(2018 - `19)

COMPUTER PROGRAMMING LABORATORY

18CPL17/27

I SEMESTER

Name : ____

USN : ____

Batch : ______________________ Section: __________________

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CONTENTS

Department of Computer Science & Engineering

COMPUTER PROGRAMMING LABORATORY
[As Per Choice Based Credit System (CBCS) System]
(Effective from the academic year 2018 -2019)
SEMESTER – I/II

Subject Code	18CPL17/27	**CIE Marks**	40
Number of Lecture Hours/Week	2	**SEE Marks**	60
Total Number of Lab Hours	32	**Exam Hours**	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 foe 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 and structures in problem solving.

Conduct of Practical Examination:

- All laboratory experiments, excluding the first, are to be included for practical examination.
- Experiment distribution:
  - For questions having only one part: Students are allowed to pick one experiment from the lot and are given equal opportunity.
  - 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.
- Strictly follow the instructions as printed on the cover page of the answer script for breakup of marks.
- Change of experiment is allowed only once and marks allotted for procedure part to be made zero.
- Marks Distribution (Subjected to change in accordance with university regulations)

a). For questions having only one part – Procedure + Execution + Viva-Voce: 15+ 70 + 15 = 100 Marks

b). For questions having part A and B

  i. Part A – Procedure + Execution + Viva = 4 + 21 + 5 = 30 Marks

  ii. Part B – Procedure + Execution + Viva = 10 + 49 + 11 = 70 Marks

INTRODUCTION

Description about Functional block diagram of Computer:

A computer is an electronic device, which mainly performs the four functions as **reading**, **processing**, **displaying** and **storing** on data. These functions of a computer system can be carried out by using the three main units namely input unit, system unit and output unit. The block diagram of a computer system is as follows:

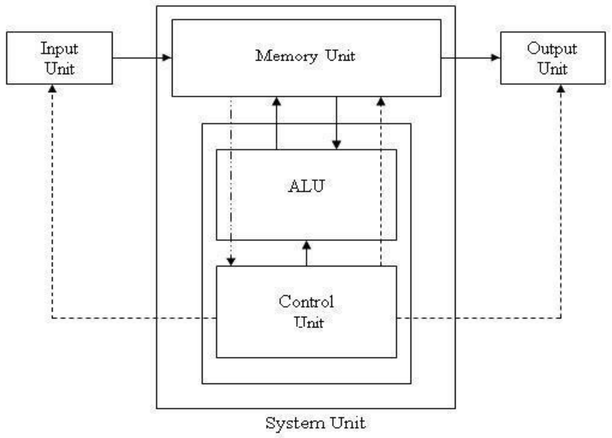


Fig 1: Block Diagram of a Computer

Notations:

- Data and Results flow ――――――――
- Control instructions to other units from control unit - - - - - - - - - - - -
- Instructions from memory unit to control unit ‒ · ‒ · ‒ · ‒ · ‒

System or Central Processing Unit (CPU): is commonly known as "processor" that executes the instructions of a computer program. It has Control Unit (CU) and Arithmetic & Logical Unit (ALU). These two units perform the basic arithmetic, logical, and input/output operations.

a) **Input unit:** is used to enter data and information into a computer. The devices like keyboard, mouse and scanner are commonly used input devices.
- A keyboard is used to enter alphanumeric characters and symbols.
- The mouse is used to pick or select a command from the monitor screen.
- A scanner is used to scan an image or read a barcode and so on.

b) **Arithmetic and Logic Unit (ALU):** is a digital circuit that perform arithmetic (Add, Sub, Multiplication, Division) and logical (AND, OR, NOT) operations. It helps in fast computation of scientific calculations on floating-point number.

c) **Control unit (CU):** is the circuitry that controls the flow of information through the processor and coordinates the activities of the other units within the processor.

Functions of Control unit

Accessing data & instructions from memory unit.
Interpreting instructions
Controlling input and output units

Overall supervision of a Computer system

d) **Memory Unit (MU):** is the unit where all the input data and results are stored either temporarily or permanently. The CPU memory is also called as memory register. The memory of a computer has two types:

   a. ***Main Memory / Primary Memory units***
      i. Random Access Memory (RAM)
      ii. Read Only Memory (ROM)

   b. ***Secondary Memory / Auxiliary Memory***

e) **Output Unit:** It is used to display or print results from a computer. Monitor, printer and plotters are commonly used output devices.

f) **Bus**: A bus is a collection of wires that carries data/Instructions. It connects physical components such as cables, printed circuits, CPU, Memory, Peripherals etc., for sharing of Information and communication with one another. The purpose of buses is to reduce the number of "pathways" needed for communication between the components, by carrying out all communications over a single data channel.

Types of Buses:

1. **System Buses:** The system buses are used to transfer the data and instructions between Main memory (Random Access Memory) and CPU. These are classified into following three types.

Data Bus	Address Bus	Control Bus
It is used to transfer the data between Processor, Memory and I/O devices	It is used to transfer the addresses of data and Instructions stored in memory.	It is used to transfer the control signals between CPU, Memory and I/O devices.
Bidirectional in nature	Unidirectional in nature	Unidirectional or Bidirectional in nature

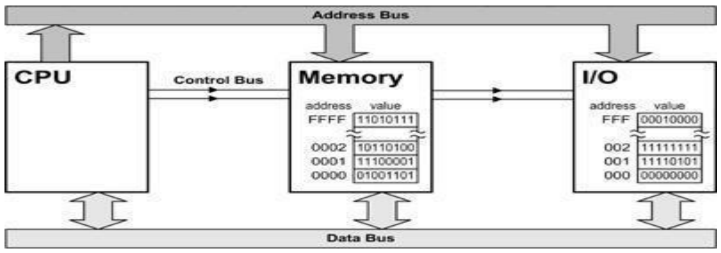


Fig 2: Types of Buses

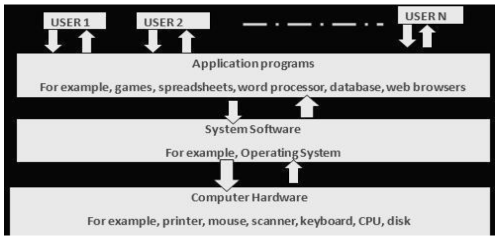
2. **I/O Buses:** The buses which are used to connect all I/O devices with CPU and Memory are called I/O buses. These are classified into following three types.

PCI Bus	ISA Bus	USB Bus
PCI stands for ***Peripheral Component Interconnect***	ISA stands for ***Industry Standard Architecture***	USB stands for ***Universal Serial Bus***
The motherboard will be having 3 or 4 PCI connectors, so that we can insert various chips.	This is simple and slowest bus used in IBM PCs	It helps to connect various I/O devices like keyboard, mouse, pen drives, printer, etc.
Fastest and presently more powerful bus	Oldest, simplest and slowest Bus	Newest and widely used bus

Main Board or Mother Board: Mother Board is a set of Integrated Chips (**ICs**) which are designed to work together. It controls the flow of data/instructions within our computer. It is the main board on which other hardware components are connected to enable the computer system to work as an integrated unit. It consists of sockets, slots, power connectors and bus.

Chip sets: Chip set is the set of integrated chips that are designed to work together. These set of chips controls the flow of information on computer. The chips may be controllers for memory, cache, hard drive, key board and peripherals.

Operating System and its types: An Operating System (OS) is system software that **controls** and **supervCSEs** the **hardware components** of a computer system and it provides the services to computer users. Also called as ***Resource Manager*** that manages the resources such as CPU, Memory, I/O devices, Job/Task/Process etc., a computer cannot run without it. The major functions of OS includes: ***CPU Management, Memory Management, File Management, Device Management, Process/Task/Job Management and Security Management.***


USER 1
USER 2
USER N
Application programs
For example, games, spreadsheets, word processor, database, web browsers
System Software
For example, Operating System
Computer Hardware
For example, printer, mouse, scanner, keyboard, CPU, disk

The primary goal of an OS is to make the computer system c*onvenient and efficient to use.* An OS ensures that the system resources (such as CPU, memory, I/O devices, etc) are utilized efficiently. For example, there may be many programs residing in the main memory. Therefore, the system needs to determine which programs are active and which need to wait for some I/O operation.

Some of the examples of Operating Systems:

Windows –XP is an O.S. is used for Personal Computers (PCs) Unix and XENIX are the OSs used for multi-user computers. Windows 7, Windows 8, Macintosh OS, Fedora, and Android, etc.

Types of Operating Systems: The operating systems are classified into 7 types based on their capability and usage.

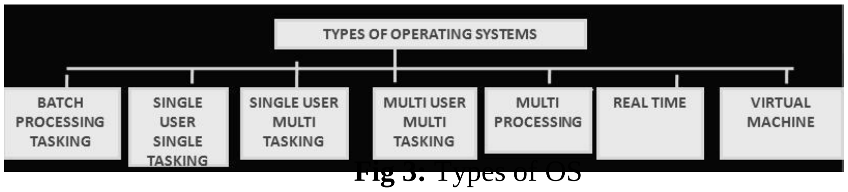


Fig 3: Types of OS

Batch Processing Tasking OS: The data is collected into a group called batch and provides only one batch (one after another) of jobs as input to the computer system at a time. The jobs in a batch are processed on first come first serve basis. In this type, the process takes place at specified time intervals i.e. weekly or monthly without user interaction. E.g. Punch cards were using to store the data in batch processing and in payroll preparation in a business batch processing was helpful.

Single user and single tasking OS: The OS that allows only one program to execute at a time is called single user single tasking operating system. Using this operating system user can do only one task at a time. **E.g.** DOS (Disk Operating System).

Single user and multi tasking OS: The OS that allows a single use to perform more than one task at a time is called single user multi tasking operating system. While working with the Ms-Word user can perform other work like print a document, listen music.**E.g.** Windows-XP, Windows Vista, Windows – 7, etc.

Multi user and multitasking OS: The O.S. that allows two or more users to use a main computer system to do more than one task is called multiuser and multitasking operating system.**E.g.** UNIX is a multiuser and multitasking operating system.

Multiprocessing OS: The OS that allows multiple programs to be executed by multiple CPUs

(Processors) is called multiprocessing operating system. Super and main frame computers have more than one CPU and multiprocessing operating system.

Real Time Operating System (RTOS): The OS that is used for real time applications and to carry out certain calculations within the specified time constraint. This OS is used in applications such as mobile phones, supporting systems in hospitals, nuclear power plants, oil refining, chemical processing, environmental applications and air-traffic control systems, disaster management etc.,

Virtual machine OS: Allows several users of a computer system to operate as if each has the only terminal attached to the computer.

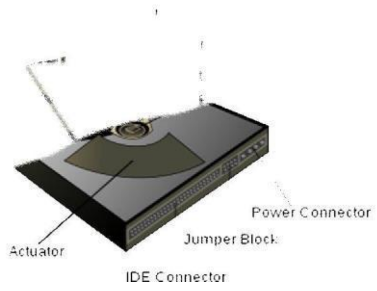
Random Access Memory (RAM): RAM is basically main memory of the computer**.** RAM is a semiconductor memory made up of small memory chips that form a memory module. These modules are installed in the RAM slots on the motherboard of computer. Every time you open a program, it gets loaded from the hard drive into the RAM. This is because reading data from the RAM is much faster than reading data from the hard drive.

Synchronous Dynamic Random Access Memory (SDRAM): It is an improvement to standard DRAM because it retrieves data alternately between two sets of memory. This eliminates the delay caused when one bank of memory addresses is shut down while another is prepared for reading. It is called "Synchronous" DRAM because the memory is synchronized with the clock speed that the computer's CPU bus speed is 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 used for digital cameras, cellular phones, networking hardware, and PC cards.

Hard disks: Hard disk is prime unit of storage of the computer. Huge amount of data can be stored and accessed in few millCSEconds. The hard disk consists of more number of disks arranged in the cylindrical order, one above another on a spindle.

The read/write heads are attached to single access mechanism so that they cannot move independently. All read/write heads are moved together to position that heads on the required track. The hard disks available today ranges from 200 GB to 2TB and so on. The present day hard disk ranges from 3600 rpm to more than 10000 rpm and so on.



Advantages: High storage capacity, high data accessing rate and permanent storage medium.

Disadvantages: It is not portable.

Optical media: An optical storage media is kind of storage, which is coated with thin metal on which bits are stored. The data can be stored in to optical storage media or read form the optical storage media.

The devices which perform read or write operation on optical storage media are called optical storage media. The laser technology is used to read the data or write the data on optical storage devices.

Examples: CD-ROM, DVD etc.

Compact Disc Read-Only-Memory (CD-ROM): It is a type of optical disc that uses laser technology to read and write data on the disc. The information stored on CDROM becomes permanent and cannot be altered. This means that the stored information can only be read for processing**.**

A CD-ROM uses the round shaped optical disk to store data, applications, games and audio files. It can store up to 700 MB of data. It has become integral part of every organization due to its features like reliability, reasonable, storage capacity and easy to use of carry.

CD-Drive will be with motor to rotate the disks to perform read and write operations. A CD-drive will
consists of the components like Disc drive, disk drive motor, laser pick up assembly tracking drive and tracking motor and so on.

Compact Disk Recordable (CD-R): The CD-R allows you to create your own CD.CD-R drives have the ability to create CDs but they can write data on the disk only once.CD-R technology also called as Write Once-Read much (WORM) technology. Laser technology is used to write the data on the compact disk.CD-R drives come in IDE, SCSI and USB models.

Compact Disc Rewritable (CD-RW): CD-RW is an erasable optical disk which is used to write data multiple times on a disk, CD-RW disks are good for data backup, data archiving or data distribution on CDs. The disk normally holds 700MB of data. Technology to write data multiple times on a CD was known as the Phase change Dual (PD) technology. The reflective properties of a CD-RW are different than regular CD-ROM disks.

Disk or Digital Versatile Disc (DVD-ROM): A DVD is a small optical disk having high density medium and capable of storing a full-length movie on a single disk. The high density is achieved by using both sides of the disk, special data-compression technology, and extremely small tracks to store the data.

Advantages: Storage capacity is more compared to CDs.

Flash Drives (Pen drives): USB flash drives are removable, rewritable, and physically much smaller drives weighing even less than 30 g. A flash drive consists of a small printed circuit board carrying the circuit elements and a USB connector, insulated electrically and protected inside a plastic, metal, or rubberized case which can be carried in a pocket or on a key chain.

Advantages

Data stored on flash drives is impervious to scratches and dust
Mechanically very robust
Easily portable
Have higher data capacity than any other removable media. Compared to hard drives, flash drives use little power
Flash drives are small and light-weight devices
Flash drives can be used without installing device drivers.

Disadvantages

Can sustain only a limited number of write and erase cycles before the drive fails. Most flash drives do not have a write-protect mechanism

Flash drives are very small devices that can easily be misplaced, left behind, or otherwCSE lost.
The cost per unit of storage in a flash drive is higher than that of hard disks

Keyboard: A keyboard is the primary input device used in all computers. Keyboard has a group of switches resembling the keys on an 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 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.

Advantages:

Easy to use; Cheap; Can be used to quickly place the cursor anywhere on the screen Helps to quickly and easily draw figures

Point and click capabilities makes it unnecessary to remember certain commands

Disadvantages:

Needs extra desk space to be placed and moved easily

The ball in the mechanical mouse needs to be cleaned very often for smooth movements

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.

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:

The process of working through details of a problem to reach a solution. There are three approaches to problem solving:

1. **Algorithm**
2. **Flowchart**
3. **Pseudo Code**

Algorithm: The algorithm is a ***step-by-step procedure*** to be followed in solving a problem. It provides a scheme to solve a particular problem in ***finite number of unambiguous steps***. It helps in implementing the solution of a problem using any of the *programming languages.*

In order to qualify as an algorithm, a sequence of instructions must possess the following characteristics:

Definiteness: Instructions must be ***precCSE*** *and* ***unambiguous*** i.e. each and every instruction should be clear and should have only one meaning.

Finiteness: Not even a single instruction must be ***repeated infinitely***. i.e., each instruction should be performed in finite time.

Termination: After the algorithm gets executed, the user should get the desired ***result***

Key features of an algorithm:

Any algorithm has a finite number of steps and some steps may involve decision making, repetition. Broadly speaking, an algorithm exhibits three key features that can be given as:

Sequence: Sequence means that 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.

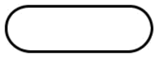
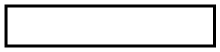
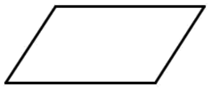
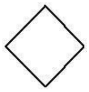
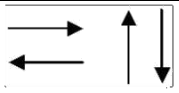
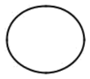
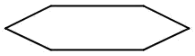
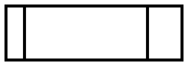
Repetition: Repetition which involves executing one or more steps for a number of times can be implemented using constructs like the while, do-while and for loops. These loops executed one or more steps until some condition is true.

Example: To compute the Area of Rectangle

ALGM: AREA_of_RECTANGLE [This algorithm takes length and breadth, the sides of the rectangle as input and computes the area of rectangle using the formula ***area=length * breadth***. Finally it prints the area of rectangle]

```
STEPS:
Step 1:[Initialize]
        Start
Step 2: [Input the sides of Rectangle]
        Read length, breadth
Step 3:[Compute the area of rectangle]
        Area←length*breadth
Step 4:[Display the Area]
        Print Area
Step 5: [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.

Symbol	Name	Description
	oval	Represents the terminal point
	Rectangle	Represents the process steps defined in algorithm
	Parallelogram	Indicate the reading Operation used for input/output or data or information from/to any device
	Diamond	Indicates the decisions (questions) and consequently branch points or the paths to be followed based on the result of the question
	Arrows	Shows the flowchart direction and connects the various flow chart symbols.
	Small circle	Shows the continuation from one point in the process flow to another.
	Hexagon	Represents Looping structures
	Process	Subroutine function

Advantages of Flowcharts:

A flowchart is a diagrammatic representation that illustrates the sequence of steps that must be performed to solve a problem. They are usually drawn in the early stages of formulating computer solutions to facilitate communication between programmers and business people.

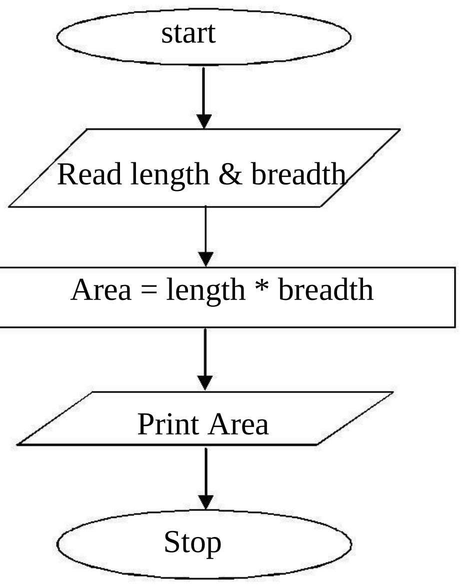
Flowcharts help programmers to understand the logic of complicated and lengthy problems. They help to analyze the problem in a more effective manner

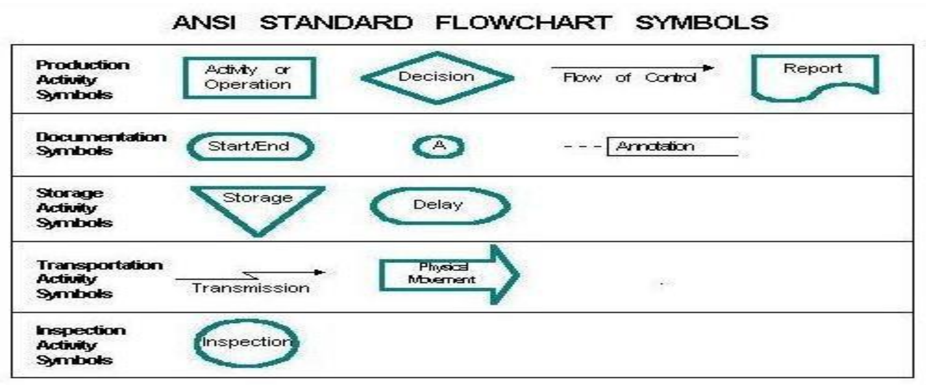
Flowchart can be used to debug programs that have error(s).

E.g.: To compute the Area of Rectangle

Limitations of using Flowcharts:

Drawing flowcharts is a laborious and a time consuming activity. Flowchart of a complex program becomes, complex and clumsy. At times, a little bit of alteration in the solution may require complete re-drawing of the flowchart Essentials of what is done may get lost in the technical details of how it is done. There are no well-defined standards that limits the details that must be incorporated in a flowchart





Pseudo code: It is a form of structured English that describes algorithms. It facilitates the designers to focus on the logic of the algorithm without getting bogged down by the details of language syntax. ***Pseudocode*** is a compact and informal high-level description of an algorithm that uses the structural conventions of a programming language. It is meant for human reading rather than machine reading, so it omits the details that are not essential for humans. Such details include keywords, variable declarations, system-specific code and subroutines. There are no standards defined for writing a ***pseudocode*** because it is not an executable program. Flowcharts can be considered as a graphical alternative to ***pseudocode***, but are more spacious on paper.

E.g.: To compute the area of Rectangle

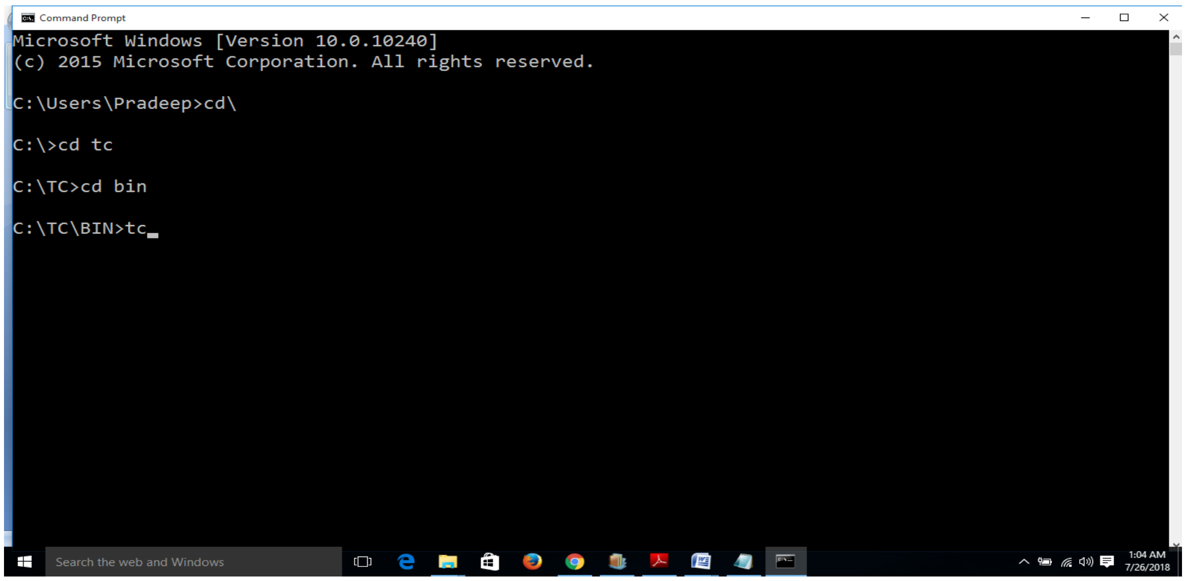
```
    Begin
    Input length, breadth
    Area=length*breadth
    Print Area
End
```

Laboratory Program-1

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

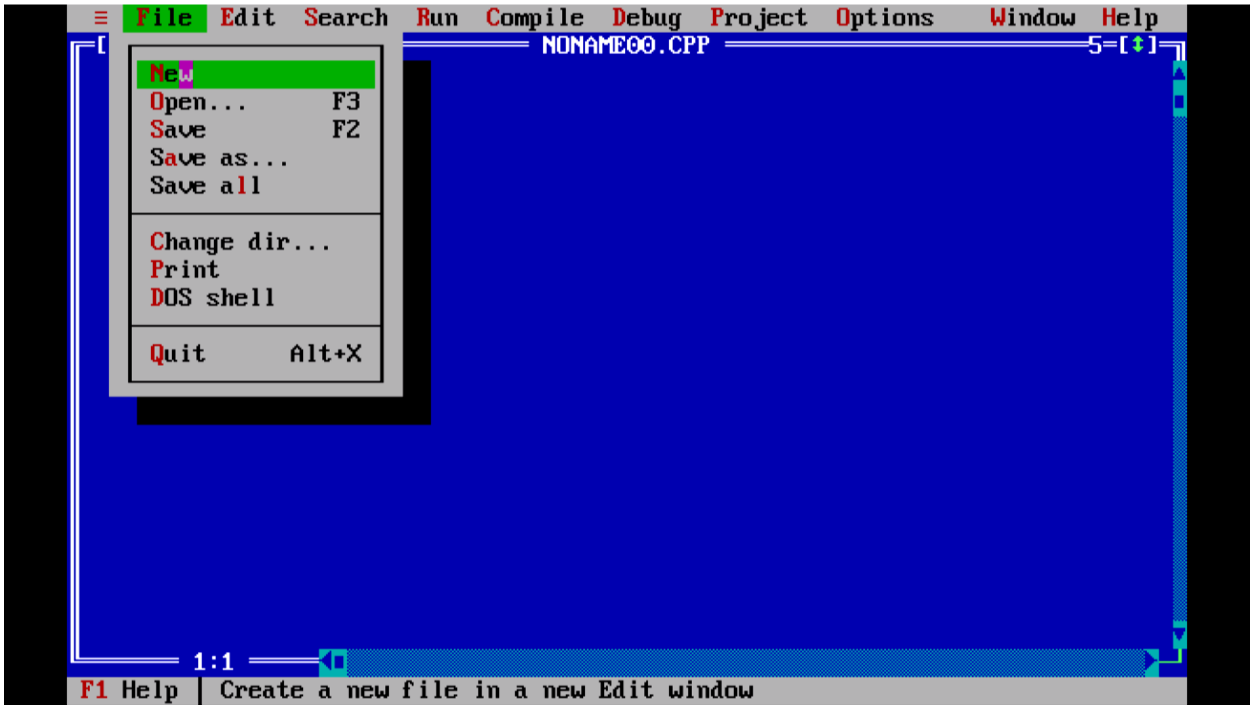
Working with TurboC

Step 1: Locate the TC.exe file and open it. You will find it at location **C:\TC\BIN\.**

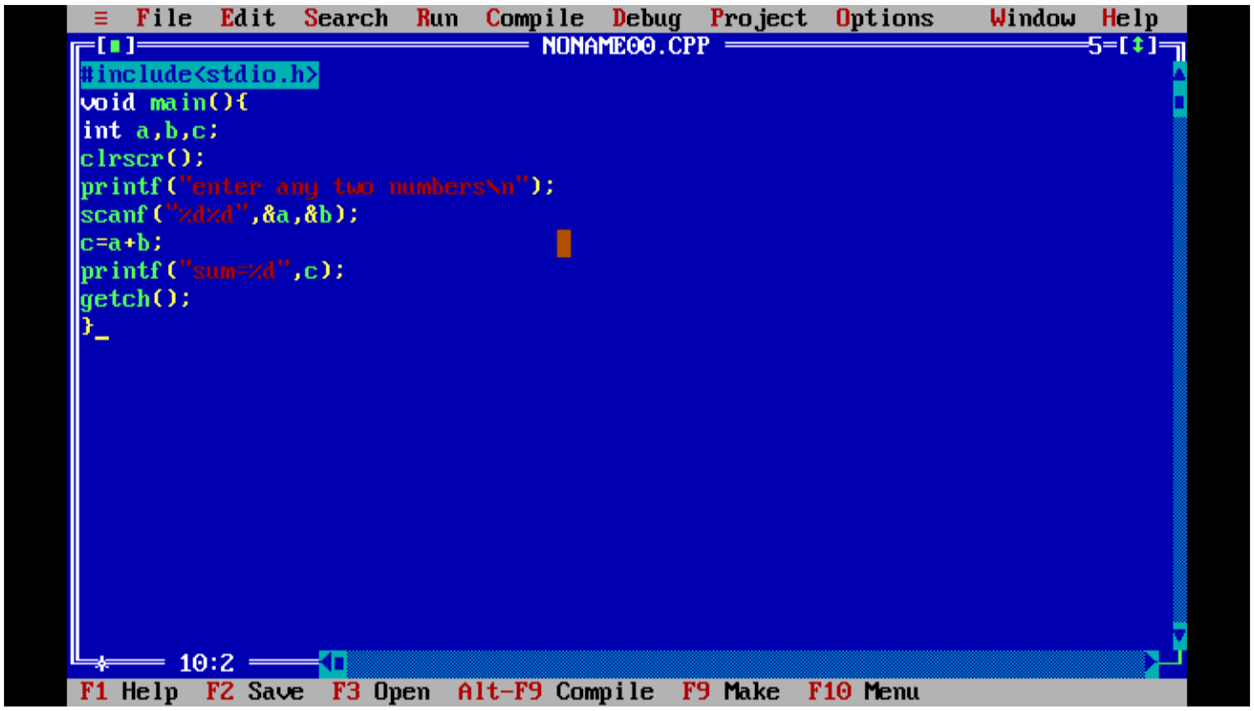


```
Microsoft Windows [Version 10.0.10240]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\Pradeep>cd\

C:\>cd tc

C:\TC>cd bin

C:\TC\BIN>tc_
```

Step 2: File > New (as shown in the below picture) and then write your C program

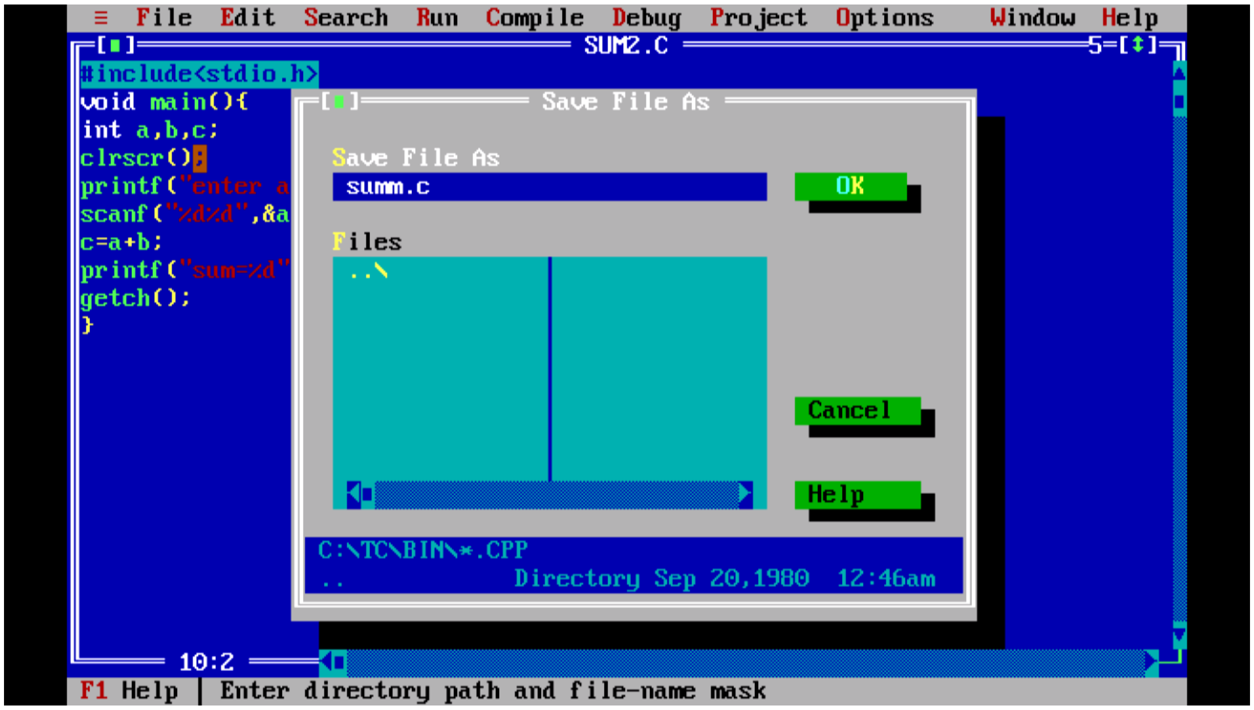


≡ File Edit Search Run Compile Debug Project Options Window Help
NONAME00.CPP
New
Open... F3
Save F2
Save as...
Save all
Change dir...
Print
DOS shell
Quit Alt+X
1:1
F1 Help | Create a new file in a new Edit window


≡ File Edit Search Run Compile Debug Project Options Window Help
NONAME00.CPP
10:2
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

```
#include<stdio.h>
void main(){
int a,b,c;
clrscr();
printf("enter any two numbers\n");
scanf("%d%d",&a,&b);
c=a+b;
printf("sum=%d",c);
getch();
}
```

Step 3: Save the program using F2 (OR file > Save), remember the extension should be ".c". In the below screenshot I have given the name as summ.c.

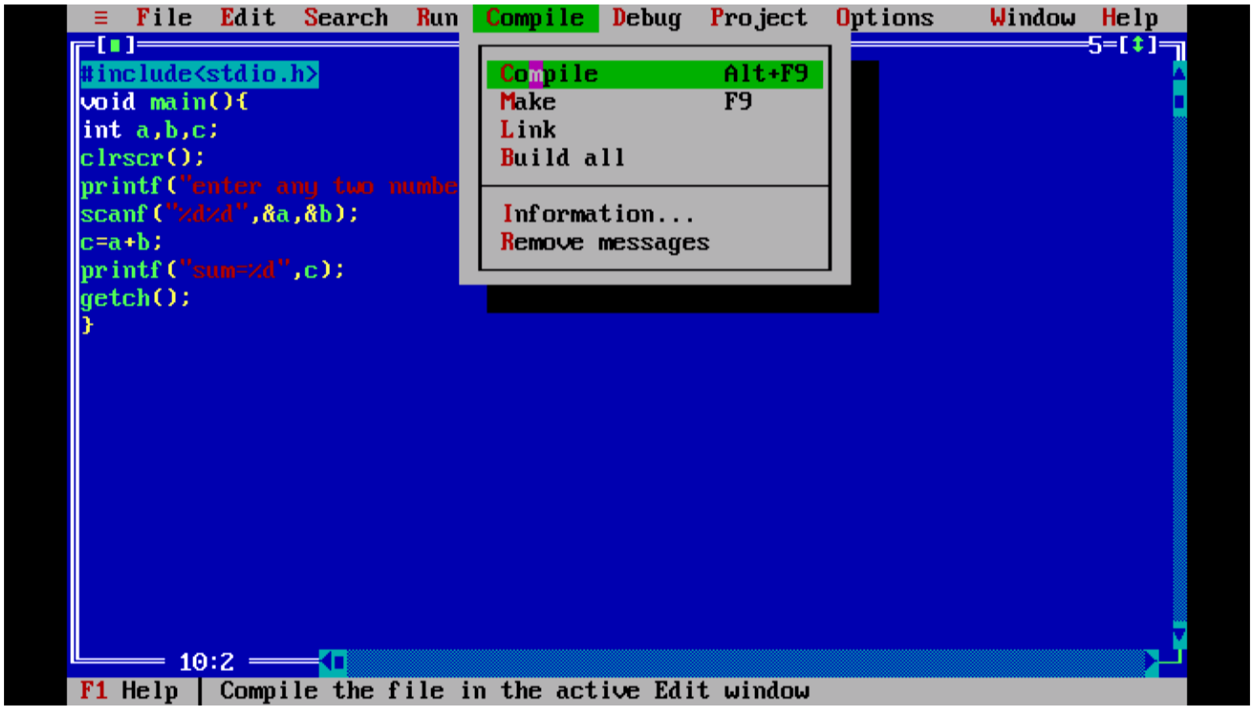

≡ File Edit Search Run Compile Debug Project Options Window Help
SUM2.C
Save File As
OK
Files
Cancel
Help
10:2

Save File As: summ.c

C:\TC\BIN\*.CPP
.. Directory Sep 20,1980 12:46am

F1 Help | Enter directory path and file-name mask

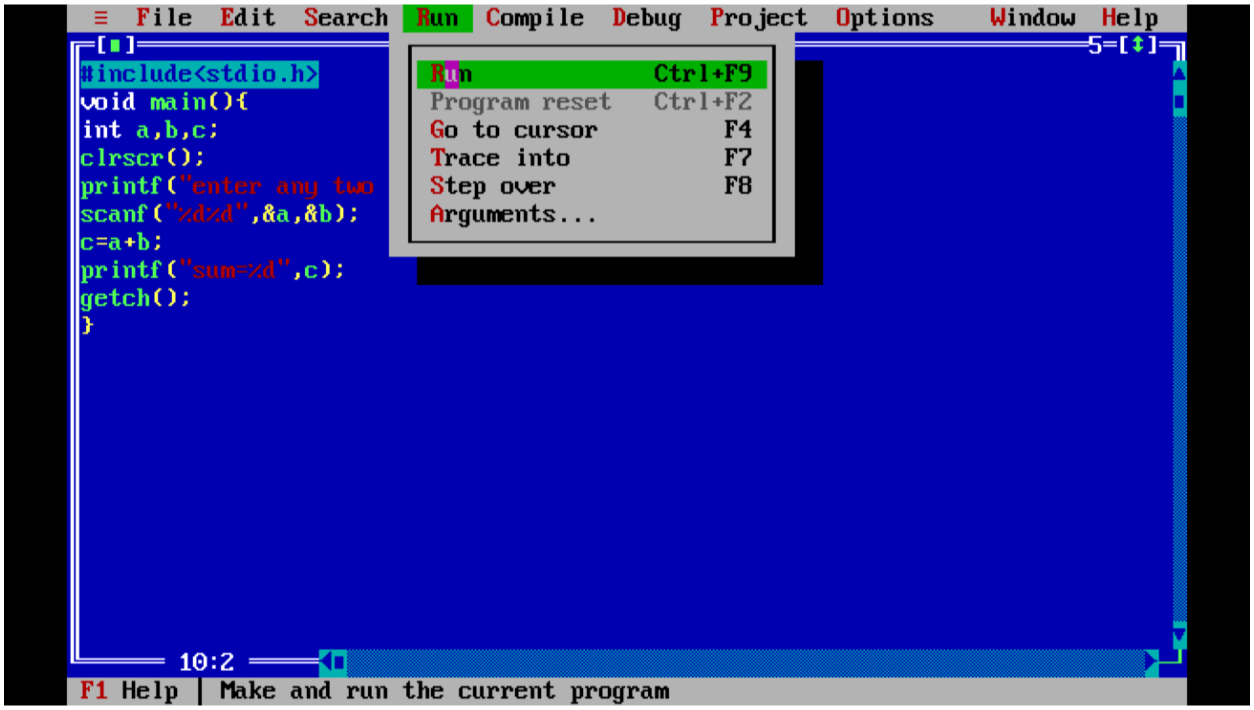
Step 4: Compile the program using Alt + F9 **OR** Compile > Compile (as shown in the below screenshot).



```
#include<stdio.h>
void main(){
int a,b,c;
clrscr();
printf("enter any two numbe
scanf("%d%d",&a,&b);
c=a+b;
printf("sum=%d",c);
getch();
}
```

Compile the file in the active Edit window

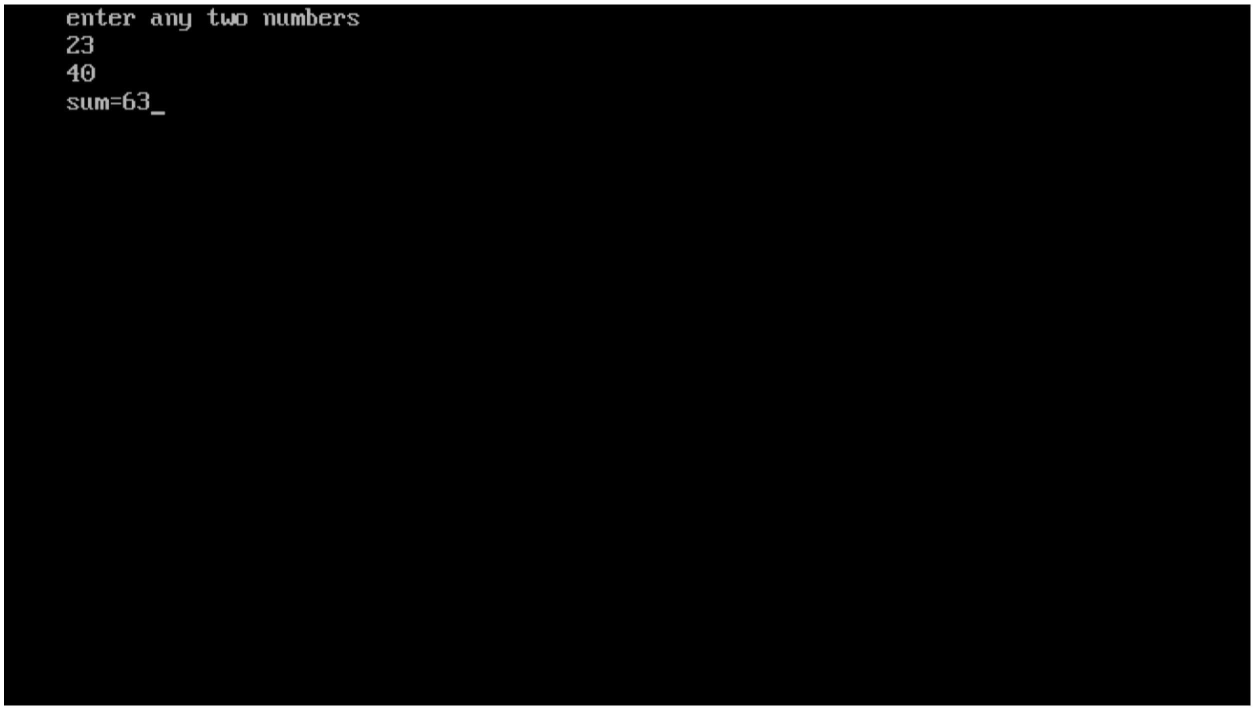
Step 5: Press Ctrl + F9 to Run (or select Run > Run in menu bar) the C program.



```
#include<stdio.h>
void main(){
int a,b,c;
clrscr();
printf("enter any two
scanf("%d%d",&a,&b);
c=a+b;
printf("sum=%d",c);
getch();
}
```

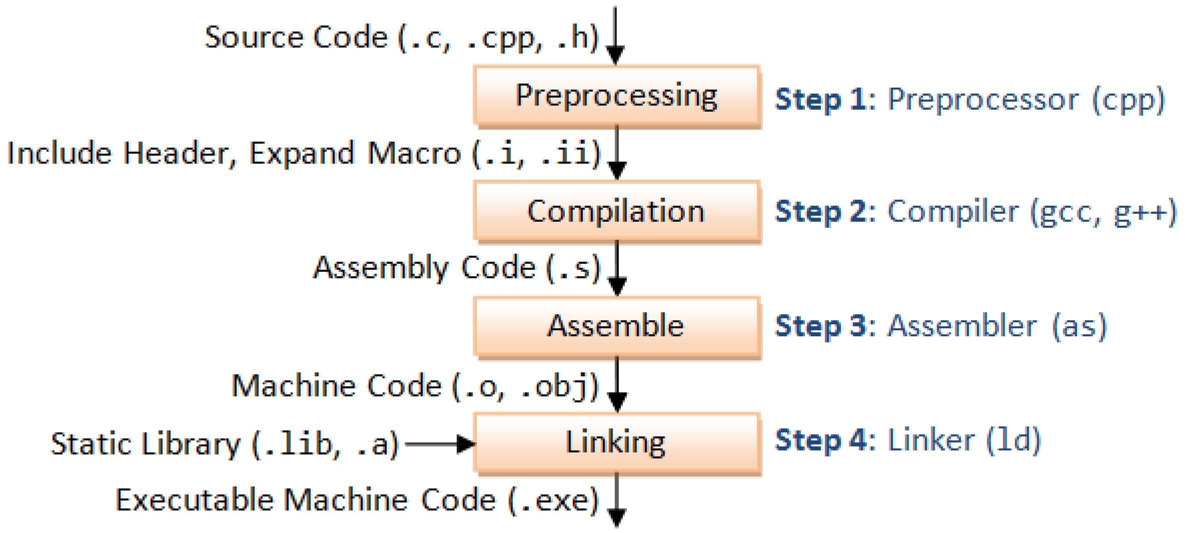
Make and run the current program

Step 6: Alt+F5 to view the output of the program at the output screen.



```
enter any two numbers
23
40
sum=63_
```

GCC Compilation Process



Source Code (.c, .cpp, .h)
Preprocessing
Step 1: Preprocessor (cpp)
Include Header, Expand Macro (.i, .ii)
Compilation
Step 2: Compiler (gcc, g++)
Assembly Code (.s)
Assemble
Step 3: Assembler (as)
Machine Code (.o, .obj)
Static Library (.lib, .a)
Linking
Step 4: Linker (ld)
Executable Machine Code (.exe)

PART-A

Laboratory Program 2

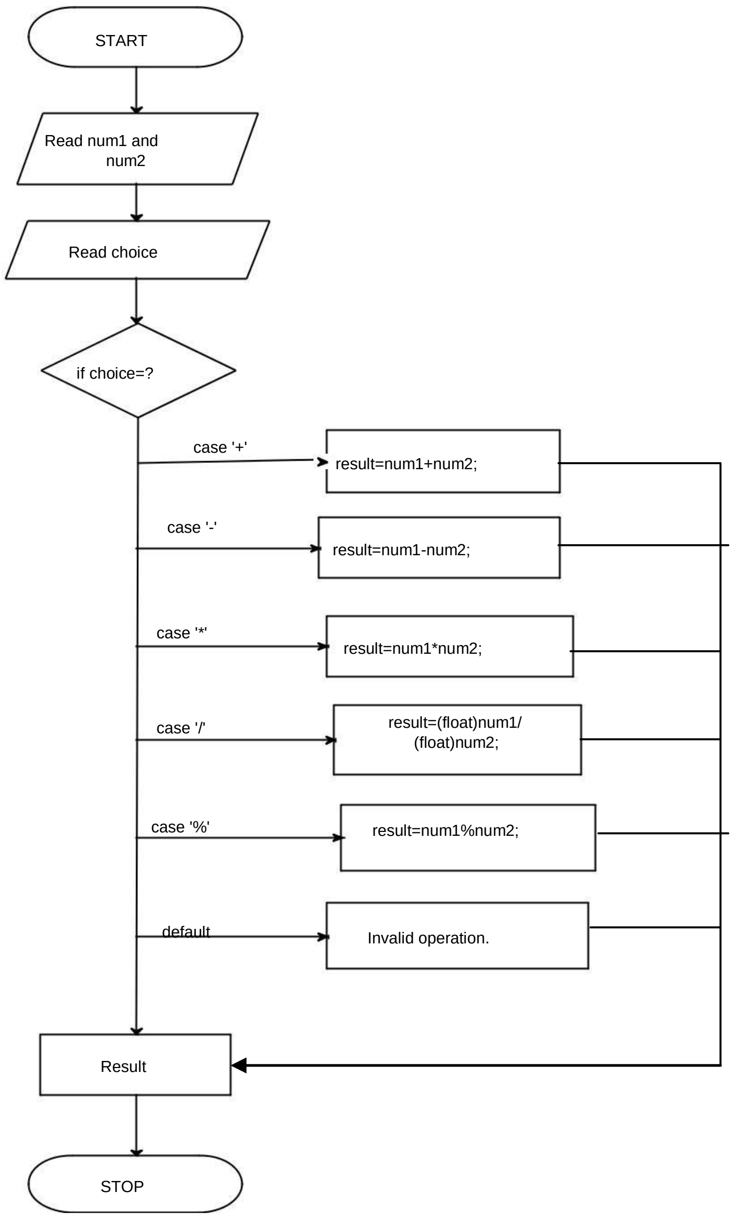
<u>Develop a program to solve simple computational problems using arithmetic expressions and use of each operator leading to simulation of a commercial calculator. (No built-in math function).</u>

2.1 ALGORITHM

```
PURPOSE : To simulate a calculator using arithmetic expressions
INPUT: Enter num1 and num2
OUTPUT: Print the result of addition or subtraction or multiplication or division or modulus.
START
Step 1: [Enter first number]
            read num1
Step 2: [Enter Second number]
           read num2
Step 3:[Enter Choice]
           read choice
Step 4:[To perform addition]
           if choice is equal to plus
           add num1 and num2
           print result
Step 5: [To perform subtraction]
           if choice is equal to minus
           subtract   num2 from num1
           print result
Step 6: [To perform multiplication]
            if choice is equal to multiplication
            multiply num1 and num2
            print result
Step 7: [To perform division]
            if choice is equal to division
```

```
        divide num1 by num2
        print result
Step 8: [To perform Modulus]
      if choice is equal to modulus
      divide num1 by num2
      print result (remainder)
STOP
```

2. 2 FLOWCHART



2.3 PROGRAM

```
#include <stdio.h>
 int main()
{
    int num1,num2;
    float result;
    char choice;    //to store operator choice

    printf("Enter first number: ");
    scanf("%d",&num1);
    printf("Enter second number: ");
    scanf("%d",&num2);
    printf("Choose operation to perform (+,-,*,/,%): ");
    scanf(" %c",&choice);
    result=0;
    switch(choice)
    {
        case '+':
            result=num1+num2;
            break;
        case '-':
            result=num1-num2;
            break;
        case '*':
            result=num1*num2;
            break;
       case '/':
            result=(float)num1/(float)num2;
            break;
       case '%':
            result=num1%num2;
            break;
        default:
            printf("Invalid operation.\n");
    }
   printf("Result: %d %c %d = %f\n",num1,ch,num2,result);
   return 0;
}
```

****

OUTPUT:

```
    First run:
    Enter first number: 10
    Enter second number: 20
    Choose operation to perform (+,-,*,/,%): +
    Result: 10 + 20 = 30.000000

    Second run:
    Enter first number: 10
    Enter second number: 3
    Choose operation to perform (+,-,*,/,%): /
    Result: 10 / 3 = 3.333333

    Third run:
    Enter first number: 10
    Enter second number: 3
    Choose operation to perform (+,-,*,/,%): >
    Invalid operation.
    Result: 10 > 3 = 0.000000
```

****

VIVA QUESTIONS:

1) What is Switch statement?
2) How many cases can you have in switch statement?
3) How switch statement works?
4) What does break in switch statement indicate?
5) What is a case in a switch statement?

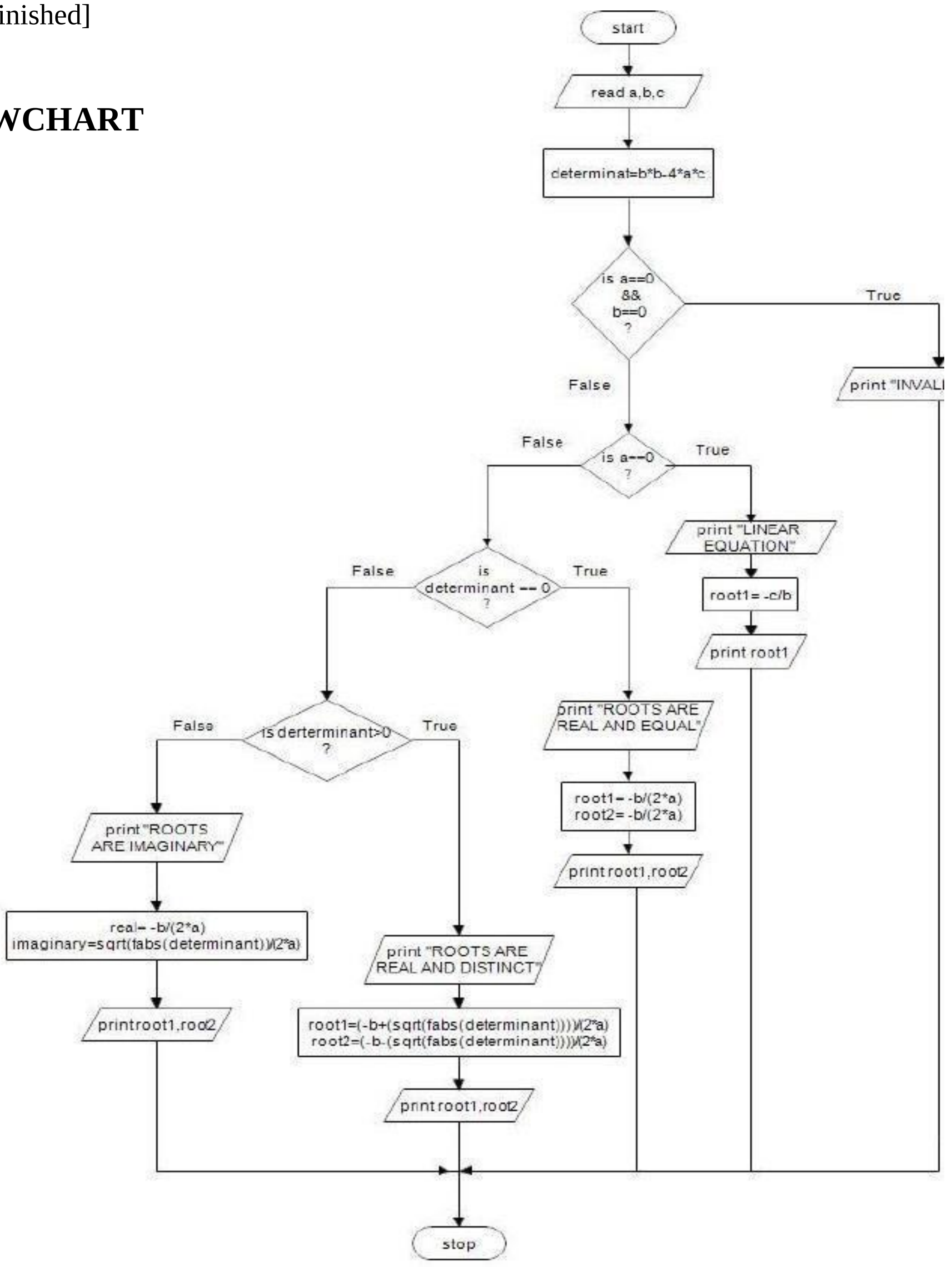
Laboratory Program 3

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

3.1 ALGORITHM

```
Purpose : To find roots of a given quadratic equation.
Input: Coefficients of quadratic equation a, b, c
Output: Root1 and Root2
START
STEP 1: [Input the values of a, b, c]
            read a, b, c
STEP 2: [Calculate the determinant]
            determinant = b*b-4*a*c
STEP 3: [Check for validity]
           If a is equal to 0 and b is equal to 0
           print "Invalid Inputs"
STEP 4: [Check for different roots]
           If a is equal to 0
           print "Linear equation"
           Root1=-c/b
           Print "Root1"
STEP 5: [Check for real and equal roots]
            If determinant is equal to 0
            print "Roots are real and equal"
            Root1= -b/(2*a)
            Root2 = -b/(2*a)
            Print "Root1 & Root2"
STEP 6: [Check for real and distinct roots]
            If determinant is greater than 0
            Then print "Roots are real and distinct"
            Root1= (-b+ (sqrt (fabs (determinant))))/(2*a)
            Root2= (-b-(sqrt (fabs (determinant))))/(2*a)
            Print "root1 and root2"
            End if
STEP 7: [Check for imaginary roots]
            print "Roots are imaginary"
            Real=-b/ (2*a)
            Imaginary=sqrt (fabs (determinant))/ (2*a)
            print "Root1 and R oot2"
STEP 8: [Finished]
STOP
```

3.2 FLOWCHART



3.3 PROGRAM

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
void main()
{
        float a,b,c;
        float root1,root2;
        float determinant,real,imaginary;
        clrscr();
        printf("Enter the Co-efficient of Quadratic Equation\n");
        scanf("%f%f%f",&a,&b,&c);
        determinant=b*b-4*a*c;
        if(a==0 && b==0)
        {
           printf("INVALID INPUTS\n");
           getch();
        }
        else if(a==0)
        {
            printf("LINEAR EQUATION\n");
             root1=-c/b;
             printf("ROOT=%f\n",root1);
         }
        else if(determinant==0)
        {
            printf("ROOTS ARE REAL AND EQUAL\n");
            root1=-b/(2*a);
            root2=-b/(2*a);
           printf("Root1=%f\n Root2=%f",root1,root2);
        }
        else if(determinant>0)
        {
              printf("ROOTS ARE REAL AND DISTINCT\n");
              root1=(-b+(sqrt(fabs(determinant))))/(2*a);
              root2=(-b-(sqrt(fabs(determinant))))/(2*a);
              printf("ROOT1=%f\n ROOT2=%f",root1,root2);
         }
         else
        {
              printf("ROOTS ARE IMAGINARY\n");
              real=-b/(2*a);
              imaginary=sqrt(fabs(determinant))/(2*a);
              printf("ROOT1=%f+i%f\n",real,imaginary);
              printf("ROOT2=%f-i%f\n",real,imaginary);
getch();
}
```

****

Output 1:

```
Enter the Co-efficient of Quadratic Equation
000
Invalid Coefficients
```

Output 2:

```
Enter the Co-efficient of Quadratic Equation
0 21
Root1=-0.5
```

Output 3:

```
Enter the Co-efficient of Quadratic Equation
121
Roots are Real and Equal
Root1=-1.0000
Root2=-1.0000
```

Output 4:

```
Enter the Co-efficient of Quadratic Equation
189
Roots are Real and Distinct
Root1=-1.354
Root2=-6.646
```

Output 5:

```
Enter the Co-efficient of Quadratic Equation
123
ROOTS ARE IMAGINARY
ROOT1=-1.000+i1.414
ROOT2=-1.000-i1.414
```

****

Viva Questions:

1) What is quadratic Equation?
2) What is math.h?
3) What are decision making capabilities of C language?
4) Write the syntax of " if "statement?
5) Difference between if and switch statements?
6) Write an unconditional control statement in C.
7) Write a flowchart for ' if' conditional construct?

Laboratory Program 4

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

4.1 ALGORITHM

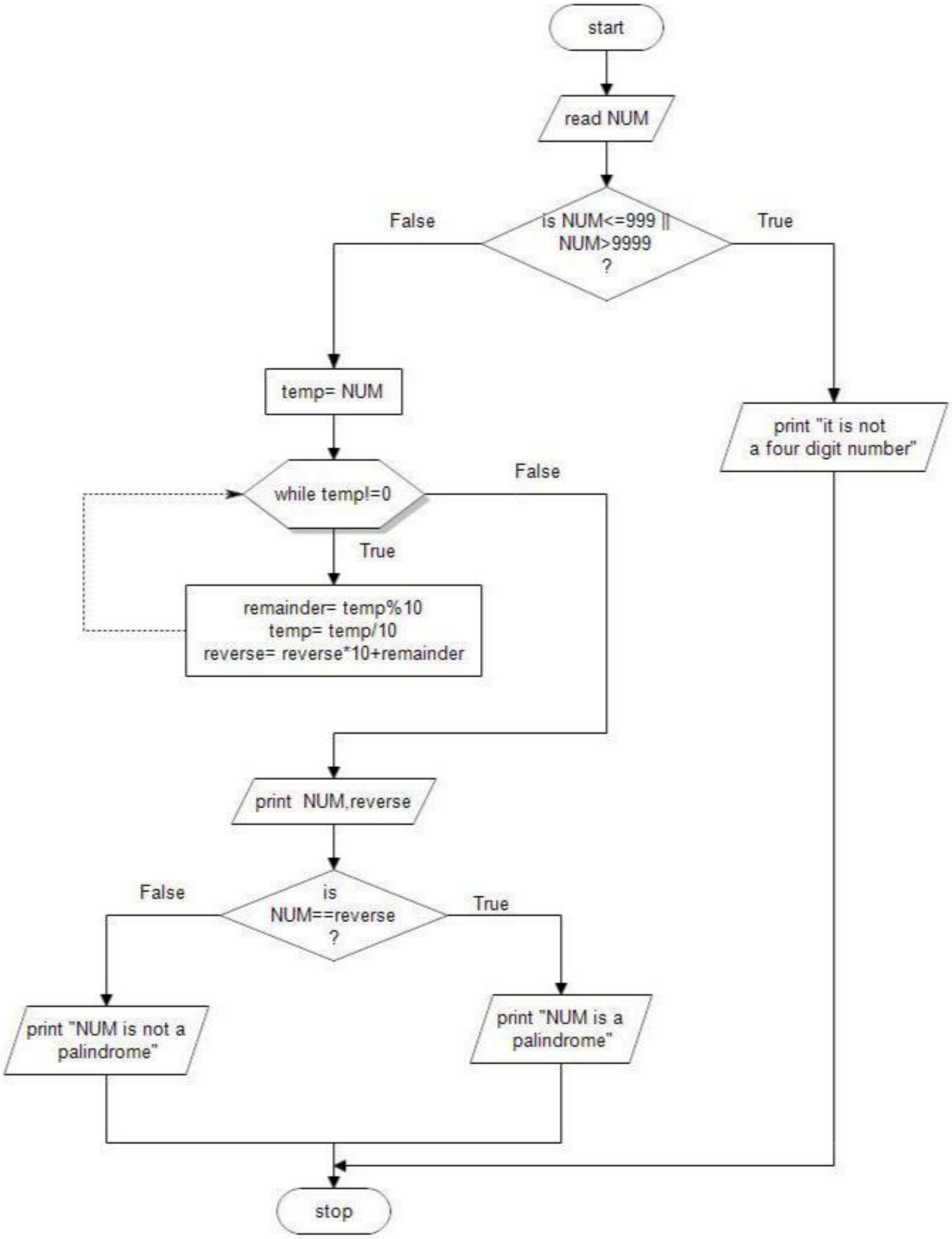
Purpose : To check the given integer is palindrome or not

Input: Num

Output : Reverse and palindrome or not

```
START
STEP 1: [Input a number]
            read num
STEP 2: [To check the given number is 4 digits or not]
           If the number less than 999 or greater than 9999
           print “it is not a four digit number”
           Goto step 5
STEP 3: [Calculate the reverse of given number]
           Temp=num
            Loop while temp not equal to 0
            Remainder=temp%10
            Temp=temp/10
            Reverse=reverse*10+remainder
           print “reverse number”
STEP 4: [Check number is a palindrome or not]
           If Num is equal to reverse
           print “Num is palindrome “
           Else
           print “Num is not palindrome”
STEP 5: [Finished]
STOP
```

4.2 FLOWCHART


start
read NUM
is NUM<=999 || NUM>9999 ?
False
True
temp= NUM
print "it is not a four digit number"
while temp!=0
False
True
remainder= temp%10
temp= temp/10
reverse= reverse*10+remainder
print NUM,reverse
is NUM==reverse ?
False
True
print "NUM is not a palindrome"
print "NUM is a palindrome"
stop

4.3 PROGRAM

```
#include<stdio.h>
#include<conio.h>
void main()
{
     int NUM,reverse=0,temp,remainder;
     clrscr();
     printf("Enter a number to check if it is a palindrome or not  \n");
     scanf("%d",&NUM);
     if(NUM<=999 || NUM>9999)
    {
            printf("Enter Four Digit Number \n");
            getch();
            exit(0);
    }
    temp=NUM;
   while (temp!=0)
  {
            remainder=temp%10;
            temp=temp/10;
            reverse=reverse*10+remainder;
 }
printf("Reverse of %d is %d \n",NUM,reverse);
if(NUM==reverse)
{
            printf("%d is a palindrome number \n", NUM);
            else
            printf("%d is not a palindrome number \n", NUM);
}
getch();
}
```

Output 1:

```
Enter a number to check if it is a palindrome or not
123
Enter Four digit number
```

Output 2:

```
Enter a number to check if it is a palindrome or not
1234
1234 is not a palindrome number
```

Output 3:

```
Enter a number to check if it is a palindrome or not
1221
1221 is a palindrome number
```

****

Viva Questions:

1. What are Looping control statements?
2. Explain while loop.
3. What is the difference between while and for loops?
4. What are the Entry controlled and Exit controlled loops in C?
5. Write the flowchart for ‘while’ loop.

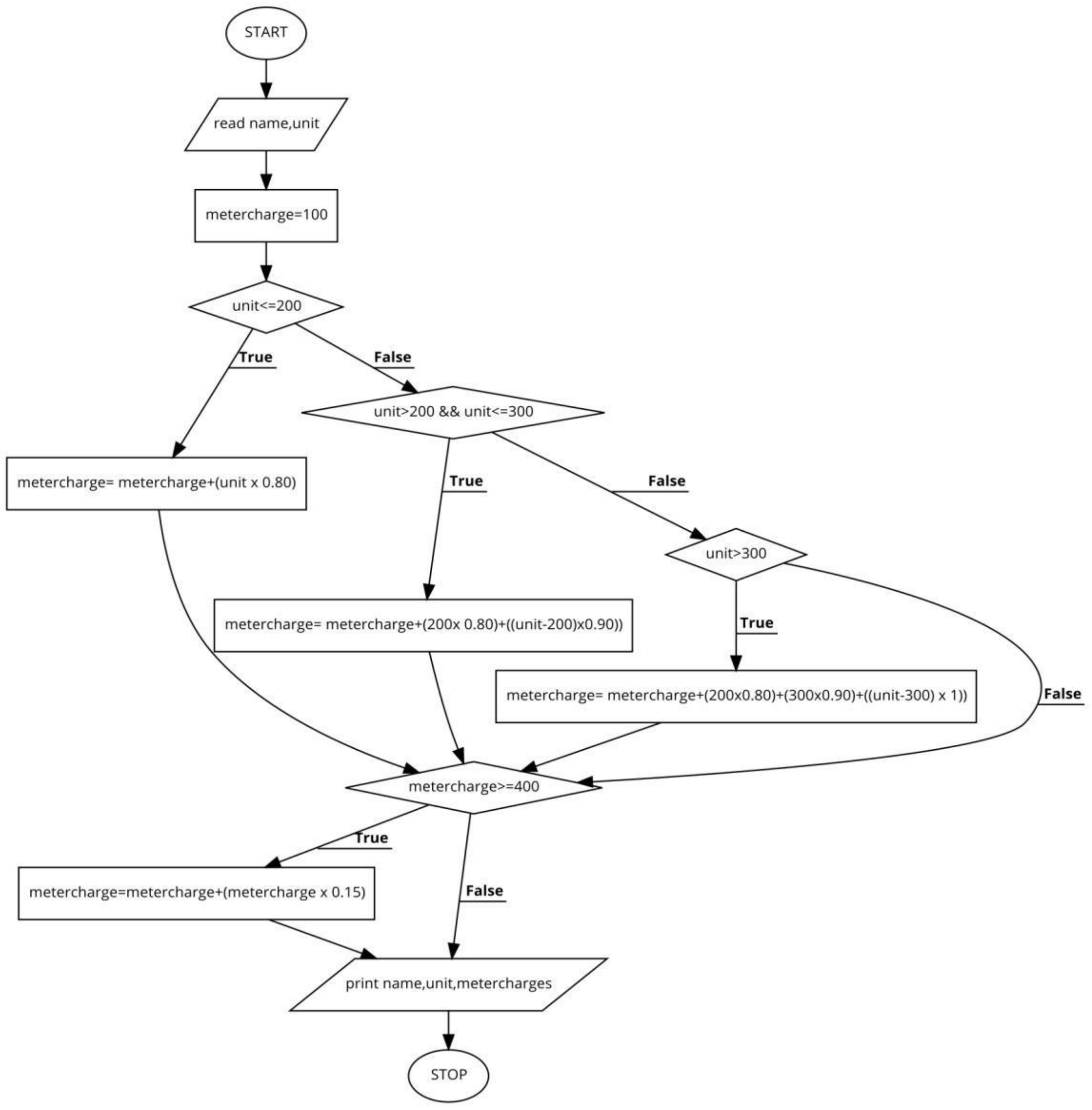
Laboratory Program 5

An electricity board charges the following rates for the use of electricity: for the first 200 units 80 paCSE per unit: for the next 100 units 90 paCSE 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.

5.1 ALGORITHM

```
PURPOSE: Read The Name Of The User, Number Of Units Consumed And Print
INPUT: name [10], unit
OUTPUT: Print the charges for total number of units consumed
START
STEP 1: [Input a name and units]
        read name and unit
STEP 2: [Initialization]
        Metercharge=100
STEP3: [To check electricity unit is less than or equal to 200 and calculate
       metercharge] If unit less than or equal to 200
       metercharge= metercharge+(unit*.80)
STEP 4: [Else check unit is greater than 200 and greater than 300 and calculate
       metercharge] If unit greater than 200 and unit greater than or equal to 300
       metercharge= metercharge+(200*0.80)+((unit-200)*0.90))
STEP 5: [Else check unit is greater than 300 and calculate metercharge]
        If unit is greater than 300
       metercharge= metercharge+(200*0.80)+(300*0.90)+((unit-300)*1))
STEP 6: [To check and calculate if meter charge is greater than 400 ]
        If metercharge greater than or equal to 400
        metercharge=metercharge+(metercharge*0.15);
STEP 7: [Finished]
STOP
```

5.2 FLOW CHART


START
read name,unit
metercharge=100
unit<=200
True
False
unit>200 && unit<=300
metercharge= metercharge+(unit x 0.80)
True
False
unit>300
metercharge= metercharge+(200x 0.80)+((unit-200)x0.90))
True
metercharge= metercharge+(200x0.80)+(300x0.90)+((unit-300) x 1))
False
metercharge>=400
True
metercharge=metercharge+(metercharge x 0.15)
False
print name,unit,metercharges
STOP

5.3 PROGRAM

```
#include <stdio.h>
int main()
{
    char name[10];
    float unit, metercharge=100;
    printf("Enter your name and unit Consumed:");
    scanf("%s %f",&name,&unit);
    if(unit<=200)
```

```
          metercharge= metercharge+(unit*.80);
     else if(unit>200 && unit<=300)
          metercharge= metercharge+(200*0.80)+((unit-200)*0.90));
    else if(unit>300)
          metercharge= metercharge+(200*0.80)+(300*0.90)+((unit-300)*1));
    if(metercharge>=400)
          metercharge=metercharge+(metercharge*0.15);
     printf("Name: %s\n Number of unit consumed: %f \n MeterCharge: %f",name,unit,metercharge);
     return 0;
  }
```

****

Output 1:

```
Enter your name and unit Consumed:Suresh 200
Name: Suresh
Number of unit consumed: 200
MeterCharge : 260.000
```

Output 2:

```
Enter your name and unit Consumed:Ramesh 400
Name: Ramesh
Number of unit consumed: 400
MeterCharge : 724.5000
```

****

Viva Questions:

1) What is else-if ladder?
2) What are the possible outputs of if statement?
3) What is conditional branching statement?
4) Write the syntax of if- statement?
5) Write the syntax of else if statement?

Laboratory Program 6

Introduce 1D Array manipulation and implement Binary search

6.1 ALGORITHM

PURPOSE: Implement Binary search

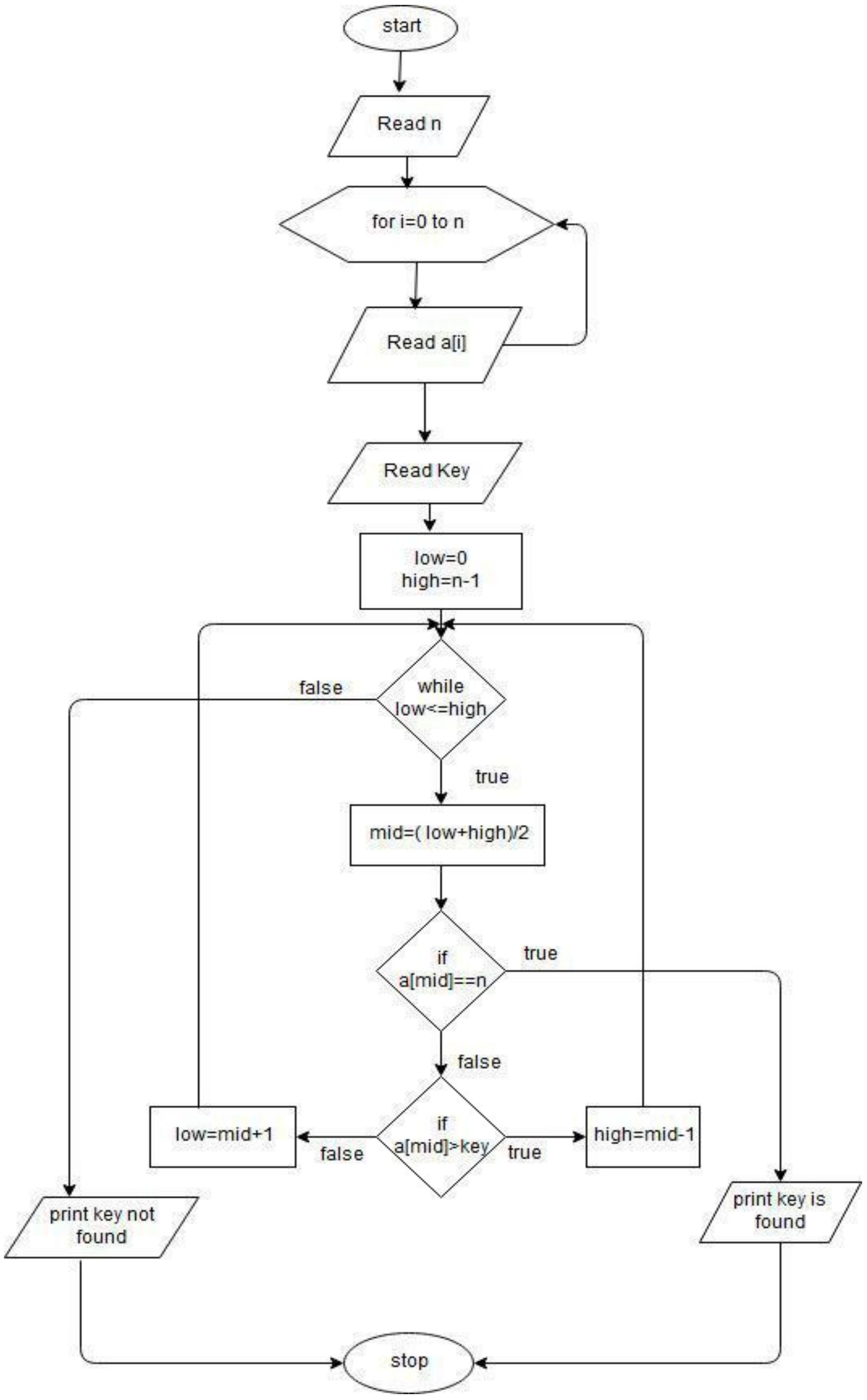
INPUT: n, a[50], key

OUTPUT: key is present at the position or key not found

```
START

STEP 1: [Read no of elements]

        Read n

STEP 2: [Read the elements of the array]

        For ← i=0 to n
        Read a[i]

        End ith for loop

STEP 3: [Read the elements to be searched]

        Read key

STEP 4: [Initiliazation]

        low = 0;

        high = n-1;

STEP 5: [ Check whearther low is less than of equal to high and calculate mid)

         while ← low <=high
         mid= (low+ high)/2;

STEP 6: [Check if mid is equal to key )

        If a[mid] is equal to key
```

```
STEP 7: [print the key position]
         Print key is present at the position mid+1
STEP 8: [if key found initialize found is 1]
        found=1
STEP 9: [check if mid is greater than key and calculate]
        If mid is greater than key
        high = mid-1;
STEP 10: [Else]
        low = mid+1;
STEP 11: [Check if key is not found]
          If not found
          Print  key not found
STEP 12:[Finished]
STOP
```

6.2 FLOWCHART



6.3 PROGRAM

```
#include <stdio.h>
int main()
{
     int a[50], key, i, n , low, high,mid, found =0;
     printf("\n Enter the number of elements in the array:
     "); scanf ("%d", &n);
     printf (" \n Enter the elements of the array: ");
     for(i=0;i<n;i++)
    {
          scanf("%d", &a[i]);
     }
    printf("\n Enter the key to be searched: \n" );
    scanf ("%d", &key);
    low = 0;
    high = n-1;
    while (low <=high)
   {
      mid= (low+ high)/2;
      if (a[mid] == key)
     {
         printf("\n %d is present at the position = %d", key, mid+1);
         found=1;
         break;
       }
      if (a[mid]>key)
                  high = mid-1;
       else
                  low = mid+1;
   }
if(!found)
            printf(“key not found”);
```

```
 return 0;
}
```

****

Output 1:

```
Enter the number of elements in the array: 5
Enter the elements of the array:
10 20 30 40 50
Enter the key to be searched: 30
30 is present at the position = 3
```

Output 2:

```
Enter the number of elements in the array: 5
Enter the elements of the array:
10 20 30 40 50
Enter the key to be searched: 60
key not found
```

****

Viva Questions:

What is searching?

What are the types of searching?

Explain with an example for binary search.

Explain with an example for linear search.

Which is better searching technique?

Laboratory Program 7

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

7.1 ALGORITHM

Purpose:To check whether a given number is prime or not.

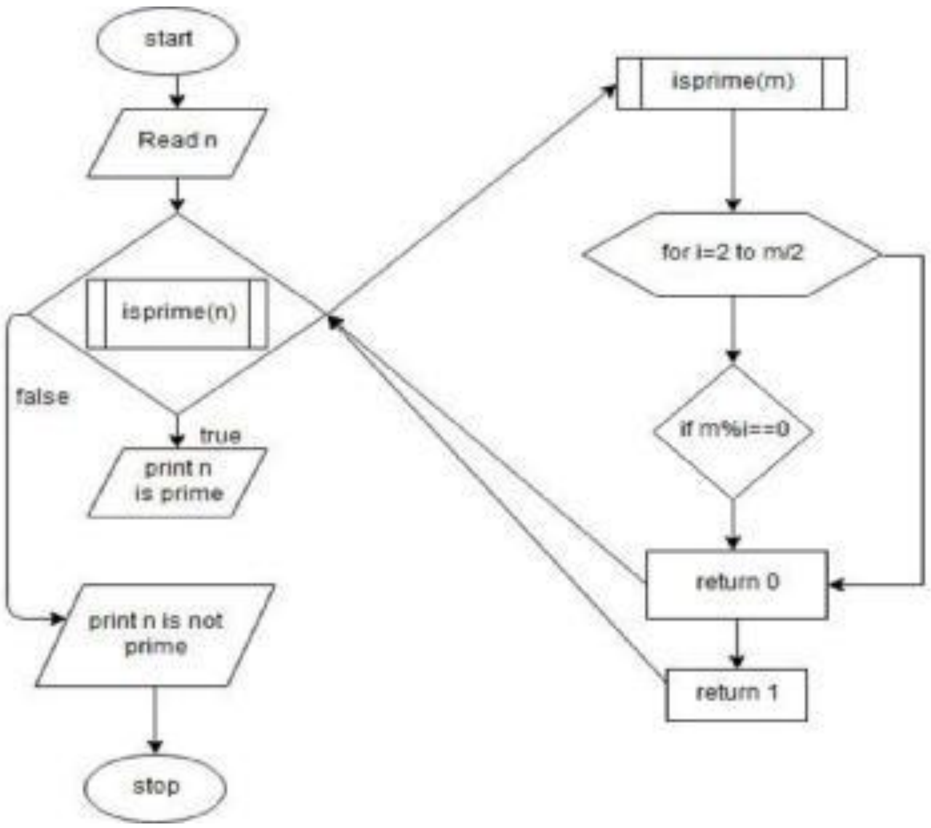
Input:A number

Output:Given Number is prime or not

```
START

STEP 1: [Enter a positive number]

read n

STEP 2: [To check number is prime or not]

        for ← i=2 to m/2
        if m% i is equal to 0
        return 0

STEP 3:[Display the output]
       Print the number is NOT prime.

STEP 4: [To check number is prime or not]

        for ← i=2 to m/2
        if m% i is equal to 1
         return 1

STEP 5:[Display the output]
       Print the number is prime.
```

STOP

7.2 FLOWCHART



7.3 PROGRAM

```
#include<stdio.h>
int isprime(int m)
{
   int i;

   for(i=2;i<=m/2;i++)
   {
       if(m%i==0)
           return 0;
   }
   return 1;
}

int main()
```

```
{
   int n;


   printf("Enter a +ve integer :");
   scanf("%d",&n);
   if (isprime(n))
       printf("%d is prime number\n",n);
   else
       printf("%d is not prime number\n",n);
   return 0;
}
```

****

OUTPUT1:

```
Enter a +ve integer : 5

5 is prime number
```

OUTPUT 2:

```
Enter a +ve integer : 12

12 is not prime number
```

****

VIVA QUESTION:

1. What is prime number?
2. What is a function?
3. What are the types of function?
4. Explain the flow of program with example
5. Explain the syntax of for loop.

PART B

Laboratory Program 8

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

8.1 ALGORITHM

PURPOSE : Multiply 2 matrices

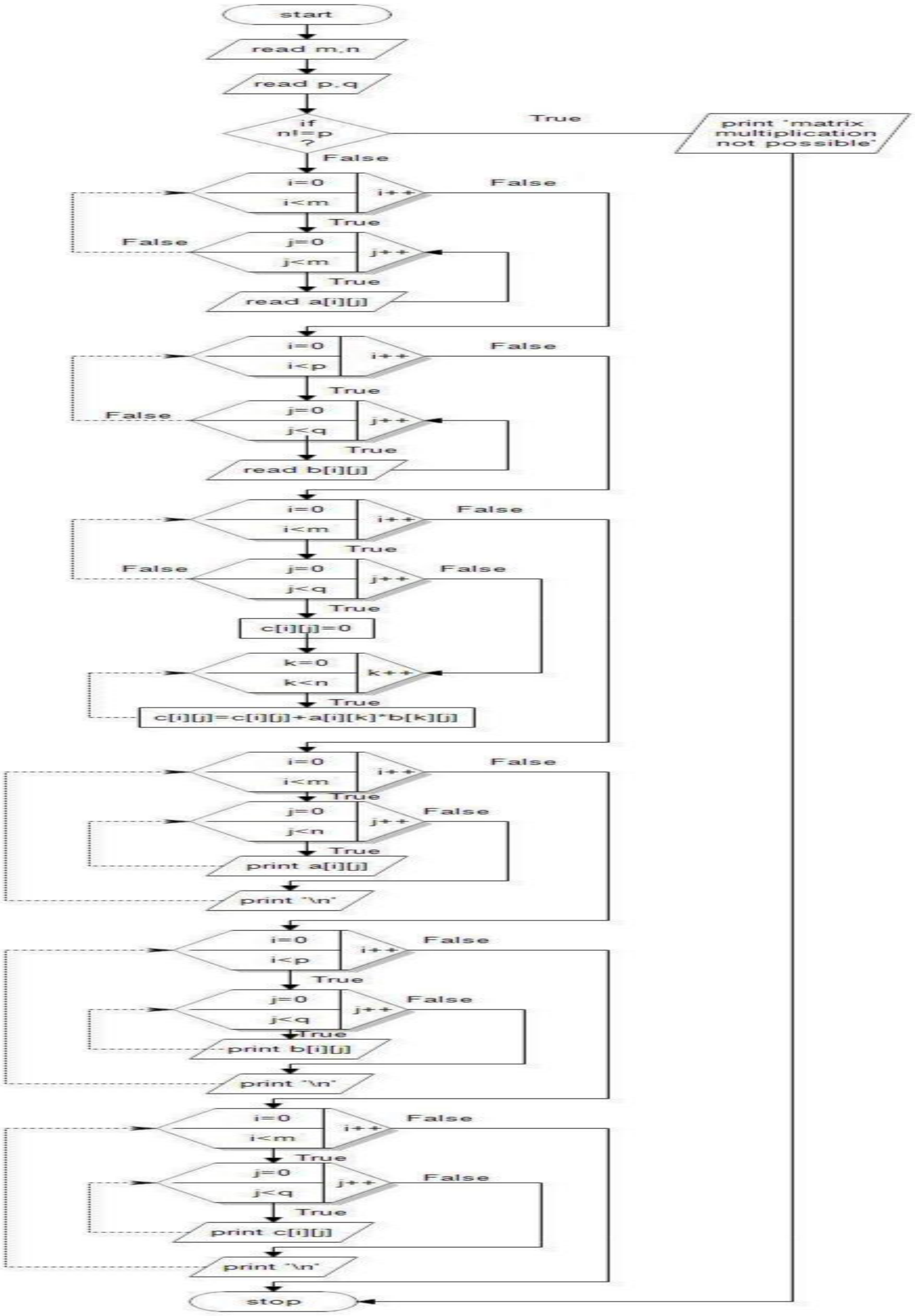
INPUT : Two matrices

OUTPUT : Multiplication of 2 matrices

```
START
STEP 1: [Read the order of matrix A]
        Read m,n
STEP 2: [Read the order of matrix B]
        Read p,q
STEP 3 : [Check for compatible]
         If n!=p
         Print matrix multiplication not possible
         Goto step 10
STEP 4 : [Read the elements of matrix A]
         For i ← 0 to m-1 do
         For j ← 0 to n-1 do
         Read a[i][j]
        End jth for loop
        End ith for loop
STEP 5: [Read the elements of matrix B]
        For i ← 0 to p-1 do
        For j ← 0 to q-1 do
        Read b[i][j]
        End jth for loop
        End ith for loop
STEP 6: [Compute multiplication]
```

```
    For i ← 0 to m-1 do
    For j ← 0 to q-1 do
    C[i][j] ← 0
    For k ← 0 to n-1 do
   C[i][j]c[i][j]+a[i][k]*b[k][j]
  End kth for loop
  End jth for loop
  End ith for loop
STEP 7: [Display matrix A]
    For i ← 0 to m-1 do
    For j ← 0 to n-1 do
    Print a[i][j]
    End jth for loop
    End ith for loop
STEP 8: [Display matrix B]
     For i ← 0 to p-1 do
     For j ← 0 to q-1 do
     Print b[i][j]
     End jth for loop
     End ith for loop
STEP 9: [Display matrix C]
    For i ← 0 to m-1 do
    For j ← 0 to n-1 do
    Print c[i][j]
   End jth for loop
   End ith for loop
STEP 10: [Finished]
STOP
```

8.2 FLOWCHART


start
read m,n
read p,q
if n!=p ?
True
print "matrix multiplication not possible"
False
i=0 i<m i++
j=0 j<m j++
read a[i][j]
i=0 i<p i++
j=0 j<q j++
read b[i][j]
i=0 i<m i++
j=0 j<q j++
c[i][j]=0
k=0 k<n k++
c[i][j]=c[i][j]+a[i][k]*b[k][j]
i=0 i<m i++
j=0 j<n j++
print a[i][j]
print "\n"
i=0 i<p i++
j=0 j<q j++
print b[i][j]
print "\n"
i=0 i<m i++
j=0 j<q j++
print c[i][j]
print "\n"
True
False
stop

8.3 PROGRAM

```
#include<stdio.h>
int main()
{
        int a[5][5],b[5][5],c[5][5],m,n,p,q,i,j,k;
        clrscr();
        printf("Enter the size of first matrix\n");
        scanf("%d %d",&m,&n);
        printf("Enter the size of second matrix \n");
        scanf("%d %d",&p,&q);
        if(n==p)
        {
                printf("Enter the elements of first matrix\n");
                for(i=0;i<m;i++)
                {
                  for(j=0;j<n;j++)
                  {
                        scanf("%d",&a[i][j]);
                   }
                 }
        printf("Enter the elements of the second matrix \n");
        for(i=0;i<p;i++)
        {
                for(j=0;j<q;j++)
                {
                scanf("%d",&b[i][j]);
                }
        }
        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("The product of two matrix is \n");
        for(i=0;i<m;i++)
        {
                for(j=0;j<q;j++)
                {
                        printf("%d\t",c[i][j]);
                }
        printf("\n");
        }
else
        printf("Multiplication is not possible");
}
```

****

Output1:

```
Enter the size of first matrix:2 2
Enter the size of second matrix:2 2
Enter the elements of first matrix :1 2 3 4
Enter the elements of second matrix :5 6 7 8
The product of two matrix
19 22
43 50
```

****

VIVA QUESTIONS:

1.What is a multi-dimensional array?

2. How to initialize two dimensional arrays?
3. How to pass a two dimensional array as function parameter?
4. How the memory is allocated for two dimensional array

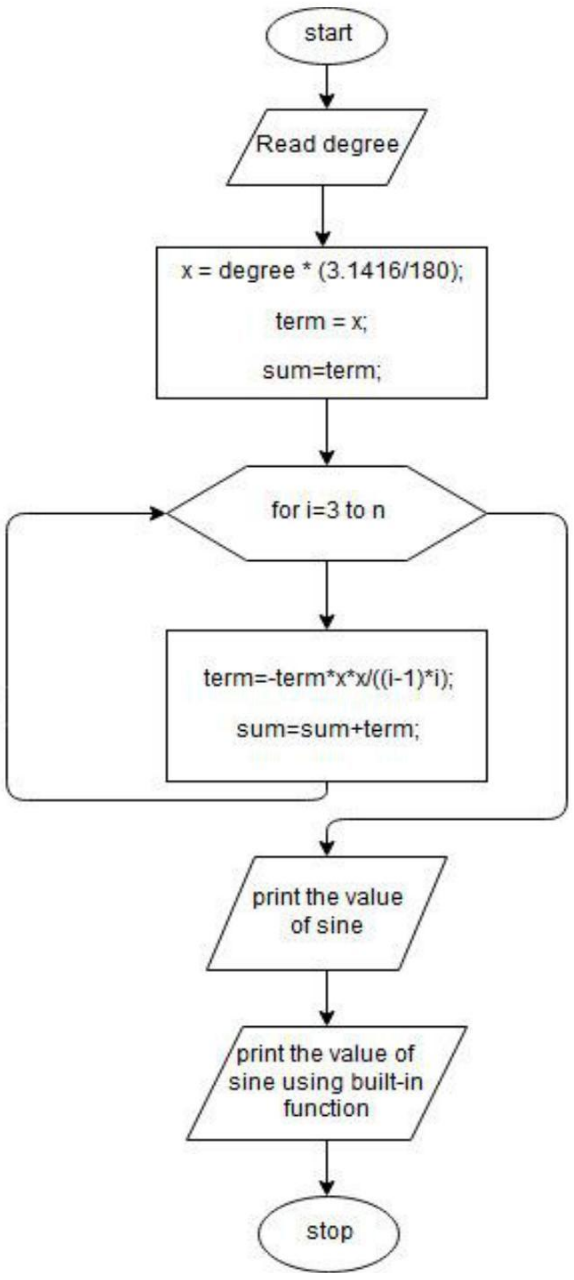
Laboratory Program 9

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.

9.1 ALGORITHM

```
Algorithm : To compute Sin(x) using Taylor series approximation
Purpose: To compute sine(x) value using Taylor's series
Input: Degree ie. Value of x
Output : sine value for given degree
START
Step 1:[Input value of degree]
        Read degree
Step 2: [Convert degree to radians]
        X=(degree *PI)/180
Step 3: [InitialCSE]
        term =x
        sum=term
Step 4: [Calcuate each term and add]
        term=-term*x*x/((i-1)*i);
        sum=sum+term;
Step 5: [Print the output]
       Print the sine value for the given degree
STOP
```

9.2 FLOWCHART


start
Read degree
x = degree * (3.1416/180);
term = x;
sum=term;
for i=3 to n
term=-term*x*x/((i-1)*i);
sum=sum+term;
print the value of sine
print the value of sine using built-in function
stop

9.3 PROGRAM

```
#include<stdio.h>
int main()
{
    int i, degree;
    float x, sum=0,term;
    printf("Enter the value of degree");
    scanf("%d",&degree);
    x = degree * (3.1416/180);
    term = x;
    sum=term;
    for (i=3;i<=n;i+=2)
    {
        term=-term*x*x/((i-1)*i);
        sum=sum+term;
    }
   printf("The sine of %d is %.3f\n", degree, sum);
   printf("The sine function of %d using library function is %.3f", degree, sin(x));
   return 0;
}
```

****

Output:

```
Enter the value of degree :90
The sine of 90 is 1.000
The sine function of 90 is 1.000
```

****

VIVA QUESTIONS:

1. What is pre-processor directive?
2. What is difference between const and #define.
3. What is use of fabs().
4. What is variable initialization and why is it important?
5. What is the difference between the = symbol and == symbol?
6. Can the curly brackets { } be used to enclose a single line of code?

Laboratory Program 10

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

10.1 ALGORITHM

Algorithm: string operations such as compare, concatenate, string length.

Purpose: To find String length, Compare two Strings, Concatenate two strings

Input: Two strings

Output: Length of string1

Concatenated String

Comparison of two strings

```
START
 STEP 1: [Store two strings in Str1 and Str2]
          Str1=String1 and Str2=String2
 STEP 2:[To find String Length]
            initialize   i to 0
           while(str[i] not equal to 0)
           increment i by 1
STEP 3: [To Compare two Strings]
          initialize i to 0
       while (str1[i] is equal to str2[i])
         if (str1[i] is equal to null)
          break;
        increment i by 1
         return str1[i]-str2[i];
STEP 4: [To Concatenate two Strings]
```

```
    initialize i to 0 and j to 0
    while (str1[i] not equal to null)
    increment i by 1
   while (str2[j] not equal to null)
      str1[i++]=str2[j++];
```

STEP 5: [Output the results]

The length of the String

res1=Comparision of two Strings

res2= Concatenated String

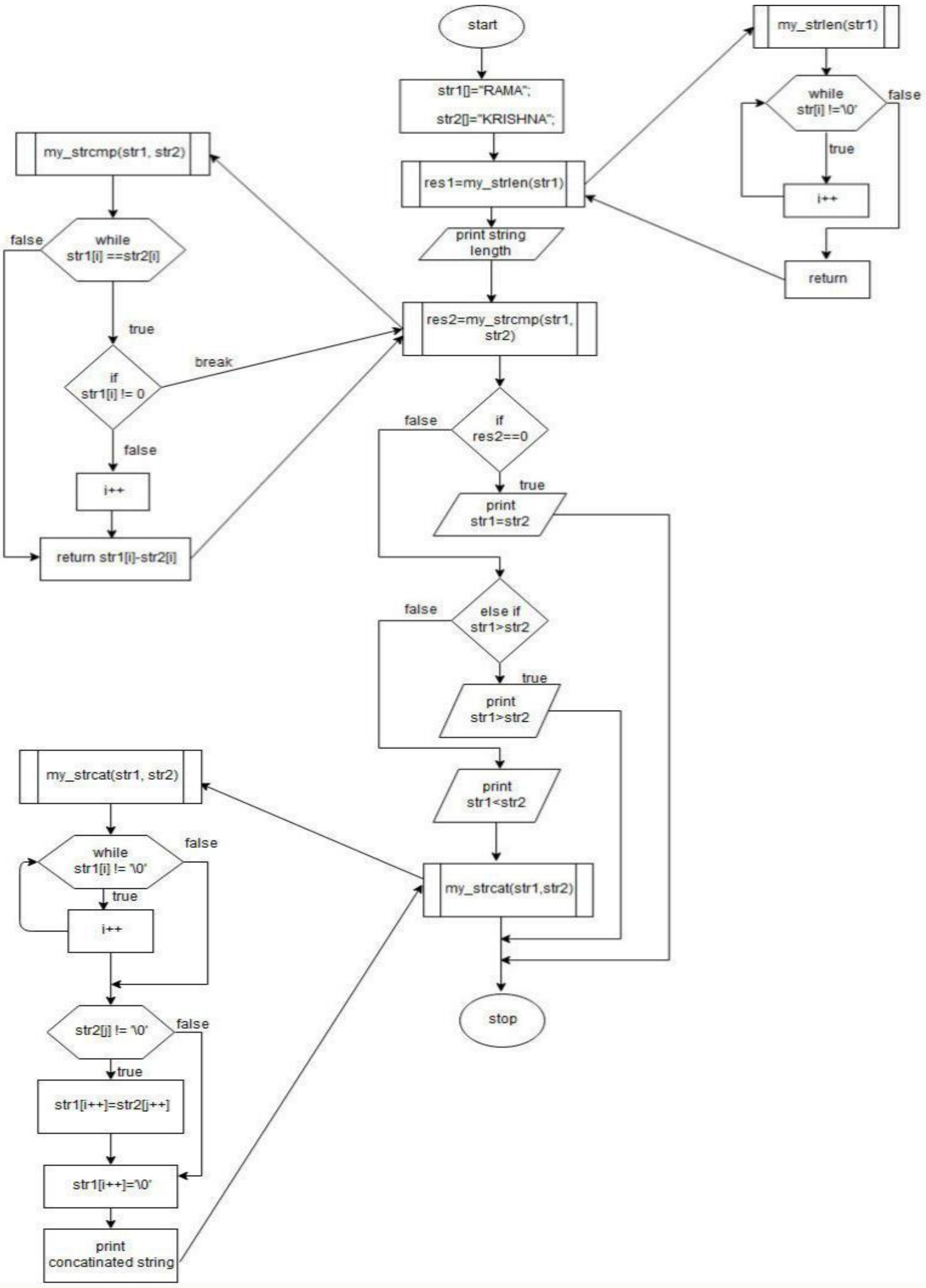
STOP

10.3 PROGRAM

```
#include<stdio.h>
int my_strlen(char str[])
{
   int i=0;

   while (str[i]!='\0') i++;

   return i;
}
int my_strcmp(char str1[],char str2[])
{
   int i;
     i=0;
   while (str1[i]==str2[i])
   {
      if (str1[i]=='\0') break;
      i++;
   }
```

```
    return str1[i]-str2[i];
}

void my_strcat(char str1[], char str2[])
{
    int i,j;
    i=0;
    while (str1[i]!='\0') i++;
    j=0;
    while (str2[j]!='\0')
    {
        str1[i++]=str2[j++];
    }
    str1[i++]='\0';
 printf("concatinated string= %s", str1);
}
void main()
{
    char str1[]="RAMA";
    char str2[]="KRISHNA";
    int res1,res2;
    res1=my_strlen(str1);
    printf("the length of the string1=%d\n", str1);
    res2=my_strcmp(str1,str2);
    if (res2==0)
        printf("%s is equal to %s\n",str1,str2);
    else if (res2>0)
        printf("%s is greater than %s\n",str1,str2;
    else
        printf("%s s is less than %s\n",str1,str2)
     my_strcat(str1,str2);
}
```

10.2 FLOWCHART



OUTPUT

```
The length of the string1=4

RAMA is greater than KRISHNA

concatinated string=RAMAKRISHNA
```

****

VIVA QUESTIONS

1) What is string?
2) What are the built-in functions of string?
3) Difference between user defined functions and built in functions.
4) What is null character?
5) Explain the flow of program with example.

Laboratory Program 11

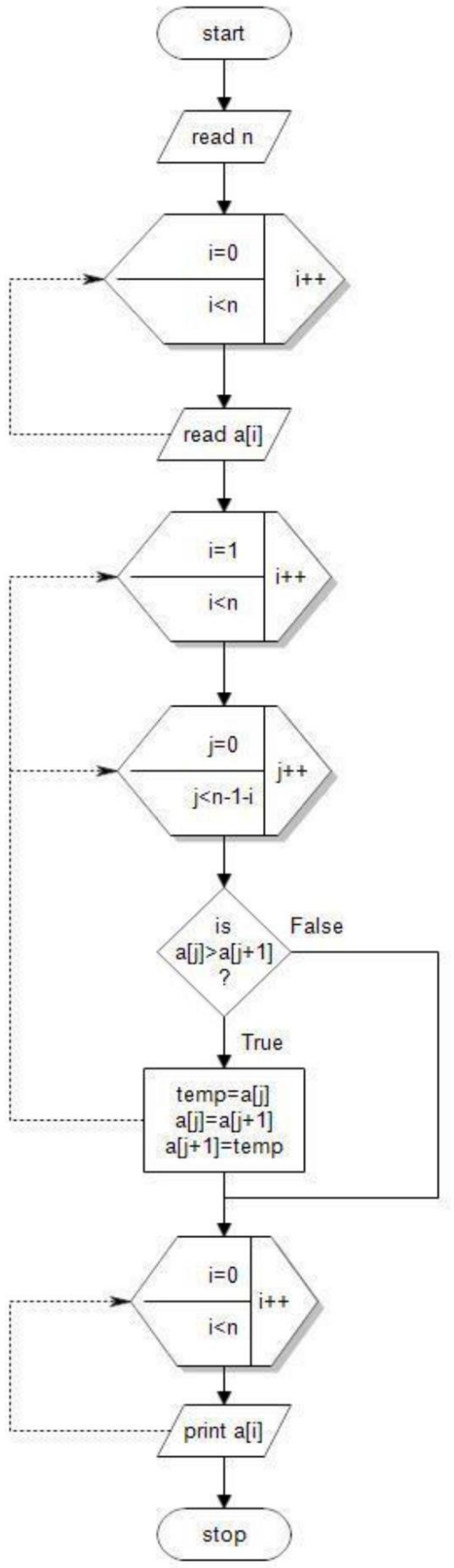
Develop a program to sort the given set of N numbers using Bubble sort.

11.1 ALGORITHM

```
ALGORITHM : Bubble sort
PURPOSE   : Arranging the numbers in ascending order using bubble sort technique
INPUT     : N, interger numbers in arrary a[i]
OUTPUT    : Numbers are arranged in ascending order
START
STEP 1: [Input number of elements]
        Read n
STEP 2: [Input the elements/numbers into array]
        For i ← 0 to n
        Read a[i]
        Endfor
STEP 3 : [Sorting the elements in ascending order]
        For i ← o to n-1
        For j ← o to n-1
        [Compare the adjacent elements]
        If(a[j]>a[j+1]) then
        [Swap these elements]
        Temp ← a[j]
        a[j] ← a[j+1]
        a[j+1] ← temp
       endif
       endfor
       endfor
STEP 4: [Display the sorted elements of array]
       For i ← 0 to n
       Print a[i]
       Endfor
STEP 5: [Finished]
```

STOP

11.2 FLOWCHART



11.3PROGRAM

```
#include<stdio.h>
#include<conio.h>
int main()
{
        int n,i,j,a[10],temp;
        clrscr();
        printf("Enter the no. of elements :\n");
        scanf("%d",&n);
        printf("Enter the array elements \n”);
        scanf("%d",&a[i]);
        printf("The original elements are \n");
        for(i = 0 ; i < n ; i++)
                printf("%d ",a[i]);
        for(i= 0 ; i < n-1 ; i++) // Number of Passes
        {
              for(j= 1 ; j< n-j-1; j++) // Comparisons
        {
        if(a[j] > a[j+1])
        {
                temp = a[j];
                a[j] = a[j+1];
                a[j+1] = temp;
        }
        }
        }
        printf("\n The Sorted elements are \n");
        for(i = 0 ; i < n ; i++)
                printf("%d\t ",a[i]);
        return 0;
}
```

****

Output 1:

```
Enter the no. of elements:5
5 4 3 2 1
The Sorted elements are 1 2 3 4 5
```

****

VIVA QUESTIONS:

1. Why the name bubble sort?

2. Mention the different types of sorting techniques?

3. Explain the logic of bubble sort with an example.

4. What is nested for loop?

Laboratory 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).

12.1 ALGORITHM

Algorithm: Square root

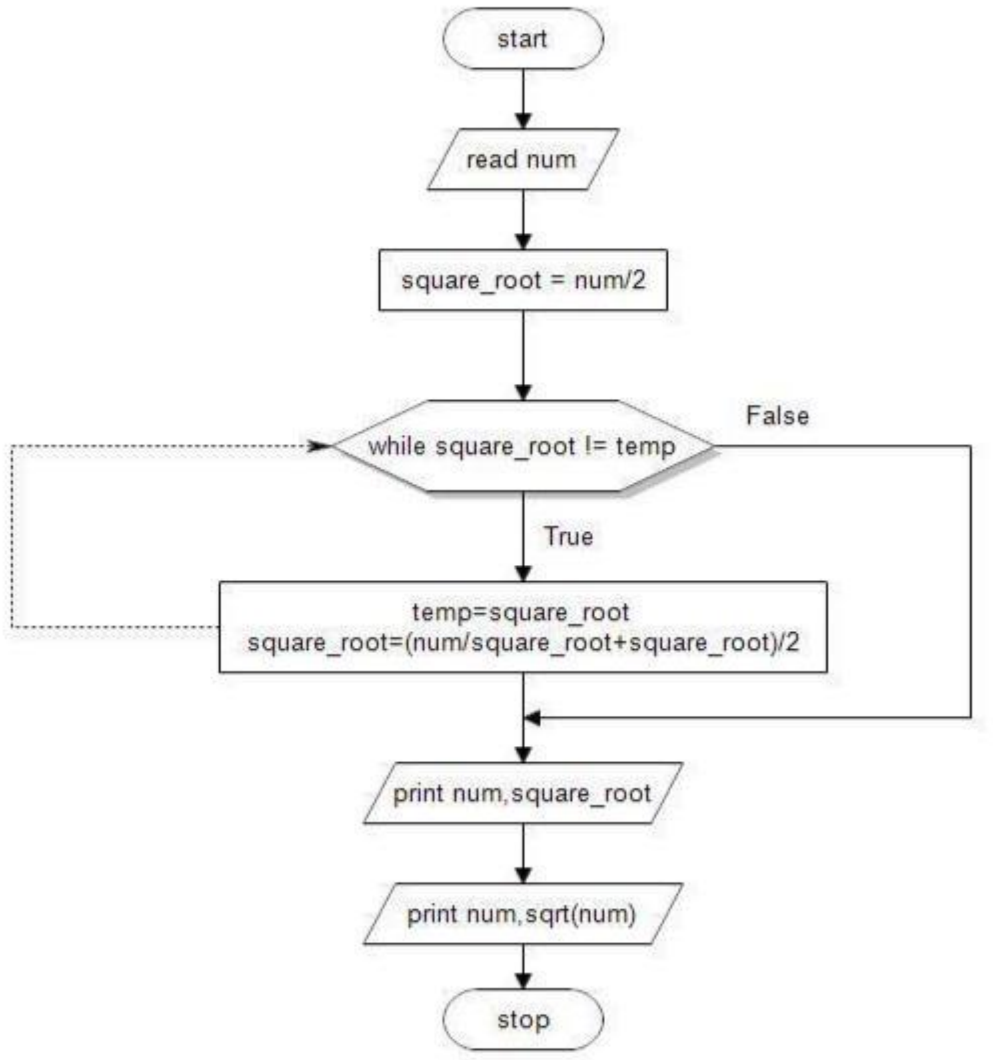
Purpose : To find square root of a given number without using inbuilt functions

Input: num

Output : Square root of a given number

```
START
STEP 1: [Initialize]
        temp ← 0
STEP 2: [Enter a number]
        Read num
STEP 3: [Divide the num by 2]
        square_root=num/2
STEP 4: [Calculate a square root of a number]
        While (square_root not equal to temp)
        temp ← square_root
        square_root ← (num/square_root+square_root)/2
STEP 5: [Display results]
        print "square root"
STEP 6: [Display result using the library function]
        print" square root of a number is", sqrt (num)
STEP 7: [Finished]
STOP
```

12.2 FLOWCHART



Program

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
void main()
{
        float num,square_root,temp=0;
        clrscr();
        printf("Enter the number to find square root\n");
        scanf("%f",&num);
        square_root=num/2;
        while(square_root!=temp)
        {
                temp=square_root;
                square_root=(num/square_root+square_root)/2;
        }
        printf("Square Root of %f is %f\n",num,square_root);
        printf("Square Root of %f usinf inbuilt Function is
        %f\n",num,sqrt(num)); getch();
}
```

****

Output 1:

```
Enter a number
49
Square root using program is 7
Square Root using Library Function is
7
```

Output2:

```
enter a number -12
Can't Find for Negative Numbers
```

****

VIVA QUESTIONS:

1. What is typecasting? Explain with examples.
2. Difference between float and double data types.
3. Explain for loop?
4. What is a use of break statement?
5. Difference between continue and break statement?

Laboratory Program 13

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

13.1 ALGORITHM

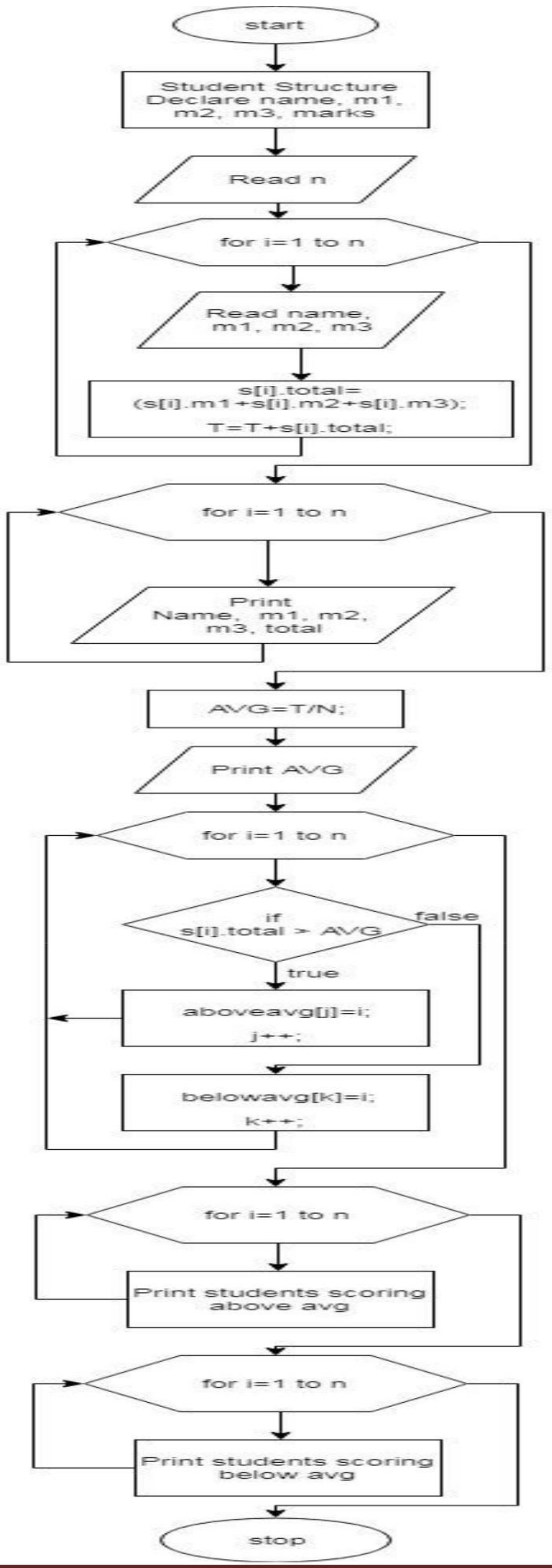
PURPOSE: To Implement structures to read, write, compute average- marks and the students scoring above and below the average marks for a class of N students.

Input: number of Students, name ,marks1,marks2,marks3

Output: Average marks and printing the students scoring above and below the average marks

```
START
STEP 1: [input number of students]
        Read n
STEP 2: [input details of students ie.name amd marks]
         Read name,m1,m2,m3
STEP 3: [ Calculate total and average]
              For ← i=0 to n
             s[i].total=(s[i].m1+s[i].m2+s[i].m3);
             T=T+s[i].total;
            AVG=T/N;
STEP 4: [Find students above and below average]
          for ← i=0 to n
           aboveavg[j]=i;
                 j++;
         else
                 belowavg[k]=i;
                 k++;
STEP 5:[Finished]
STOP
```

13.2 FLOWCHART



13.3PROGRAM

```
#include<stdio.h>
struct student
{
    char name[20];
    float m1,m2,m3,total;
};

main()
{
    int i,j,k,m,n,aboveavg[10],belowavg[10];
    float N,T=0,AVG=0;
    struct student s[10];
    system("clear");
    printf("Enter number of students.\n");
    scanf("%d",&n);
    N=(float)n;
    printf("Enter the details of students.\n");
    for (i=0;i<n;i++)
    {
        printf("Enter the details of student %d \n",i+1);
        printf("Enter name \n");
        scanf("%s",s[i].name);
        printf("Enter m1,m2,m3 \n");
        scanf("%f%f%f",&s[i].m1,&s[i].m2,&s[i].m3);
        s[i].total=(s[i].m1+s[i].m2+s[i].m3);
        T=T+s[i].total;
    }
    printf("Details of Students.\n");
    printf("Name\t m1\t m2\t m3\t total\n"); for
    (i=0;i<n;i++)
        printf("%s\t%f\t%f\t%f\t%f\t\n",s[i].name,s[i].m1,s[i].m2,s[i].m3,s[i].total);
    AVG=T/N;
    printf("AVG = %f \n",AVG);
    j=0;
    k=0;
    for (i=0;i<n;i++)
    {
        if (s[i].total > AVG)
        {
            aboveavg[j]=i;
            j++;
        }
        else
        {
            belowavg[k]=i;
            k++;
        }
    }
    printf("Students scoring above avg. \n");
    for (i=0;i<j;i++)
    {
        printf("%s %f\n",s[aboveavg[i]].name,s[aboveavg[i]].total);
    }
    printf("Students scoring below avg. \n");
    for (i=0;i<k;i++)
    {
        printf("%s %f\n",s[belowavg[i]].name,s[belowavg[i]].total);
    }

}
```
****

OUTPUT

```
Enter number of students. 3

Enter the details of students.

Enter the details of student 1

Enter name Anil

Enter m1,m2,m3

67 84 72

Enter the details of student 2

Enter name Sudhir

Enter m1,m2,m3

76 55 68

Enter the details of student 3

Enter name Suresh

Enter m1,m2,m3

58 79 92

Details of Students.

Name    m1          m2            m3      total

Anil    67.000000    84.000000    72.000000    223.000000

Sudhir 76.000000     55.000000    68.000000    199.000000

Suresh 58.000000     79.000000    92.000000    229.000000

AVG = 217.000000
```

```
Students scoring above avg.

Anil 223.000000

Suresh 229.000000

Students scoring below avg.

Sudhir 199.000000
```

****

Laboratory 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.

14.1 ALGORITHM

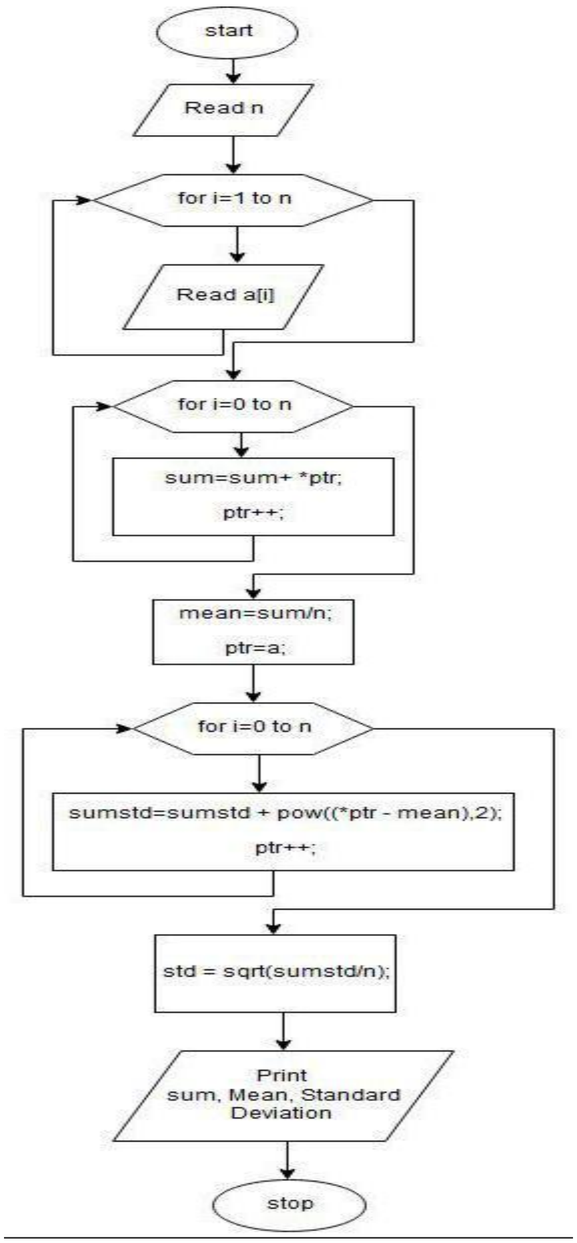
Algorithm

Purpose:To find sum,mean and standard deviation of all elements in an array Input: An array of n elements

Output:Print the sum,mean and standard deviation of all elements in an array START

```
STEP 1:[ Input the no of elements]
            Read n
STEP 2:[input array elements]
           For ← i= 0 to n
          Read array
STEP 3: [Calculate sum]
           For ← i= 0 to n
            sum=sum+ *ptr
            increment ptr by 1
STEP 4: [Calculate mean]
             mean=sum/n
STEP 5: [Calculate Standard deviation]
               sumstd=sumstd + pow((*ptr - mean),2);
                increment ptr by 1
               std = sqrt(sumstd/n)
STEP 6:[Display output]
Print the sum,mean and standard deviation of all elements in an array

STOP
```

14.2 FLOW CHART



14.3 PROGRAM

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
main()
{
   float a[10], *ptr, mean, std, sum=0, sumstd=0;
```

```
        int n,i;
        clrscr();
         printf("Enter the no of elements\n");
         scanf("%d",&n);
         printf("Enter the array elements \n");
         for(i=0;i<n;i++)
        {
                scanf("%f",&a[i]);
        }
        ptr=a;
        for(i=0;i<n;i++)
        {
                sum=sum+ *ptr; ptr++;
        }
        mean=sum/n;
        ptr=a;
        for(i=0;i<n;i++)
       {
                sumstd=sumstd + pow((*ptr - mean),2);
                ptr++;
        }
        std = sqrt(sumstd/n);
        printf("Sum=%.3f\t",sum);
        printf("Mean=%.3f\t",mean);
        printf("Standard deviation=%.3f\t", std);
    }
```

****

Output 1:

```
Enter the number of elements 5
Enter the array elements are 1 5 9 6 7
Sum=28
Mean=5.6
Standard deviation=2.09
```

****

Viva Questions:

1. Define pointer?

2. How do you declare a pointer variable?

3. What is * and & in pointer concept.

4. What are the advantages and disadvantages of using pointer?

5. Give the difference between static allocation and dynamic allocation of memory space.

6. What is the effect of the ++ and -- operators on pointer variable

7. Explain the pointers to arrays concept?

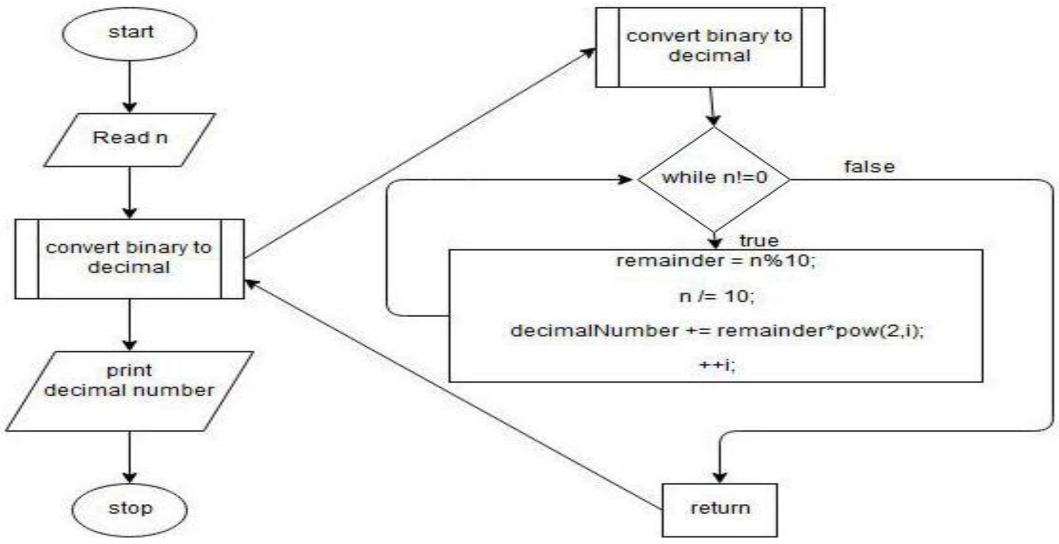
Laboratory Program 15

Implement Recursive functions for Binary to Decimal Conversion.

15.1 ALGORITHM

```
Algorithm

Purpose: To convert binary number to decimal number

Input: Binary number

Output: Decimal number

START

STEP 1: [input binary number]
            read binary number, n
STEP 2:[till n not equal to zero]
           remainder = n%10
           n=n/10
           decimalnumber=decimalnumber+ remainder*pow(2,i)
STEP 3:[Output decimal number]
           Print decimal number
STOP
```

15.2 FLOWCHART



15.3 PROGRAM

```
#include <stdio.h>
#include <math.h>
int convertBinaryToDecimal(long n);

int main()
{
   long long n;
   printf("Enter a binary number: ");
   scanf("%ld", &n);
   printf("%ld in binary = %d in decimal", n,
   convertBinaryToDecimal(n)); return 0;
}

int convertBinaryToDecimal(long n)
{
   int decimalNumber = 0, i = 0, remainder;
   while (n!=0)
   {
      remainder = n%10;
      n /= 10;
      decimalNumber += remainder*pow(2,i);
      ++i;
```

```
    }
    return decimalNumber;
}
```

****

OUTPUT 1:

```
Enter a binary number:1010
1010 in binary = 10 in decimal
```

OUTPUT 2:

```
Enter a binary number:11111111
11111111 in binary = 255 in decimal
```

****

Viva Question:

1) What is recursion?
2) What are binary numbers?
3) What is recursive function?
4) Explain the flow of program with example
5) Explain mathematical functions.