OBJ OBJ OBJ

CMR INSTITUTE OF TECHNOLOGY

Department of Computer Science and Engineering

LAB MANUAL

COMPUTER PROGRAMMING LABORATORY (BPOPS103/203)

Semester II

Academic Year: 2024-25

COMPUTER PROGRAMMING LABORATORY

[As per Choice Based Credit System (CBCS) scheme]
(Effective from the academic year 2022 -2023)

SEMESTER – I/II

Course Code	BPOPS103/203	**CIE Marks**	50
Teaching Hours/Week	(L: T:P: S) 0:0:2:0	**SEE Marks**	50
Total Number of Lecture Hours	40	**Exam Hours**	03

Course Objectives

1. Explain problem statements and identify appropriate solutions
2. Demonstrate the use of IDE, C Compiler, and identify and rectify the syntax and syntactic errors during programming.
3. Development of algorithms and programs using constructs of C programming language
4. Reporting the observations

Prerequisites

- ➢ Basic knowledge of Programming
- ➢ Analytical and Logical thinking to analyze problem and deriving solution to problem

Course CO-PO-PSO Mapping

Course Outcomes		Programs covered	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3	PSO4
CO1	Develop algorithm, flowchart for implementation of solution of the given problem in C Language	1-11	3	2	2	2	2	-	-	-	2	-	-	2	2	-	2	-
CO2	Demonstrate coding, debugging and execution of control and	1-11	3	2	3	2	2	-	-	-	3	-	-	2	2	-	2	-

	compound statements in C Language																	
CO3	Implement searching and sorting solutions in C Language	**1-11**	2	-	2	2	2	-	-	-	2	-	-	2	2	-	2	-
CO4	Apply use of functions, recursive functions, arrays, strings, structures and pointers in C Language	**1-11**	2	2	2	2	2	-	-	-	2	-	-	3	2	-	2	-
CO5	Report with documents of observation	**1-11**	3	2	2	2	2	-	-	-	3	-	2	-	-	-	3	-

PROGRAM OUTCOMES (PO), PROGRAM SPECIFIC OUTCOMES (PSO)				**CORRELATION LEVELS**	
PO1	Engineering knowledge	**PO7**	Environment and sustainability	**0**	No Correlation
PO2	Problem analysis	**PO8**	Ethics	**1**	Slight/Low
PO3	Design/development of solutions	**PO9**	Individual and teamwork	**2**	Moderate/ Medium
PO4	Conduct investigations of complex problems	**PO10**	Communication	**3**	Substantial/ High
PO5	Modern tool usage	**PO11**	Project management and finance		
PO6	The Engineer and society	**PO12**	Life-long learning		
PSO1	Design and develop applications using different stacks of web and programming technologies.				
PSO2	Design and develop secure, parallel, distributed, networked, and digital systems.				
PSO3	Apply software engineering methods to design, develop, test and manage software systems.				
PSO4	Design and develop intelligent applications for business and industry.				

Syllabus

1. Simulation of a Simple Calculator.
2. Compute the roots of a quadratic equation by accepting the coefficients. Print appropriate messages.
3. 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.
4. Write a C Program to display the following by reading the number of rows as input,

   1
   1 2 1
   1 2 3 2 1
   1 2 3 4 3 2 1
   ---
   nth row
5. Implement Binary Search on Integers.
6. Implement Matrix multiplication and validate the rules of multiplication.
7. Compute sin(x)/cos(x) using Taylor series approximation. Compare your result with the built-in library function. Print both the results with appropriate inferences.
8. Sort the given set of N numbers using Bubble sort.
9. Write functions to implement string operations such as compare, concatenate, and find string length. Use the parameter passing techniques.
10. Implement structures to read, write and compute average- marks of the students, list the students scoring above and below the average marks for a class of N students.
11. Develop a program using pointers to compute the sum, mean and standard deviation of all elements stored in an array of N real numbers.
12. Write a C program to copy a text file to another, read both the input file name and target file name.

Content Beyond Syllabus

Prog #	Title
1	Write a C program to print prime numbers within a range using functions.
2	Write a C program to find GCD and LCM using recursion.

Introduction

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

C is a high level language, and it needs a compiler to convert it into an executable code. Then, we can execute the program by running the executable code or file.

A compiler is a software that converts a program written in High Level Language to Machine Language. Few examples of compiler are cc, gcc, cpp, java etc.

gcc and cc converts C program to machine language.
cpp converts C++ program to machine language.
java converts java programs to machine language.

I Execution of C program:

In order to execute or run a C program the following steps needs to be followed -

1. Edit the program using the editor - gedit
2. Save the program with a proper name.
3. Compile the program.
4. Run the program.

1. To edit a C program

To edit a C program, we need a text editor. We will be using the editor - **gedit** to edit C programs.

There are two ways to invoke gedit

1. From the Terminal Window
2. From gedit icon

1. **From Terminal Window:**
   To edit C program file from terminal window, follow the following steps:
   1. Open the terminal window either by typing **<ctrl>+<Alt>+t** or by double clicking on the Terminal icon in the Taskbar.
   2. .In the Terminal window, at the command prompt - $ (dollar), type 'gedit <filename>.c' as shown below
      $ gedit hello.c

2. From gedit icon:

Double click the gedit icon in the taskbar.

If gedit icon is not present in the taskbar, click on **'Search your Computer**' icon (The very first icon at the top left corner of the monitor) and type 'e' in the search box. Text editor icon is displayed. Drag the icon and place it in the taskbar and double click the gedit icon.

Preferably, invoke gedit from the gedit icon.

The editor opens with an **untitled document.** Now, type the code for the program. Let us type the very first program of C - The Hello World program. Type the following program in the editor.

```
/* Program to print Hello World*/
#include <stdio.h>.
int main()
{
        printf("Hello World");
        Return 0;
}.
```

2. To Save the Program with Proper name and extension:

After typing the complete program, it must be saved. To save the program press the shortcut key **ctrl+s** or click on the **save** button at the top right corner of the gedit window. If it is not named, it has to be given a proper name. Name the file appropriate to the program and the extension must be '.c' (dot followed by lower case c).

Example: hello.c palin.c, sqroot.c, binsrch.c, matmul.c etc.

Note: The C compiler is case sensitive, it treats the file as C program file, only if the extension name is ".c" and not ".C".

3. To Compile the Program:

To compile the program, open terminal window and at the prompt, type the following command and press enter key.

$ cc hello.c

An output file (executable file) with name **a.out** is created if there are no errors in the program. In case there are errors/warnings, they are displayed on the terminal window.

To run or Execute the program:
To run the program, type **./a.out** at the prompt as shown below

$./a.out

The program gets executed and the output is displayed on the terminal window.

Basic Linux Commands

1. **ls**

List files and/or directories. If no argument is given, the contents of current directory are shown.
$ ls
example file1.txt file2.txt file3.txt
If a directory is given as an argument, files and directories in that directory are shown.
$ ls /usr
bin games include lib lib64 local sbin share src
'ls -l' displays a long listing of the files.
$ ls -l
total 4

drwxr-xr-x 2 raghu raghu 4096 2012-07-06 12:52 example
-rw-r--r-- 1 raghu raghu 0 2012-07-06 12:52 file1.txt

-rw-r--r-- 1 raghu raghu 0 2012-07-06 12:52 file2.txt
-rw-r--r-- 1 raghu raghu 0 2012-07-06 12:52 file3.txt

2. **cd**

The cd command - change directory - will allow the user to change between file directories. As the name command name suggest, you would use the cd command to circulate between two different directories.

Ex:
$ pwd
/home/raghu
$ cd /usr/share/
$ pwd
/usr/share
$ cd doc
$ pwd
/usr/share/doc
$cd .. To exit from current directory
$pwd
/usr/share

3. **mv** – move command

$ mv source destination

Move files or directories. The 'mv' command works like 'cp' command, except that the original file is removed. But, the mv command can be used to rename the files (or directories).

4. **pwd** - present working directory

'pwd' command prints the absolute path to current working directory.

$ pwd
/home/raghu

5. **mkdir**

To create a directory, the 'mkdir' command is used.
$ mkdir example
$ ls -l
total 4
drwxr-xr-x 2 raghu raghu 4096 2012-07-06 14:09 example

6. **rmdir**

rm files|directories
$ rm file2

'rmdir' command removes any empty directories, but cannot delete a directory if a file is present in it. To use ‘rmdir’ command, you must first remove all the files present in the directory you wish to remove (and possibly directories if any).

7. **cat**

The 'cat' command is actually a concatenator but can be used to create and view the contents of a file..

$cat filename

Displays the contents of the file

$ cat>filename

Press enter, then you can write something in the file and then to save the file contents press ctrl+d then enter.

8. **touch**

For creating an empty file, use the touch command.
$ touch file1 file2 file3

9. **cp** - copy command

$cp source destination
Copy files and directories. If the source is a file, and the destination (file) name does not exit, then source is copied with new name i.e. with the name provided as the destination.

10. **rm**

Removes files from the directory:
rm testfile.txt

11. **cal**

Displays the calendar of the current month.

$ cal July 2012 Su Mo Tu We Th Fr Sa 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21	$ cal 08 1991 August 1991 Su Mo Tu We Th Fr Sa 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17

22 23 24 25 26 27 28 29 30 31	18 19 20 21 22 23 24 25 26 27 28 29 30 31

12. clear

The clear command does exactly what it says. The clear command clears the screen and wipes the board clean. Using the clear command will take the user back to the start prompt of whatever directory you are currently operating in. To use the clear command simply type clear.

QUESTIONS FOR VIVA:

1. What is a compiler? What is the function of a compiler?
2. Explain intermediate code.
3. Which are the four steps followed to convert a C program to executable code?
4. What is the difference between assembly and machine language?
5. What are the steps followed in pre-processing phase?
6. What happens during the linking phase?
7. Why does the size of the file increase after linking phase?
8. Which are the different LINUX commands and what is their purpose.?

Prog 1: Simulation of a Simple Calculator.

AIM: Simulation of a Simple Calculator.

ALGORITHM:

Algorithm:

Input: Two integers(operands) and operator
Output: Result of the operation

Step 1: Start
Step 2: Read two operands and an arithmetic operator

Step 3: Check if operator equals '+', if yes, then goto step 4 else goto step 5
Step 4: Compute addition operation - res = num1 + num2 and goto step 18

Step 5: Check if operator equals '-', if yes, then goto step 6 else goto step 7
Step 6: Compute subtraction operation - res = num1 – num2 and goto step 18

Step 7: Check if operator equals '*', if yes, then goto step 8 else goto step 9
Step 8: Compute multiplication operation - res = num1 * num2 and goto step 18

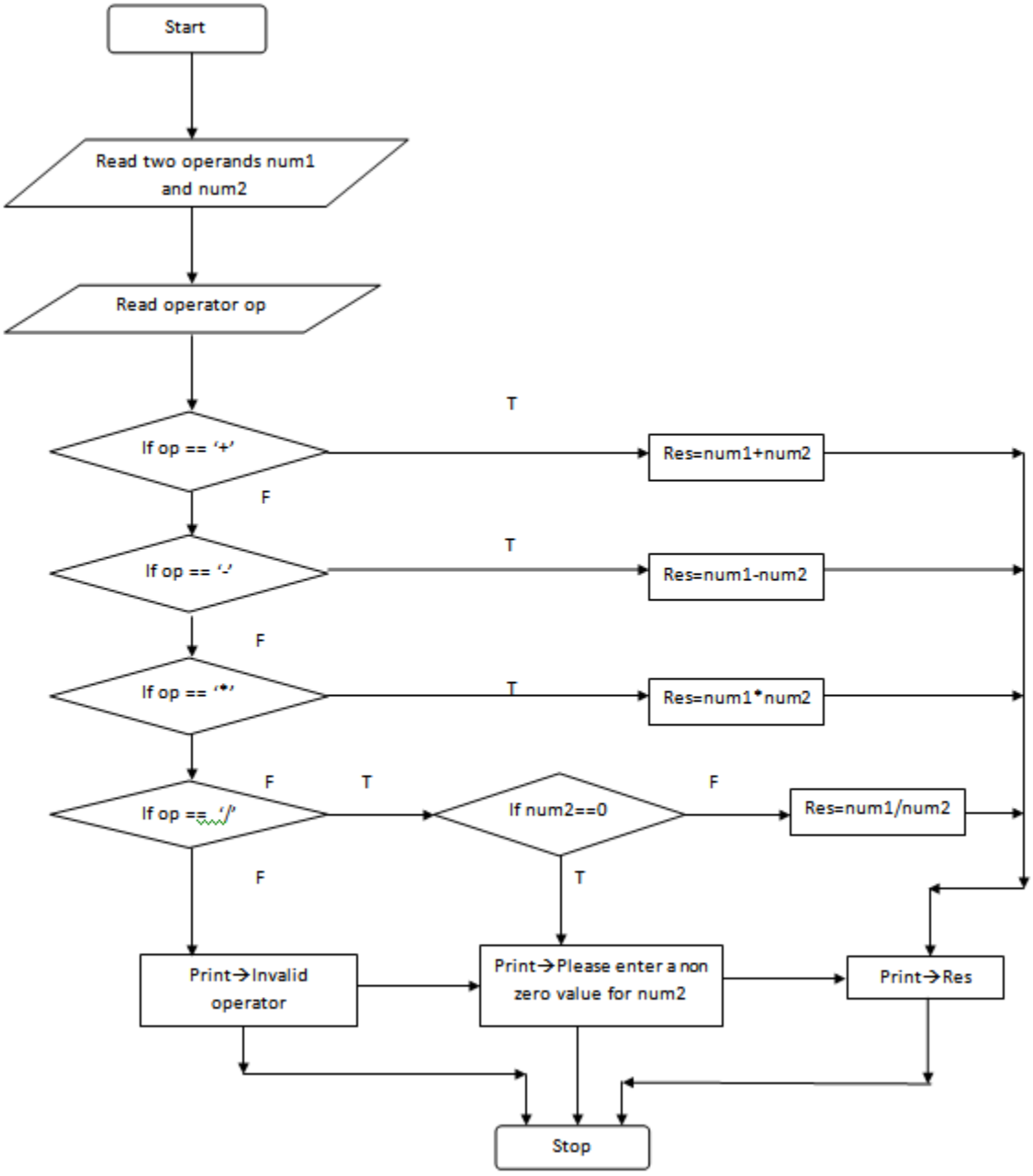
Step 9: Check if operator equals '/', if yes, goto step 10 else goto step 13
Step 10: Check if num2 equals Zero, if yes, then goto step 11 else goto step 12
Step 11:Display – "Divide by zero error." and goto step 19
Step 12: Compute division operation - res = num1 / num2 and goto step 18

Step 13: Check if the operator equals '%', if yes, then goto step 14 else goto step 17
Step 14: Check if num2 equals zero, if yes, then goto step 15 else goto step 16
Step 15: Display - "Divide by zero error." and goto step 19.
Step 16: Compute modulus operation – res = num1 % num2 and goto step 18

Step 17: Display "Invalid operator" and goto step 19.

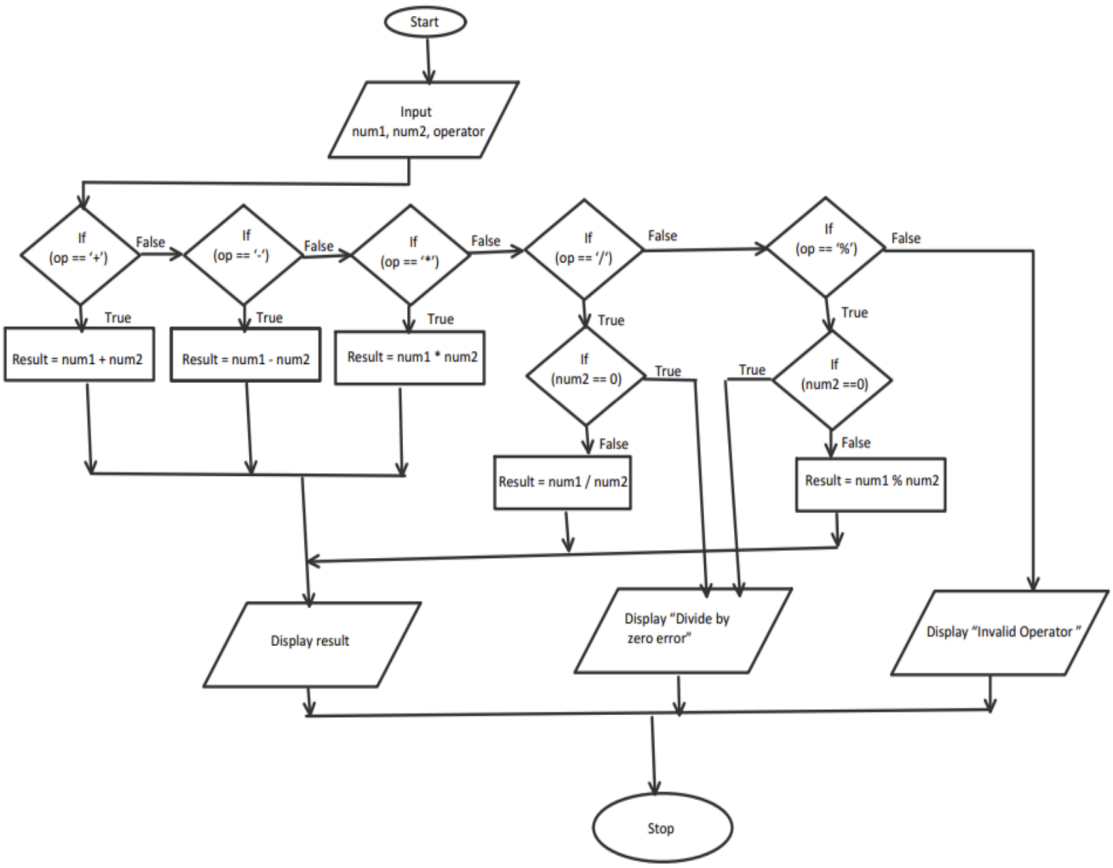
Step 18: Display the result
Step 19: Stop

FLOWCHART:



OR

Flow Chart:



PROGRAM CODE:

```
# include<stdio.h>
 int main()
{
        // Variable declaration
        int num1, num2;
        int result;
        char op;
```

```
// Accept the input
printf("Enter the operator \n");
scanf("%c",&op);
printf("Enter two integers :");
scanf("%d%d", &num1,&num2);

if (op == '+')
{
        result=num1+num2;
}
else if (op == '-')
{
        result=num1-num2;
}
else if (op == '*')
{
        result=num1*num2;
}
else if (op == '/')
{
        if (num2 == 0)
        {
                printf("Divide by zero error \n");
                return 1;
        }
        else
        {
                result=num1/num2;
        }
}
 else if (op == '%')
 {
        if (num2 == 0)
        {
                printf("Divide by zero error \n");
```

```
                return 1;
        }
        else
        {
                result=num1%num2;
         }
    }
    else
    {
        printf("Invalid operator...\n");
        return 1;
    }


    printf("%d %c %d  = %d\n", num1, op, num2, result);
    return 0;
} // End of main function
```

SAMPLE OUTPUT:

QUESTIONS FOR VIVA:

1. Why do we need to include a stdio.h file in a C program?
2. What is an operator in C?
3. What are the different types of arithmetic operators?
4. What are the different types of relational operators?
5. What is the difference between / and % operator?
6. Why do we use printf()?
7. Why do we use scanf()?
8. Explain looping structures in C.
9. Explain decision control structures in C.

Prog 2: To compute the roots of quadratic equation

AIM: Compute the roots of a quadratic equation by accepting the coefficients. Print appropriate messages.

PROBLEM STATEMENT: Design and develop a flowchart or an algorithm that accepts 3 coefficients of a quadratic equation as input and compute all possible roots. Implement a C program for the developed flowchart and execute to find the possible roots for the given set of coefficients and print messages accordingly.

ALGORITHM:

Input: Three non zero coefficients a,b and c of a quadratic equation.
Output: To compute roots for the quadratic equation.

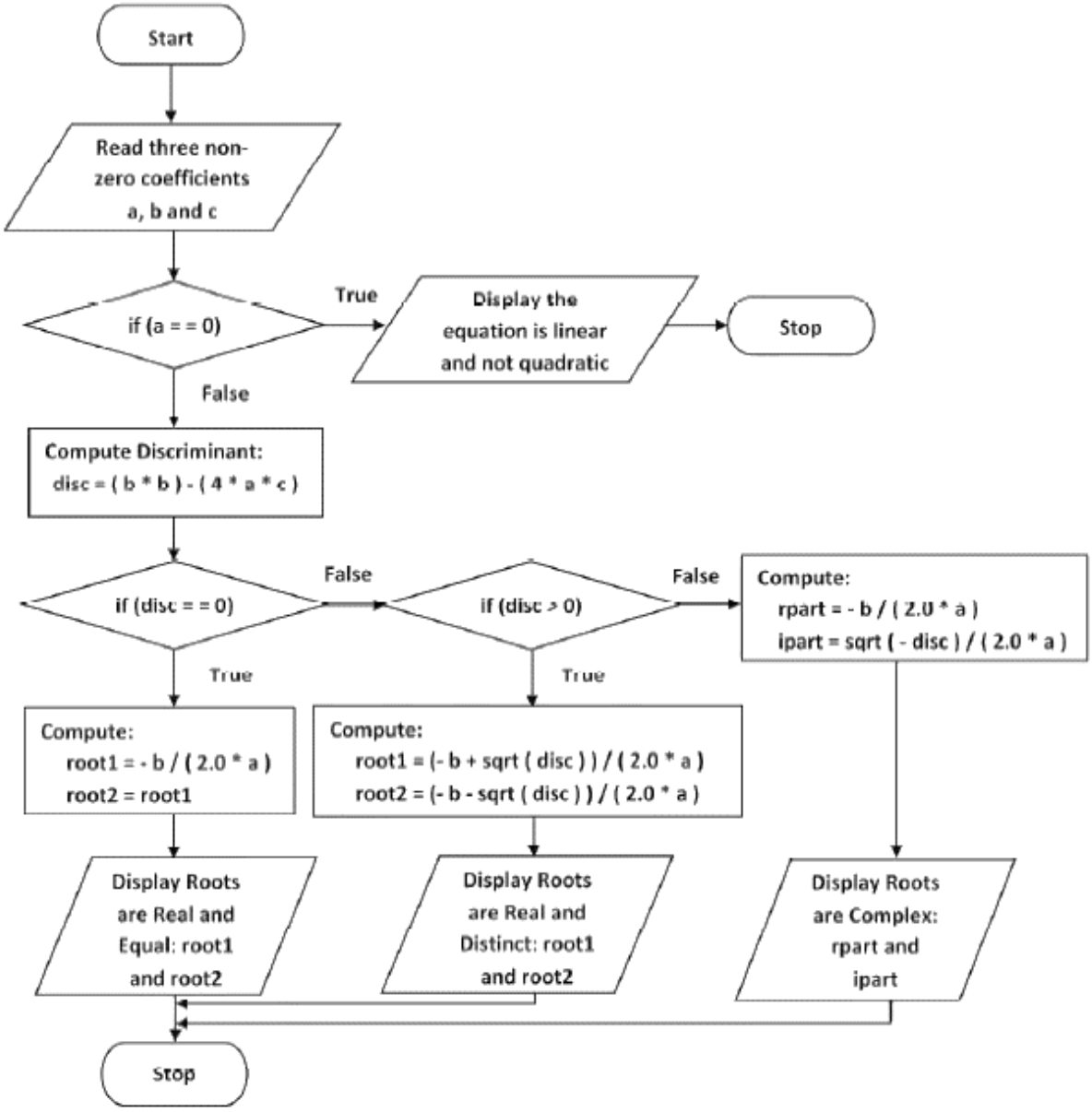
1. Start
2. Read three non zero coefficients a,b and c.
3. Check if a is equal to zero. If true, go to step 4 else go to step 5.
4. Display the equation is linear and go to step 11.
5. Calculate the discriminant as follows:
   D= b*b - 4* a*c
6. Check if discriminant is 0. If true, go to step 7, else go to step 8.
7. Compute roots as follows:
   Root1 = -b/2*a
   Root2 = root 1
   Display: Roots are real and equal. Root1. Go to step 11.
8. Check is discriminant is greater than 0. If true, go to step 9 else go to step 10.
9. Compute roots as follows:

   root1= -b + sqrt(D)/(2*a)
   Root2 = -b -sqrt(D)/(2*a)

   Display: Roots are real and distinct. Root1 and Root2. Go to step 11.

10. Calculate
    Rpart = -b/2*a
    Ipart = sqrt(-D)/(2*a)
    Display. Roots are imaginary. Rpart and ipart. Go to step 11.
11. Stop

FLOWCHART:


Start
Read three non-zero coefficients a, b and c
if (a = = 0)
True
Display the equation is linear and not quadratic
Stop
False
Compute Discriminant: disc = (b * b) - (4 * a * c)
if (disc = = 0)
False
if (disc > 0)
False
Compute: rpart = - b / (2.0 * a) ipart = sqrt (- disc) / (2.0 * a)
True
True
Compute: root1 = - b / (2.0 * a) root2 = root1
Compute: root1 = (- b + sqrt (disc)) / (2.0 * a) root2 = (- b - sqrt (disc)) / (2.0 * a)
Display Roots are Real and Equal: root1 and root2
Display Roots are Real and Distinct: root1 and root2
Display Roots are Complex: rpart and ipart
Stop

PROGRAM CODE:

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

int main()
{
        float a,b,c,desc,r1,r2,realpart,imgpart;

        printf("Enter the coefficients of a, b and c :");
        scanf("%f%f%f",&a,&b,&c);

        if(a == 0)
        {
                printf("Coefficient of a cannot be zero....\n");
                printf("Please try again....\n");
                return 1;
        }

        desc=(b*b)-(4.0*a*c);

        if(desc==0)
        {
                printf("The roots are real and equal\n");
                r1=r2=(-b)/(2.0*a);
                printf("The two roots are r1=r2=%f\n",r1);
        }
        else if(desc>0)
        {
                printf("The roots are real and distinct\n");
                r1=(-b+sqrt(desc))/(2.0*a);
                r2=(-b-sqrt(desc))/(2.0*a);
                printf("The roots are r1=%f and r2=%f\n",r1,r2);
        }
        else
        {
                printf("The roots are imaginary\n");
                realpart=(-b)/(2.0*a);
                imgpart=sqrt(-desc)/(2.0*a);
                printf("The roots are r1=%f + i %f\n",realpart,imgpart);
                printf("r2=%f - i %f\n",realpart,imgpart);
        }
```

```
        return 0;
}
```

SAMPLE OUTPUT:

cc Prog2.c -lm

./a.out
Enter the coefficients of a, b and c :1 2 1
The roots are real and equal
The two roots are r1=r2=-1.000000

./a.out
Enter the coefficients of a, b and c :1 -2 1
The roots are real and equal
The two roots are r1=r2=1.000000

./a.out
Enter the coefficients of a, b and c :1 7 12
The roots are real and distinct
The roots are r1=-3.000000 and r2=-4.000000

./a.out
Enter the coefficients of a, b and c :1 1 1
The roots are imaginary
The roots are r1=-0.500000 + i 0.866025
r2=-0.500000 - i 0.866025

QUESTIONS FOR VIVA:

1. What are library files? How are they called in a C program?
2. What are format strings? Explain with example.
3. How can we find the square root of a given number without using sqrt()?
4. What is a preprocessor? Which preprocessor is used in this program?
5. Why do we use \n ?
6. What is the return type of main function? Can we use void()? If yes, explain what changes need to be made to the program.

Prog 3: 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 the total amount is charged. Write a program to read the name of the user, the number of units consumed, and print out the charges.

AIM: 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.

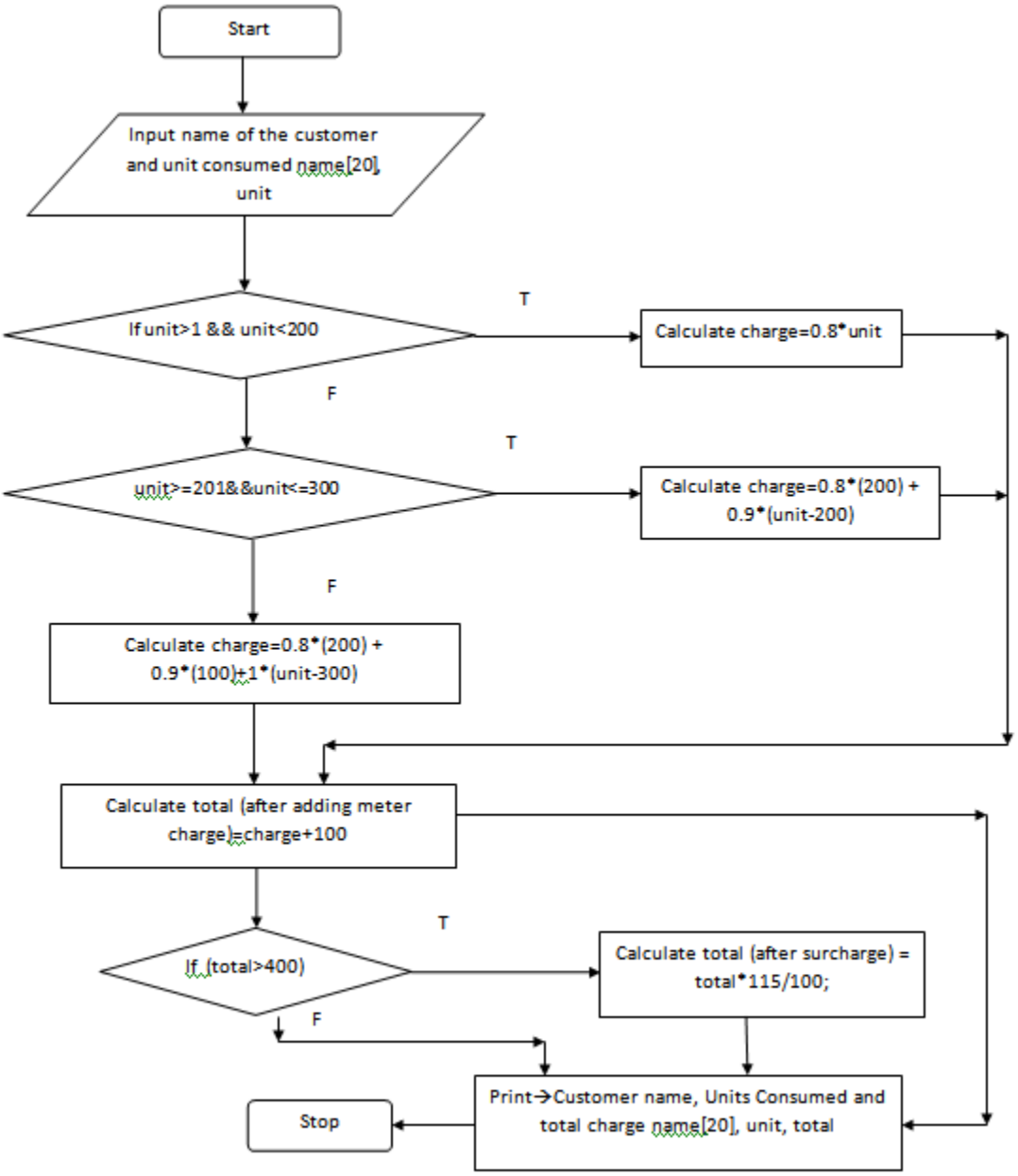
PROBLEM STATEMENT: Write a C program to demonstrate the calculations followed while computing electricity bill.

ALGORITHM:
Input: Customer name and unit consumed
Output: Customer name, units consumed and total amount to be paid

1. Start
2. Read the name of customer and the unit consumed by the customer
3. Check if the unit consumed is greater than 1 and less than 200,if true goto step 4 else goto step 5
4. Compute: Charge=0.8*unit consumed
5. Check if unit is greater than or equal 201 and less than 300,if true goto step 6 else goto step 7

6. Compute: Charge=0.8*(200)+0.9*(unit 200)
7. Compute: Charge=0.8*200+0.9*(100)+1*(unit-300)
8. After calculating charge ,find total:
   Total=charge+100
9. Check if total is greater than 400,if true goto step 10 else goto step 11
10. Compute total(after surcharge)=total *115/100 and proceed to step 11
11. Display the customer name, Units Consumed and total charge per unit
12. Stop

FLOWCHART:


Start
Input name of the customer and unit consumed name[20], unit
If unit>1 && unit<200
T
Calculate charge=0.8*unit
F
unit>=201&&unit<=300
T
Calculate charge=0.8*(200) + 0.9*(unit-200)
F
Calculate charge=0.8*(200) + 0.9*(100)+1*(unit-300)
Calculate total (after adding meter charge)=charge+100
If (total>400)
T
Calculate total (after surcharge) = total*115/100;
F
Print→Customer name, Units Consumed and total charge name[20], unit, total
Stop

PROGRAM CODE:

```
        #include <stdio.h>
 int main()
 {
        float unit,total,charge;
        char name[20];

        // Accept Customer's name
        printf("Enter the name : ");
        gets(name);

        // Accept No. of units consumed
        printf("Enter the units : ");
        scanf("%f",&unit);

        charge=0;

        if(unit>=1&&unit<=200)                  // Charge 80 paise per unit for the first 200 units
        {
                charge=0.8 * unit;
        }
        else if(unit>=201&&unit<=300)           // Charge 80 paise per unit for the first 200 units and
        {                                       // 90 paise per unit for the next 100 units
                charge = 0.8*(200) + 0.9*(unit-200);
        }
        else if (unit > 300)                    //Charge 80 paise per unit for the first 200 units,
        {                                       // 90 paise per unit for the next 100 units and
                                                // one rupee per unit for all units above 300
                        charge = 0.8*(200) + 0.9*(100) + 1*(unit-300);

        }
        total=charge+100;                       // Add Meter charge of Rs. 100
        if(total>400)
        {
                total = total + (0.15*total);       // Add additional surcharge of 15 percent of total
amount
        }

        // Display the electricity bill
        printf("\n\nELECTRICITY BILL\n");
        printf("----------------\n");
```

```
        printf("\nName : %s\n",name);
        printf("No. of units: %.2f\n",unit);
        printf("Total Bill Amount: Rs. %.2f\n",total);
        printf("----------------\n");

        return 0;
}
```

SAMPLE OUTPUTS:

1.

```
$ cc prog3.c
$ ./a.out
Enter the name : Anitha
Enter the units : 0


ELECTRICITY BILL
----------------

Name : Anitha
No. of units: 0.00
Total Bill Amount: Rs. 100.00
----------------
```

2.

```
$ ./a.out
Enter the name : Harish
Enter the units : 167


ELECTRICITY BILL
----------------

Name : Harish
No. of units: 167.00
Total Bill Amount: Rs. 233.60
----------------
```

3.

```
$ ./a.out
Enter the name : Anuradha
```

Enter the units : 598

ELECTRICITY BILL

Name : Anuradha
No. of units: 598.00
Total Bill Amount: Rs. 745.20

4.
$ ./a.out
Enter the name : Rajeev Gupta
Enter the units : 876

ELECTRICITY BILL

Name : Rajeev Gupta
No. of units: 876.00
Total Bill Amount: Rs. 1064.90

QUESTIONS FOR VIVA:

1. What is the difference between getc(), getch() and gets()?
2. What is the difference between printf() and scanf()?
3. What are looping structures in C? Which looping structure have you used in your C program? Justify.
4. What are control structures in C? Give some examples.
5. Suppose you want to limit the input by user upto 2 decimal places. Which format specifier will you use?
6. When do we use && operator?
7. What are relational operators? Give some examples.

Prog 4: Implement Binary Search on Integers / Names.

AIM: Implement Binary Search on Integers / Names

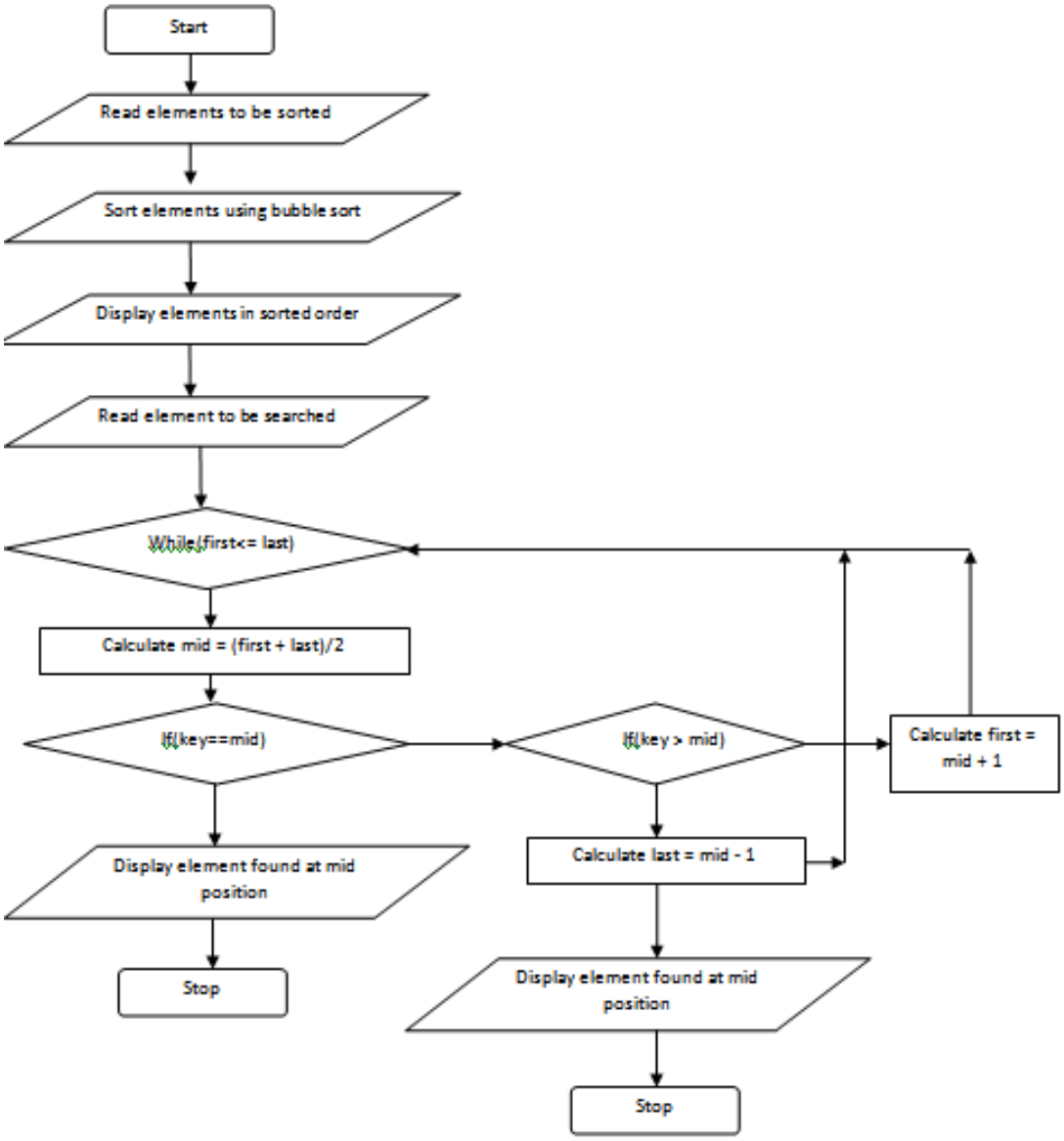
PROBLEM STATEMENT: Write a C program to demonstrate the performance of binary search.

ALGORITHM:

Input: List of unsorted elements and elements to be searched.
Output: Sorted list and element is present or not.

1. Start
2. Read size of the array and list of elements which are not sorted.
3. Use swap function in a loop to sort the given array.
4. Display the sorted list of elements.
5. Read the element to be searched. key.
6. Check if first is not equal to last. If true, find the middle element.
7. A. Check if the key is equal to the middle element. If true, element is found at n/2 index and exit loop. Go to step 8.
   B. Else, check if the key is greater than middle element. If true, change first to mid. Go to step 6.
   C. Else, check if the key is smaller than middle element. If true, change last to mid. Go to step 6.
8. Stop.

FLOWCHART:


Start
Read elements to be sorted
Sort elements using bubble sort
Display elements in sorted order
Read element to be searched
While(first<= last)
Calculate mid = (first + last)/2
If(key==mid)
If(key > mid)
Calculate first = mid + 1
Display element found at mid position
Calculate last = mid - 1
Stop
Display element found at mid position
Stop

PROGRAM CODE :

```
#include <stdio.h>
int main()
{
        // Define variables
        int a[20];
        int n,i,j,temp,key;
        int first,mid,last;

        // Accept the size of the array
        printf("Enter the size of the array :");
        scanf("%d",&n);

        // Accept elements into the array
        printf("Enter %d elements :",n);
        for(i=0;i<n;i++)
        {
                scanf("%d",&a[i]);
        }

        // Print the elements of the array before sorting
        printf("The elements of the array before sorting is ----\n");
        for(i=0;i<n;i++)
        {
                printf("%d\t",a[i]);
        }

        /* Sort the array as the data
          must be sorted for binary search
         */

        for(i=0;i<n-1;i++)
        {
```

```
        for(j=0;j<n-1-i;j++)
        {
                if(a[j]>a[j+1])
                {
                        temp=a[j];
                        a[j]=a[j+1];
                        a[j+1]=temp;
                }
        }
}

// Display the sorted elements of the array
printf("\n\nThe sorted array is ---\n");
for(i=0;i<n;i++)
{
        printf("%d\t",a[i]);
}

// Accept the element to be searched
printf("\n\nEnter the element to be searched :");
scanf("%d",&key);

// search for the element in the sorted array
first=0;
last=n-1;

while(first <= last)
{
        mid=(first+last)/2;
        if(key==a[mid])
        {
                printf("\nThe element %d is found at location %d\n",key,mid+1);
                return (0);
        }
        else if (key < a[mid])
        {
                last = mid-1;
        }
        else
        {
                first = mid+1;
```

```
                }
        }

        printf("\nThe element %d is not found in the array\n",key);
        return (1);
}
```

SAMPLE OUTPUT:

Sample Output 1: Search for the first element in the array

```
$ ./a.out
Enter the size of the array :5
Enter 5 elements :87
0
4
2
3
The elements of the array before sorting is ----
87      0       4       2       3

The sorted array is ---
0       2       3       4       87

Enter the element to be searched :0

The element 0 is found at location 1
```

Sample Output 2: Search for the middle element in the array

```
$ cc prog6.c
$ ./a.out
Enter the size of the array :5
Enter 5 elements :2
3
4
87
0
The elements of the array before sorting is ----
2       3       4       87      0

The sorted array is ---
0       2       3       4       87
```

Enter the element to be searched :4

The element 4 is found at location 4

Sample Output 3: Search for middle element in the array

```
$ ./a.out
Enter the size of the array :5
Enter 5 elements :2
0
4
3
87
The elements of the array before sorting is ----
2       0       4       3       87

The sorted array is ---
0       2       3       4       87

Enter the element to be searched :3

The element 3 is found at location 3
```

Sample Output 4: Search for the last element in the list

```
$ ./a.out
Enter the size of the array :5
Enter 5 elements :3
2
4
0
87
The elements of the array before sorting is ----
3       2       4       0       87

The sorted array is ---
0       2       3       4       87

Enter the element to be searched :87
```

The element 87 is found at location 5
*/

QUESTIONS FOR VIVA:

1. What is binary search?
2. Why do we need to perform sorting?
3. What is an array? Give an example.
4. Can an array store both integer and float variables?

Prog 5: Implement Matrix multiplication and validate the rules of multiplication

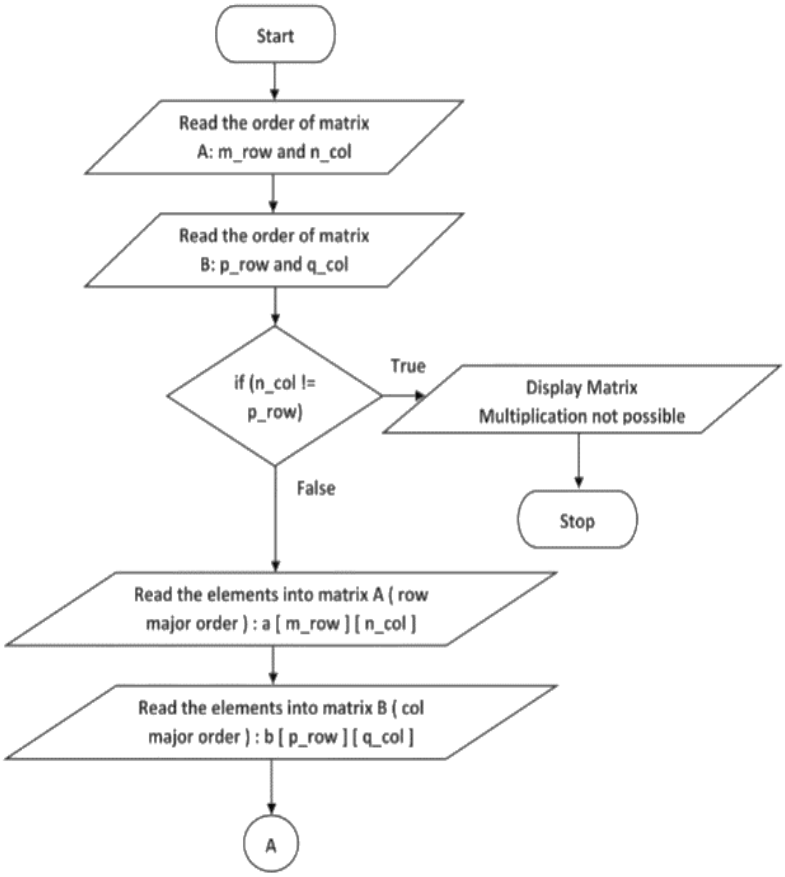
AIM: Implement Matrix multiplication and validate the rules of multiplication

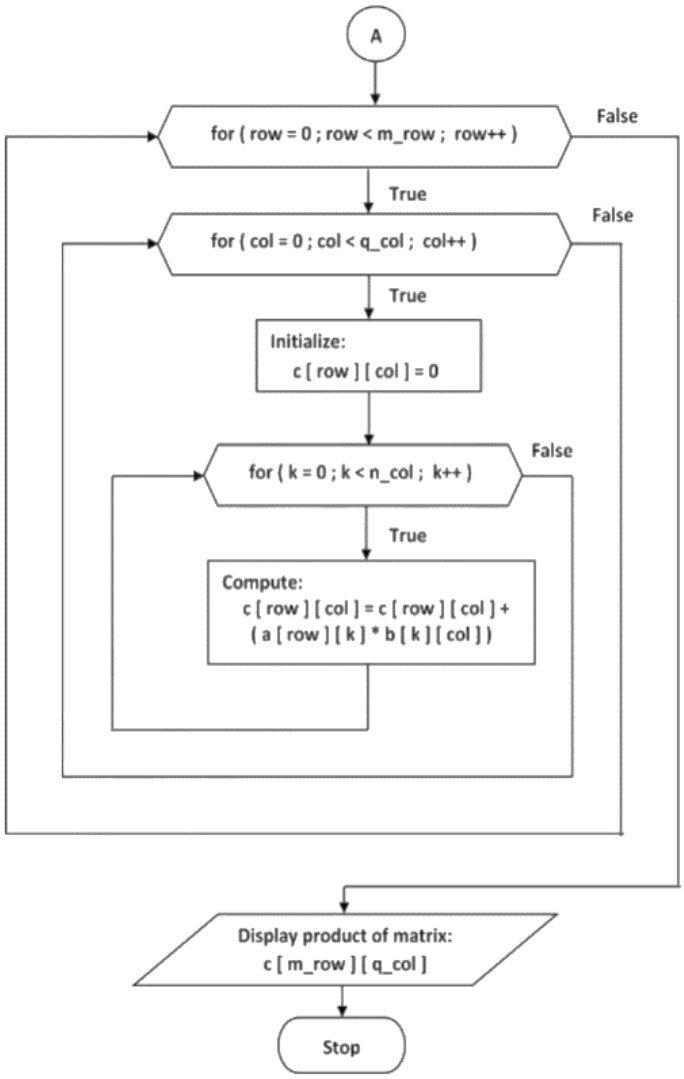
PROBLEM STATEMENT: Write a C program to input matrices A and B. Check for the number of columns in first matrix is equal to number of rows in second matrix. If true, perform matrix multiplication.

ALGORITHM:
Input: Two matrices A and B.
Output: Result of A x B.

1. Start
2. Input 2 matrices A and B and store them in 2D arrays. Display A and B.
3. Check multiplication condition where the number of columns in A is equal to the number of rows in B. If true, go to step 4. If false, display that matrix multiplication is not possible.
4. Perform matrix multiplication. Display resultant matrix.
5. Exit.

FLOWCHART


Start
Read the order of matrix A: m_row and n_col
Read the order of matrix B: p_row and q_col
if (n_col != p_row)
True
Display Matrix Multiplication not possible
Stop
False
Read the elements into matrix A (row major order) : a [m_row] [n_col]
Read the elements into matrix B (col major order) : b [p_row] [q_col]
A



PROGRAM CODE:

```
#include<stdio.h>

int main()
{
        int a[10][10],b[10][10],c[10][10];
        int m,n,p,q;
        int i,j,k;

        // Input the order of Matrix A - m x n
        printf("Enter the order of matrix A :");
        scanf("%d%d",&m,&n);
```

```
        // Input the order of Matrix B - p x q
        printf("Enter the order of matrix B:");
        scanf("%d%d",&p,&q);

        /* For multiplication of two matrices, the number of columns in the first
           matrix should be equal to the number of rows in the second matrix */
        if(n!=p)
        {
                printf("Number of columns of Matrix A is not equal to number of rows of matrix
B\n");
                printf("Multiplication of matrices not possible....\n");
                return (-1);
        }

        // Input the elements into Matrix A
        printf("\nEnter %d elements into matrix A : ", m*n);
        for(i=0;i<m;i++)
                for(j=0;j<n;j++)
                        scanf("%d",&a[i][j]);

        // Display matrix A in matrix format
        printf("\nThe  matrix A is ---\n");
        for(i=0;i<m;i++)
        {
                for(j=0;j<n;j++)
                {
                        printf("%d\t",a[i][j]);
                }
                printf("\n");
        }

        // Input the elements into Matrix B
        printf("\nEnter %d elements into matrix B : ", p*q);
        for (i=0;i<p;i++)
                for (j=0;j<q;j++)
                        scanf("%d",&b[i][j]);

        // Display Matrix B in matrix format
        printf("\nThe matrix B is ---\n");
        for(i=0;i<p;i++)
        {
                for(j=0;j<q;j++)
```

```
            {
                printf("%d\t",b[i][j]);
            }
            printf("\n");
        }

        // Compute (Matrix A) X (Matrix B)
        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]);
                }
            }
        }


        // Display product matrix - Matrix C
        printf("\nThe product matrix is ---\n\n");
        for(i=0;i<m;i++)
        {
            for(j=0;j<q;j++)
            {
                printf("%d\t",c[i][j]);
            }
            printf("\n");
        }

        return 0;
}
```

SAMPLE OUTPUT:

1.

```
$cc prog5.c
$./a.out
Enter the order of matrix A :2
2
Enter the order of matrix B:2
```

```
2
Enter 4 elements into matrix A : 1
2
3
4
The  matrix A is ---
1       2
3       4
Enter 4 elements into matrix B : 1
3
2
4
The matrix B is ---
1       3
2       4
The product matrix is ---

5       11
11      25

2.
$./a.out
Enter the order of matrix A :3
2
Enter the order of matrix B:2
3

Enter 6 elements into matrix A : 6
5
4
3
2
1

The  matrix A is ---
6       5
4       3
2       1

Enter 6 elements into matrix B : 1
3
5
```

2
4
6

The matrix B is ---
1 3 5
2 4 6

The product matrix is ---

16 38 60
10 24 38
4 10 16

QUESTIONS FOR VIVA:

1. What is a array? What are the different types of arrays?
2. Consider a 5 x 5 matrix. What will be the index to fetch the element at the 3rd row and 3rd column?
3. What is increment operator?
4. What is decrement operator?
5. What is right to left associativity?
6. What is left to right associativity?

Prog 6: Compute sin(x)/cos(x) using Taylor series approximation. Compare your result with the built-in library function. Print both the results with appropriate inferences.

AIM

Compute sin(x)/cos(x) using Taylor series approximation. Compare your result with the built-in library function. Print both the results with appropriate inferences.

PROBLEM STATEMENT

Draw the flowchart and write a C program to compute Sin(x) using Taylor series approximation given by Sin(x) = x - (x^3/3!) + (X^5/5!) - (x^7/7!) + …. Compare your result with the built-in library function. Print both the results with appropriate messages

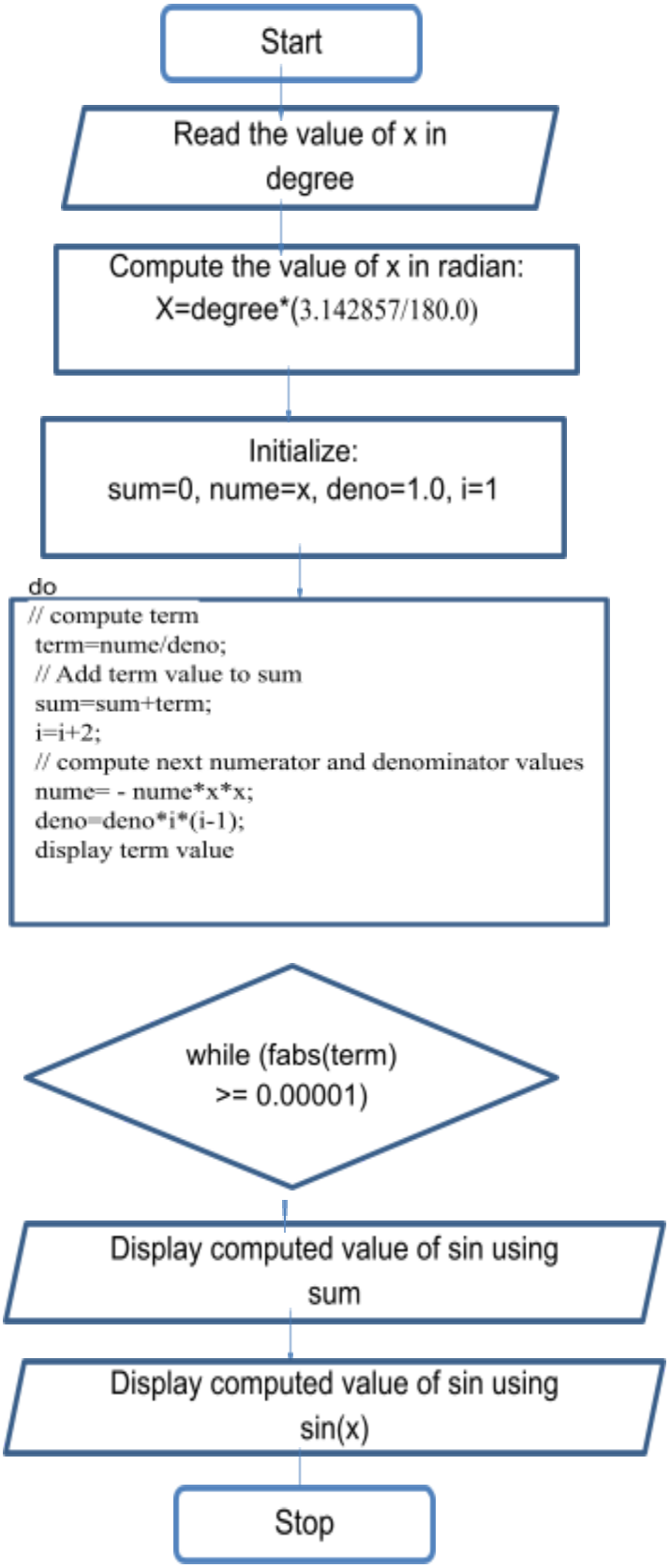
ALGORITHM

Input: The value of x in degree

Output: To compute Sin(x) using Taylor Series.

1. Start
2. Read the value of x in degree
3. Compute the value of x in radian: x = degree* (3.142857/180.0)
4. Initialise:
   sum=0;
   nume=x;
   deno=1.0;
   i=1;
5. compute term as , term=nume/deno;
   Add term value to sum: sum=sum+term;
   i=i+2;
   compute next numerator and denominator values
   nume= - nume*x*x
   deno=deno*i*(i-1)
   print term value
6. Check if the absolute value of term is greater than 0.0001. If true goto Step 5 else goto Step 7.
7. Display sin(x) using iteration : sum and display sin(x) using library function: sin(x)
8. Stop

FLOWCHART


Start
Read the value of x in degree
Compute the value of x in radian:
X=degree*(3.142857/180.0)
Initialize:
sum=0, nume=x, deno=1.0, i=1
do
// compute term
term=nume/deno;
// Add term value to sum
sum=sum+term;
i=i+2;
// compute next numerator and denominator values
nume= - nume*x*x;
deno=deno*i*(i-1);
display term value
while (fabs(term) >= 0.00001)
True
Display computed value of sin using sum
Display computed value of sin using sin(x)
Stop

PROGRAM CODE:

```
#include<stdio.h>
#include<math.h>
int main()
 {
        float x,nume,deno,term,degree;
        int i;
        printf("Enter degree:");
        scanf("%f",&degree);
        x=degree*(3.14/100);
        nume=x;
        deno=1.0;
        i=1;
do
        {
        term=nume/deno;
        sum=sum+term;
        i=i+2;
        nume= -nume*x*x;
        deno=deno*i*(i-1);
        }while (fabs(term) >= 0.00001);

printf("Computed value of Sin(%f)=%f\n",degree,sum);
printf("Value from library function is sin(%f) = %f\n",degree,sin(x));
return 0;
 }
```

—----------

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

#define PI 3.142857

int main()
```

```
{
        float x,degree,nume,deno,sum,term;
        int i;

        /* Accept value of x in degree */
        printf("Enter degree:");
        scanf("%f",&degree);

        // Convert degree into radians
        x=degree*(PI/180.0);

        // Initialize values of sum, nume, deno and i variables
        sum=0;
        nume=x;
        deno=1.0;
        i=1;

        do
        {
                // compute term
                term=nume/deno;
                // Add term value to sum
                sum=sum+term;
                i=i+2;
                // compute next numerator and denominator values
                nume= -nume*x*x;
                deno=deno*i*(i-1);
                // printf("Term=%f\n",term);
        } while (fabs(term) >= 0.00001);

        printf("Computed value of Sin(%f)=%f\n",degree,sum);

        printf("Value from library function is sin(%f) = %f\n",degree,sin(x));

        return 0;
}
```

SAMPLE OUTPUT:

1.

```
$cc prog6.c -lm
$./a.out
Enter degree:60
```

Computed value of Sin(60.000000)=0.866236
Value from library function is sin(60.000000) = 0.866236

2.
$ ./a.out
Enter degree:30
Computed value of Sin(30.000000)=0.500182
Value from library function is sin(30.000000) = 0.500182

3.
$ ./a.out
Enter degree:90
Computed value of Sin(90.000000)=1.000000
Value from library function is sin(90.000000) = 1.000000

4.
$ ./a.out
Enter degree:45
Computed value of Sin(45.000000)=0.707330
Value from library function is sin(45.000000) = 0.707330

5.
$ ./a.out
Enter degree:0
Computed value of Sin(0.000000)=0.000000
Value from library function is sin(0.000000) = 0.000000
*/

QUESTIONS FOR VIVA

1. What is the difference between sin(), sinf(). sinl()?

Prog 7: Sort the given set of N numbers using Bubble sort.

AIM: Sort the given set of N numbers using Bubble sort.

PROBLEM STATEMENT

Develop an algorithm, implement and execute a C program that reads N integer numbers and arrange them in ascending order using BUBBLE SORT

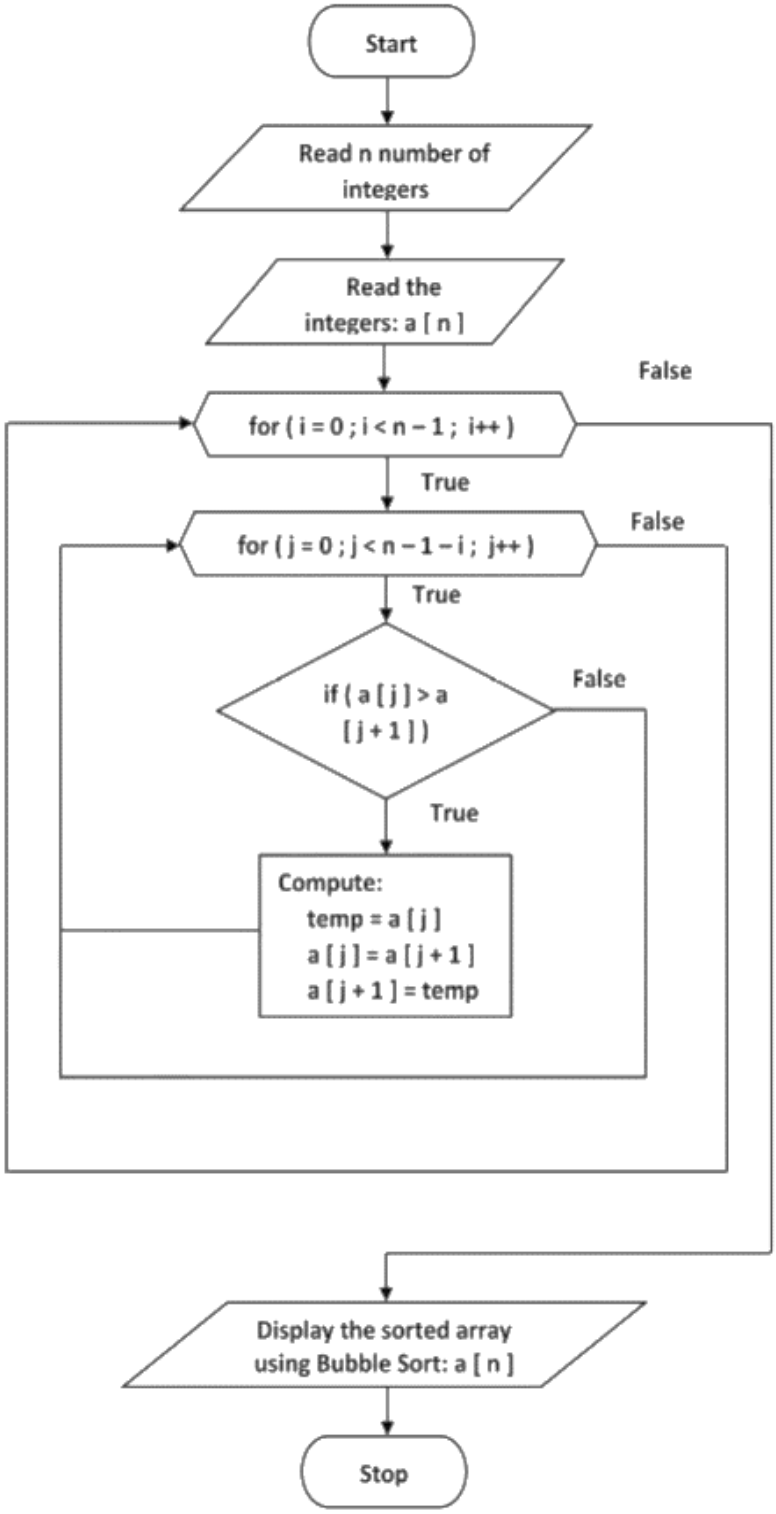
ALGORITHM

Input: Size of the array and elements to be sorted
Output: List of elements in sorted order

1. Start
2. Read the size of array: a[n]
3. Read the elements of array: n
4. Check if i<n-1 where i is the position of elements in the array and n is the total count of elements in the array. If the condition is true go to step 5 and increment i value. Repeat this step until the condition is false. If the condition is false go to step 8.
5. Check if j<n-1-i, if true go to step 6 and increment j value. Repeat this step until the condition is false. If the condition is false go to step 4.
6. check if (a[j] > a[j+1]), if true go to step 7 else go to step 5
7. Compute:

   temp=a[j]
   a[j]=a[j+1]
   a[j+1]=temp
8. Display the elements of array in sorted order: a[n]
9. Stop

FLOWCHART


Start
Read n number of integers
Read the integers: a [n]
for (i = 0 ; i < n – 1 ; i++)
False
True
for (j = 0 ; j < n – 1 – i ; j++)
False
True
if (a [j] > a [j + 1])
False
True
Compute:
temp = a [j]
a [j] = a [j + 1]
a [j + 1] = temp
Display the sorted array using Bubble Sort: a [n]
Stop

PROGRAM CODE:

```
#include<stdio.h>

int main()
{
        int a[20],n,i,j,temp;

        // Accept array size
        printf("Enter the number of elements :");
        scanf("%d",&n);

        // Accept n elements into the array
        printf("Enter %d integers :",n);
        for(i=0;i<n;i++)
        {
                scanf("%d",&a[i]);
        }

        // Sort the array elements

        for(i=0;i<n-1;i++)
        {
                for(j=0;j<n-1-i;j++)
                {
                        if(a[j] > a[j+1])
                        {
                                temp=a[j];
                                a[j]=a[j+1];
                                a[j+1]=temp;
                        }
                }
        }

        // print the sorted array
        printf("The sorted array is ....\n");
        for(i=0;i<n;i++)
        {
                printf("%d\t",a[i]);
        }
```

```
        printf("\n");
        return 0;
}
```

SAMPLE OUTPUT

1.

```
$ cc prog7.c
$ ./a.out
Enter the number of elements :5
Enter 5 integers :1
3
4
2
5
The sorted array is ....
1       2       3       4       5
```

2.

```
$ ./a.out
Enter the number of elements :6
Enter 6 integers :23
55
10
-10
0
-400
The sorted array is ....
-400    -10     0       10      23      55
```

QUESTIONS FOR VIVA

1. What is bubble sorting?
2. How does nested for loop works?
3. What are the passes in a bubble sorting method?

Prog 8: Write functions to implement string operations such as compare, concatenate, string length. Convince the parameter passing techniques.

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

PROBLEM STATEMENT: Write a C program to demonstrate the working of string functions given in string library file.

ALGORITHM:

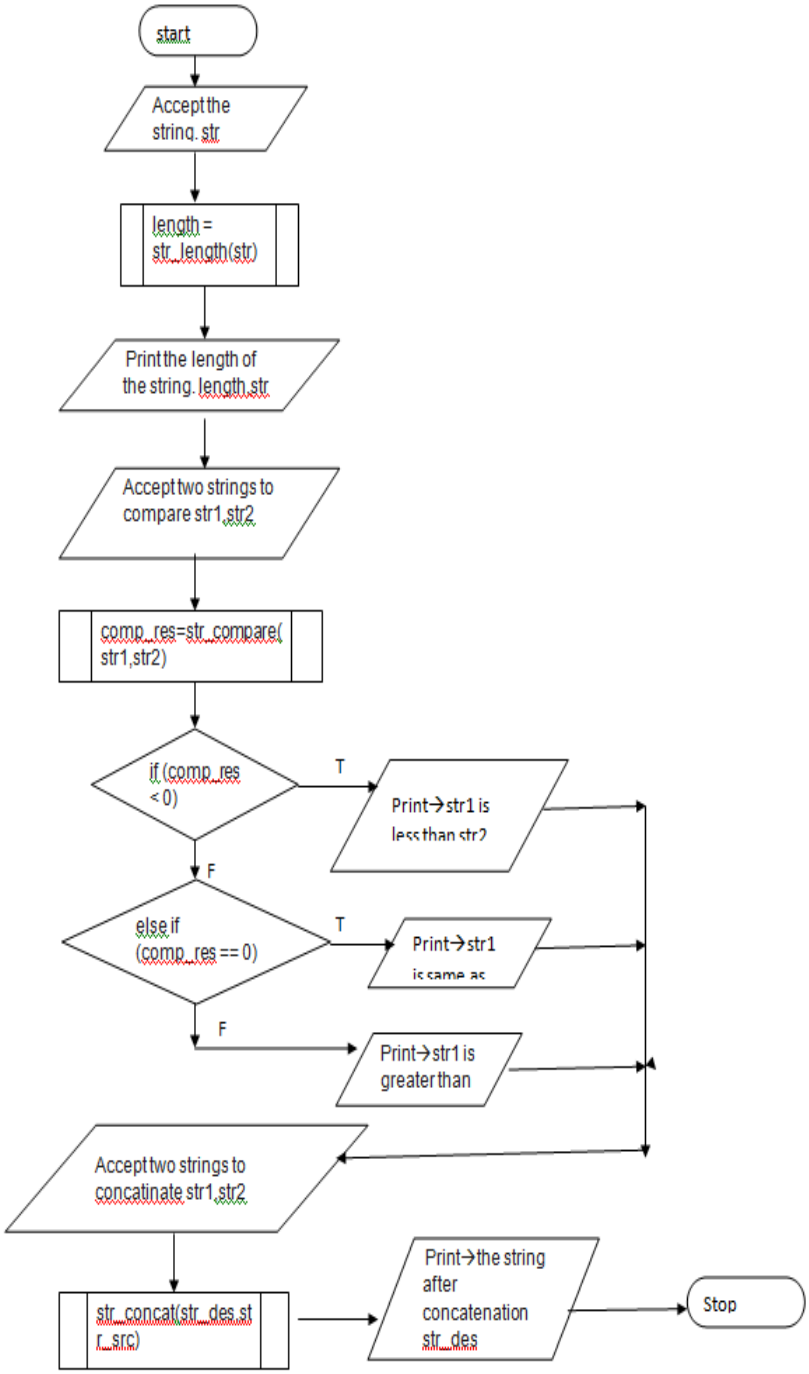
10.a.String length:
1. Start
2. Read the string given by the user
3. To find the length of string, call the function str_length (str)
4. In str_length (str): count the characters in the string one by one until it encounters null character then goto step 5
5. Display the length of string.

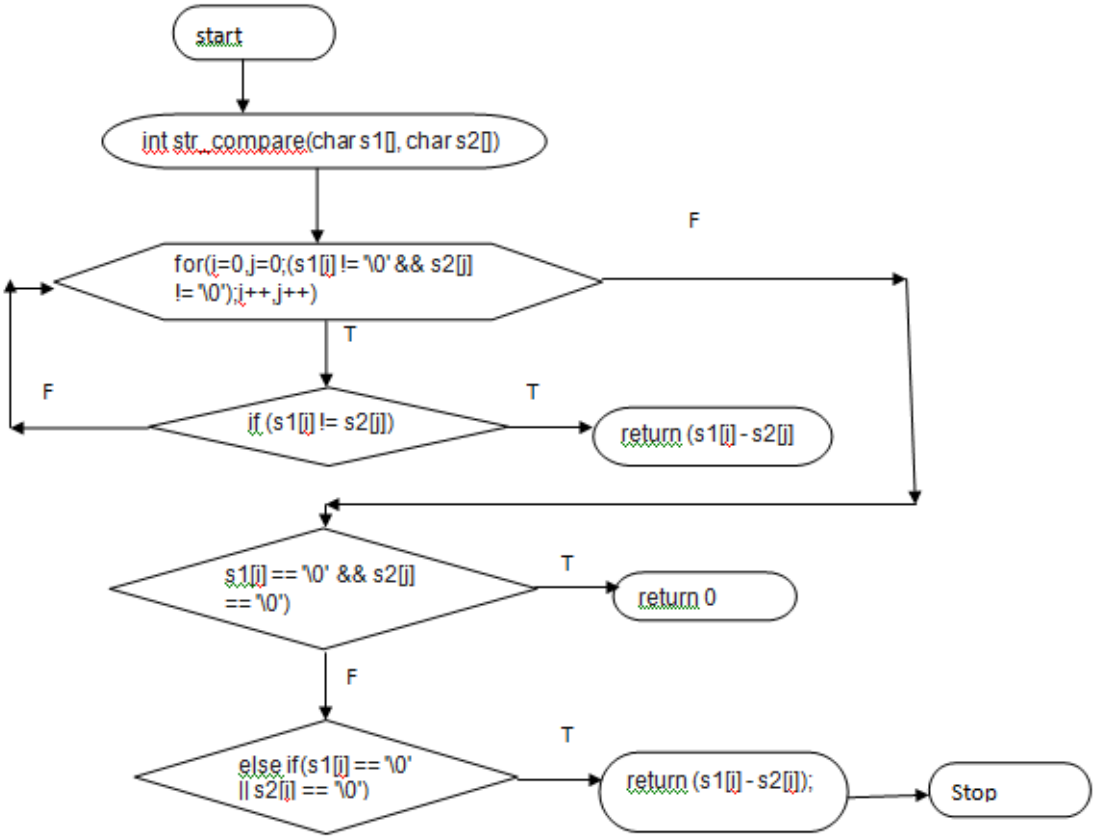
10.b String Compare:
1. Start
2. Read two strings str1 and str2 given by the user
3. To find comparison between two strings goto step 4
4. Call the function str_compare(str1,str2)
    a. Compare ASCII value of a character in string1 with string2,if both value equals then goto step 5
    b. Check if ASCII value of character in string1 is greater than string2 character then goto step 6
    c. Check if ASCII value of character in string1 is lesser than string2 character then goto step 7
5. Display both strings are same
6. Display String1 is greater than String2
7. Display String1 is lesser than String2
8. Stop

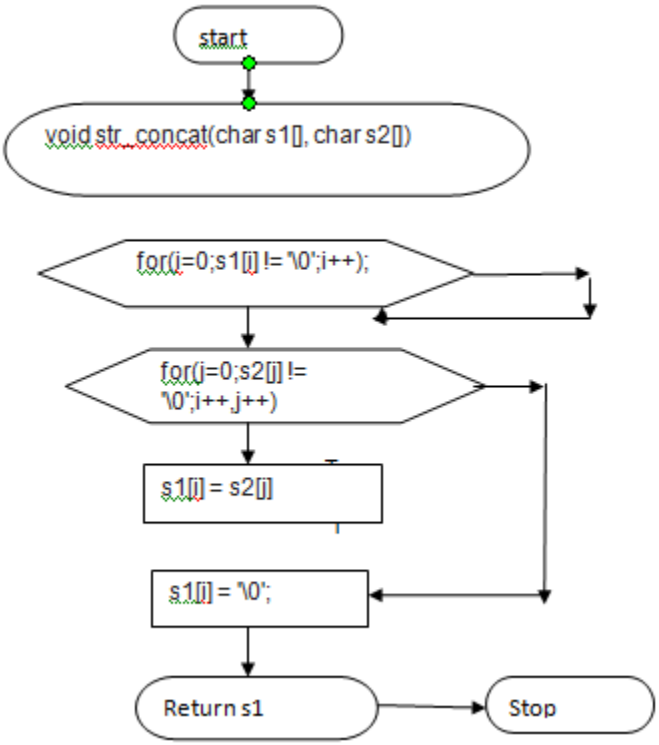
10. c. String Concatenation
1. Start
2. Read two strings str1, str2 from user
3. Call the concatenation function str_concat (str1, str2)
    a. Find the last character in string1 by checking the null character,then copy each character in string2 to string1
    b. After concatenation of string2 to string1, append null character at the end of string1.

4. Display the concatenated string
5. Stop

FLOWCHART




start
int str_compare(char s1[], char s2[])
for(i=0,j=0;(s1[j] != '\0' && s2[j] != '\0');i++,j++)
F
T
if (s1[j] != s2[j])
F
T
return (s1[j] - s2[j]
s1[j] == '\0' && s2[j] == '\0')
T
return 0
F
else if(s1[j] == '\0' || s2[j] == '\0')
T
return (s1[j] - s2[j]);
Stop


start
void str_concat(char s1[], char s2[])
for(j=0;s1[j] != '\0';i++);
for(j=0;s2[j] != '\0';i++,j++)
s1[j] = s2[j]
s1[j] = '\0';
Return s1
Stop

PROGRAM CODE:

```
#include <stdio.h>
// Prototypes for the user defined functions
int str_length(char []);
int str_compare(char [], char []);
void str_concat(char [], char []);

int main()
{
        // Declare the variables
        char str[50];
        char str1[50], str2[50];
        char str_des[100], str_src[50];

        int length, comp_res;

        // Accept the string from the user to find the length
        printf("\nEnter a string :");
        scanf("%s",str);

        // Invoke the function to find the length of the string
        length = str_length(str);

        // Print the length of the string
        printf("The length of %s is %d\n",str,length);

        // Accept two strings to compare
        printf("\nEnter two strings for string compare :");
        scanf("%s%s",str1,str2);


        // Invoke string compare function to compare the str1 and str2 strings
        comp_res=str_compare(str1,str2);

        if (comp_res < 0)
        {
                printf("String \"%s\" is less than string \"%s\"\n",str1,str2);
        }
        else if (comp_res == 0)
        {
```

```
                printf("String \"%s\" is same as string \"%s\"\n",str1,str2);
        }
        else
        {
                printf("String \"%s\" is greater than string \"%s\"\n", str1,str2);
        }

        // Accept two strings for string concatenation
        printf("\nEnter two strings for string concatenation :");
        scanf("%s%s",str_des,str_src);

        // Invoke string concatenation function
        str_concat(str_des,str_src);

        // Print the concatenated string
        printf("The string after concatenation is \"%s\"\n", str_des);

        return 1;
}

int str_length(char s[])
{
        int i;
        for(i=0;s[i]!='\0';i++);
        return i;
}


int str_compare(char s1[], char s2[])
{
        int i,j;
        for(i=0,j=0;(s1[i] != '\0' && s2[j] != '\0');i++,j++)
        {
                if (s1[i] != s2[j])
                {
                        return (s1[i] - s2[j]);
                }
        }
        if (s1[i] == '\0' && s2[j] == '\0')
        {
                return 0;
        }
```

```
        else if(s1[i] == '\0' || s2[i] == '\0')
        {
                return (s1[i] - s2[i]);
        }

}



void str_concat(char s1[], char s2[])
{
        int i,j;
        for(i=0;s1[i] != '\0';i++);

        for(j=0;s2[j] != '\0';i++,j++)
        {
                s1[i] = s2[j];
        }
        s1[i] = '\0';
}
```

SAMPLE OUTPUT:

1.

```
$ cc prog8.c
$ ./a.out
```

2.

```
$ ./a.out

Enter a string :thinkpad
The length of thinkpad is 8

Enter two strings for string compare :thinkpad
think
String "thinkpad" is greater than string "think"

Enter two strings for string concatenation :think
pad
The string after concatenation is "thinkpad"
```

*/

QUESTIONS FOR VIVA

1. What is string concatenation?
2. How does string compare take place?
3. What is the terminating character while counting string length?
4. What is the length of a string?

Prog 9: 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.

AIM

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

PROBLEM STATEMENT

Write a C program to maintain a record of n student details using an array of structures with fields to store student id, name, marks of subjects and the average. Assume appropriate data type for each field. Print the marks of the student scoring above and below average.

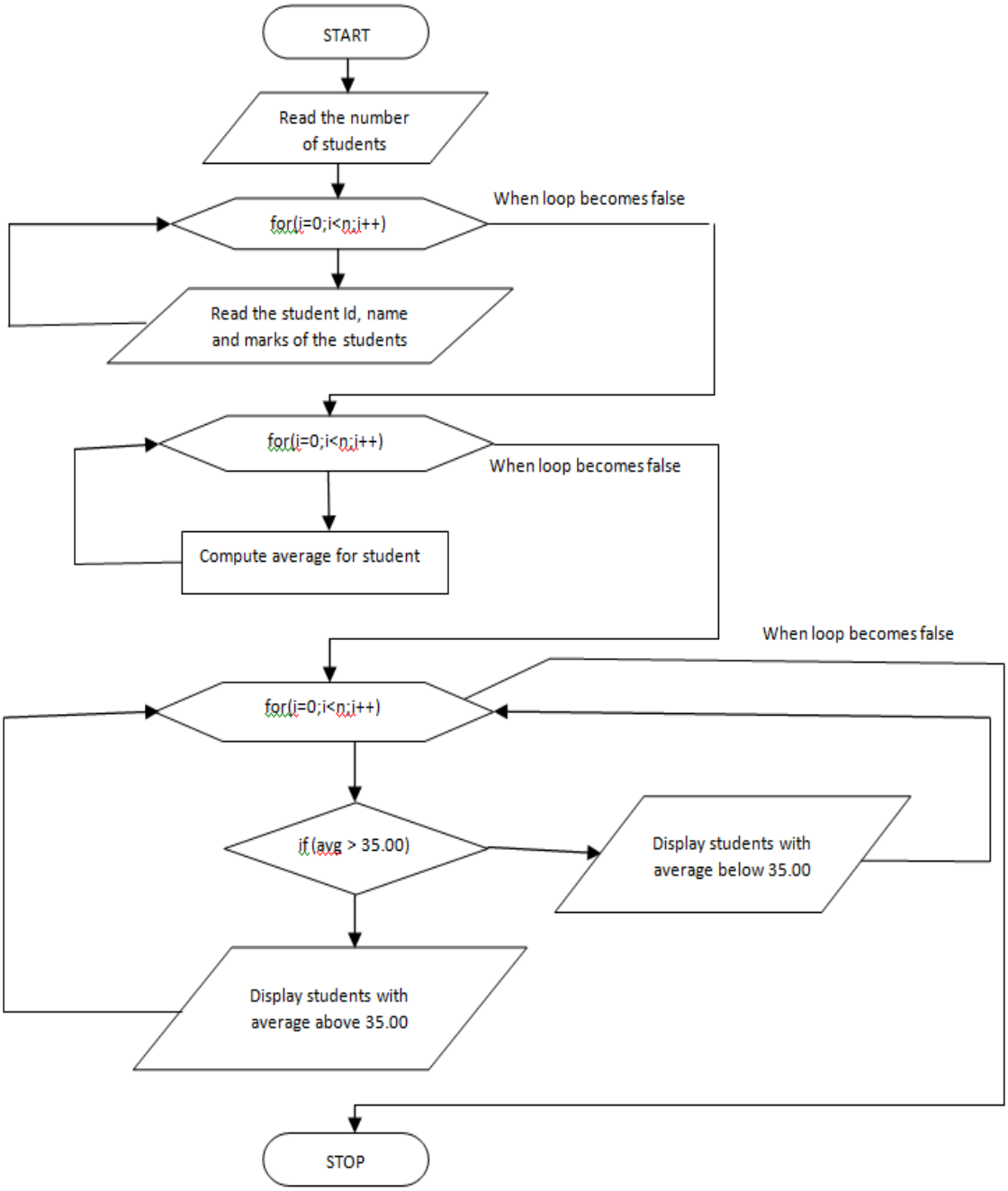
ALGORITHM

Input: Student details such as student id, name and marks

Output: To print the details of those students scoring above and below average

1. START
2. Read the number of students
3. For each student, read the student id, name and marks for all subjects.
4. Calculate the average of the marks and store in the avg field.
5. Print results.
6. Initialise loop
7. Read the average of each student
8. Check if avg>35.00
9. If yes print result else go to next iteration
10. Initialise loop
11. Read average of each student
12. Check if avg<35.00
13. If yes print result else go to next iteration
14. STOP

FLOWCHART


START
Read the number of students
for(i=0;i<n;i++)
When loop becomes false
Read the student Id, name and marks of the students
for(i=0;i<n;i++)
When loop becomes false
Compute average for student
When loop becomes false
for(i=0;i<n;i++)
if (avg > 35.00)
Display students with average below 35.00
Display students with average above 35.00
STOP

PROGRAM CODE:

```
#include<stdio.h>

 int main()
 {
        struct student
        {
                int stu_id;
                char name[20];
                float lang1_marks;
                float lang2_marks;
                float sc_marks;
                float mat_marks;
                float sst_marks;
                float comp_marks;
                float avg;
        };

        struct student s[20];
        int i,n;

        // Accept the number of records/students
        printf("Enter the number of records :");
        scanf("%d",&n);

        // Accept data for all the fields/members of each record
        printf("Enter %d student details...\n",n);

        for(i=0;i<n;i++)
        {
                printf("\n\nEnter student ID :");          // Student ID
                scanf("%d",&s[i].stu_id);

                printf("Enter student name :");          // Student Name
                scanf("%s",s[i].name);
```

```
		printf("Enter lang1 Marks:");		// Language 1 marks
		scanf("%f",&s[i].lang1_marks);

		printf("Enter lang2 Marks :");		// Language 2 Marks
		scanf("%f",&s[i].lang2_marks);

		printf("Enter science Marks :");	// Science Marks
		scanf("%f",&s[i].sc_marks);

		printf("Enter Maths Marks :");		// Maths Marks
		scanf("%f",&s[i].mat_marks);

		printf("Enter SST Marks :");		// SST Marks
		scanf("%f",&s[i].sst_marks);

		printf("Enter Computer Marks :");	// Computer Marks
		scanf("%f",&s[i].comp_marks);
	}

	// Compute the average of each student
	for(i=0;i<n;i++)
	{

		s[i].avg=(s[i].lang1_marks + s[i].lang2_marks + s[i].sc_marks + s[i].mat_marks +
s[i].sst_marks + s[i].comp_marks)/6.0;
	}

	// Display student ID, name and average of all students
	// who have scored above average marks
	printf("Students scoring above the average marks....\n");
	printf("\n\nID\tName\tAverage\n\n");

	for(i=0;i<n;i++)
	{
		if(s[i].avg>=35.0)
		printf("%d\t%s\t%f\n",s[i].stu_id,s[i].name,s[i].avg);
	}

	// Display student ID, name and average of all students
	// who have scored below average marks
```

```
        printf("\n\nStudents scoring below the average marks....\n");
        printf("\n\nID\tName\tAverage\n\n");

        for(i=0;i<n;i++)
        {
                if(s[i].avg<35.0)
                printf("%d\t%s\t%f\n",s[i].stu_id,s[i].name,s[i].avg);
        }

        return 0;
}
```

SAMPLE OUTPUT

./a.out

```
Enter the number of records :5
Enter 5 student details...

Enter student ID :101
Enter student name :manjula
Enter lang1 Marks:56
Enter lang2 Marks :54
Enter science Marks :34
Enter Maths Marks :32
Enter SST Marks :45
Enter Computer Marks :67


Enter student ID :102
Enter student name :kavitha
Enter lang1 Marks:90
Enter lang2 Marks :99
Enter science Marks :98
Enter Maths Marks :97
Enter SST Marks :96
Enter Computer Marks :100
```

Enter student ID :103
Enter student name :banu
Enter lang1 Marks:54
Enter lang2 Marks :22
Enter science Marks :1
Enter Maths Marks :12
Enter SST Marks :12
Enter Computer Marks :3

Enter student ID :104
Enter student name :pallavi
Enter lang1 Marks:3
Enter lang2 Marks :44
Enter science Marks :5
Enter Maths Marks :55
Enter SST Marks :53
Enter Computer Marks :21

Enter student ID :105
Enter student name :lalitha
Enter lang1 Marks:54
Enter lang2 Marks :2
Enter science Marks :34
Enter Maths Marks :56
Enter SST Marks :78
Enter Computer Marks :9
Students scoring above the average marks....

ID Name Average

101 manjula 48.000000
102 kavitha 96.666664
105 lalitha 38.833332

Students scoring below the average marks....

ID Name Average

103 banu 17.333334
104 pallavi 30.166666

QUESTIONS FOR VIVA

1. What is the size of a structure?
2. What is the difference between array and structures?
3. What is the syntax for creating a structure?
4. What is a structure?
5. How do you declare a structure variable?
6. How to access a structure variable?
7. What is typedef?

Prog 10: Develop a program using pointers to compute the sum, mean and standard deviation of all elements stored in an array of N real numbers.

AIM

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

PROBLEM STATEMENT

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

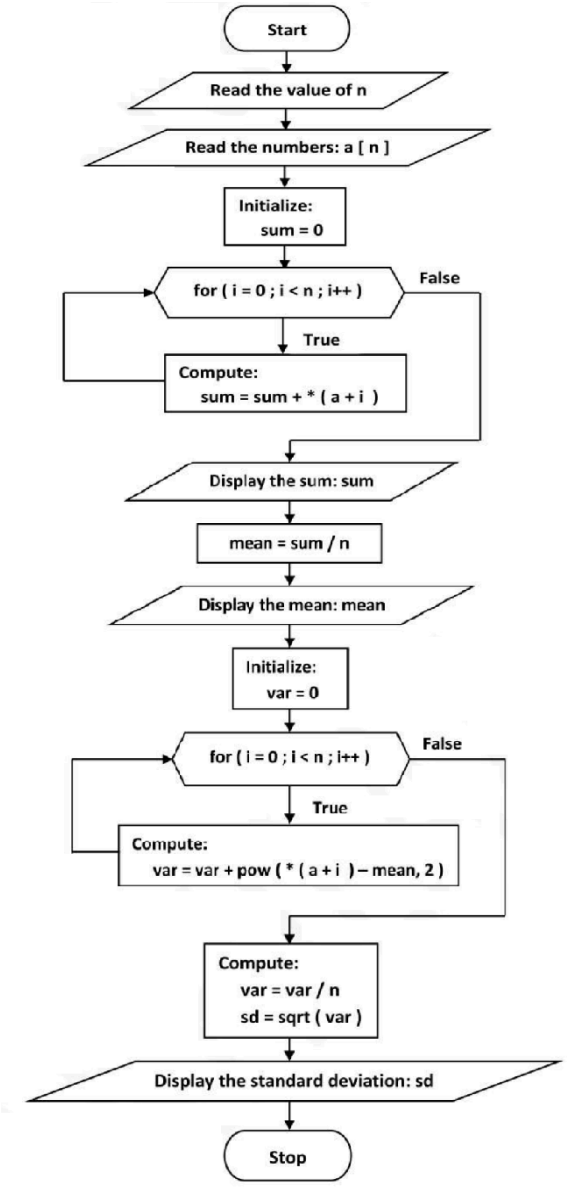
ALGORITHM

Input: An array of numbers

Output: To compute sum, mean and standard deviation

1. START
2. Read the value of n
3. Read the numbers into the array ‘a’ using the pointer ‘p’.
4. Initialise: i=0 and sum=0
5. Iterate the loop and perform the following
    a. Sum = sum + *p
    b. Increment i
6. Display the sum
7. Calculate the mean using the formula mean = sum/n
8. Display mean
9. Initialise i=0 and var=0
10. Iterate the loop and perform the following
    a. var=var+pow((*p-mean),2);
    b. Increment i
11. Display the variance
12. Compute
    a. var = var/n
    b. Sd = sqrt(var)
13. Display sd
14. STOP

FLOWCHART


Start
Read the value of n
Read the numbers: a [n]
Initialize:
sum = 0
for (i = 0 ; i < n ; i++)
False
True
Compute:
sum = sum + * (a + i)
Display the sum: sum
mean = sum / n
Display the mean: mean
Initialize:
var = 0
for (i = 0 ; i < n ; i++)
False
True
Compute:
var = var + pow (* (a + i) – mean, 2)
Compute:
var = var / n
sd = sqrt (var)
Display the standard deviation: sd
Stop

PROGRAM CODE:

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

int main()
{
        int i,n;
        float a[10],mean,sd,sum,var;
        float *p;                   // p is a pointer to float value

        printf("\n Enter Number of elements :");
        scanf("%d",&n);
        printf("\n Enter the elements :");




        p=a;                        // pointer p points to first element of a
        for(i=0;i<n;i++)
        {
                scanf("%f",p);
                p++;                // pointer p points to the next element of the array
        }

        p=a;                        // Initialize p to the first element of the array
        printf("\n input Elements are:\n");
        for(i=0;i<n;i++)
        {
                printf("%f",*p);
                p++;                // Pointer p is made to point to the next element
        }

        p=a;                        // Initialize p to the first element of the array

        sum=sd=mean=var=0;

        // Find the sum of the array elements
        for(i=0;i<n;i++)
        {
```

```
                sum=sum+(*p);
                p++;
        }
        // Find the mean
        mean=sum/n;

        // Find variance
        p=a;
        for(i=0;i<n;i++)
        {
                var=var+pow((*p-mean),2);
                p++;
        }
        var=var/n;

        // Find Standard Deviation
        sd=sqrt(var);

        // Print Sum, mean and Standard Deviation
        printf("\n\n mean=%f\nsum=%f\nsd=%f\nvar=%f\n",mean,sum,sd,var);
        return 0;
}
```

SAMPLE OUTPUT

1.

```
$ cc prog10.c -lm
$ ./a.out

 Enter Number of elements :5

 Enter the elements :1
2
3
4
5

The Sum=15.000000

The mean=3.000000
```

The Standard Deviation=1.414214

2.
$ ./a.out

Enter Number of elements :3

Enter the elements :55
88
22

The Sum=165.000000

The mean=55.000000

The Standard Deviation=26.944387

QUESTIONS FOR VIVA

1. What is a pointer?
2. How to declare a pointer variable?
3. What is dangling pointer?
4. What is a pointer to pointer?
5. What is pointer to array?
6. What is pass by reference?

Prog 11: Implement Recursive functions for Binary to Decimal Conversion

AIM: Implement Recursive functions for Binary to Decimal Conversion.

PROBLEM STATEMENT: Write a C program to implement Recursive functions for Binary to Decimal Conversion.

PROGRAM CODE:

```
#include<stdio.h>

int btod(int);          // Prototype of btod() function

int main()
{
        int binary,decimal;

        // Accept input in binary format
        printf("Enter binary input :");
        scanf("%d",&binary);

        // Invoke btod() to convert binary to decimal
        decimal=btod(binary);

        // Print the decimal equivalent of binary
        printf("Decimal equivalent = %d\n", decimal);

        return 0;
}

// Function Definition of btod() function
int btod(int bin)
{
        if (bin==0)
        {
```

```
                return 0;
        }
        else
        {
                return (bin%10 + btod(bin/10) * 2);
        }
}
```

SAMPLE OUTPUT:

```
$ cc prog11.c
$ ./a.out
Enter binary input :01
Decimal equivalent = 1

$ ./a.out
Enter binary input :100
Decimal equivalent = 4

$ ./a.out
Enter binary input :1001
Decimal equivalent = 9

$ ./a.out
Enter binary input :1101
Decimal equivalent = 13

$ ./a.out
Enter binary input :110011001
Decimal equivalent = 409
*/
```

04: Write a C Program to display the following by reading the number of rows as input,

```
        1
      1 2  1
    1 2 3 2 1
  1 2 3 4 3 2 1
nth row
```

Aim:

To develop a C Program to display the pattern by reading the number of rows as input,

```
        1
      1 2  1
    1 2 3 2 1
  1 2 3 4 3 2 1
nth row
```

Algorithm:

Step 1: Start

- Begin the execution of the program.

Step 2: Declare Variables

- Declare three integer variables i, j, and n:
  - i will be used as the row counter.
  - j will be used as the column counter.
  - n will store the number of rows entered by the user.

Step 3: Input the Number of Rows

- Print the message: "Input number of rows : ".
- Read the user's input for n using scanf("%d", &n);.

Step 4: Outer Loop (for rows)

- Create a for loop for i starting from 0 up to n. This loop will iterate n+1 times, where i represents the current row.

Step 5: Inner Loop 1 (Leading Spaces)

- Inside the outer loop, create another loop to print spaces. The loop runs from j = 1 to n - i. This loop prints spaces in decreasing order as the row number increases.

Step 6: Inner Loop 2 (Ascending Numbers)

- After printing the leading spaces, create another loop to print ascending numbers. The loop runs from j = 1 to i, printing numbers from 1 up to i.

Step 7: Inner Loop 3 (Descending Numbers)

- After the ascending numbers, create a loop that prints descending numbers. The loop runs from j = i - 1 down to 1, printing numbers from i-1 to 1.

Step 8: Newline Character

- After printing spaces and numbers for the current row, print a newline character (\n) to move to the next row.

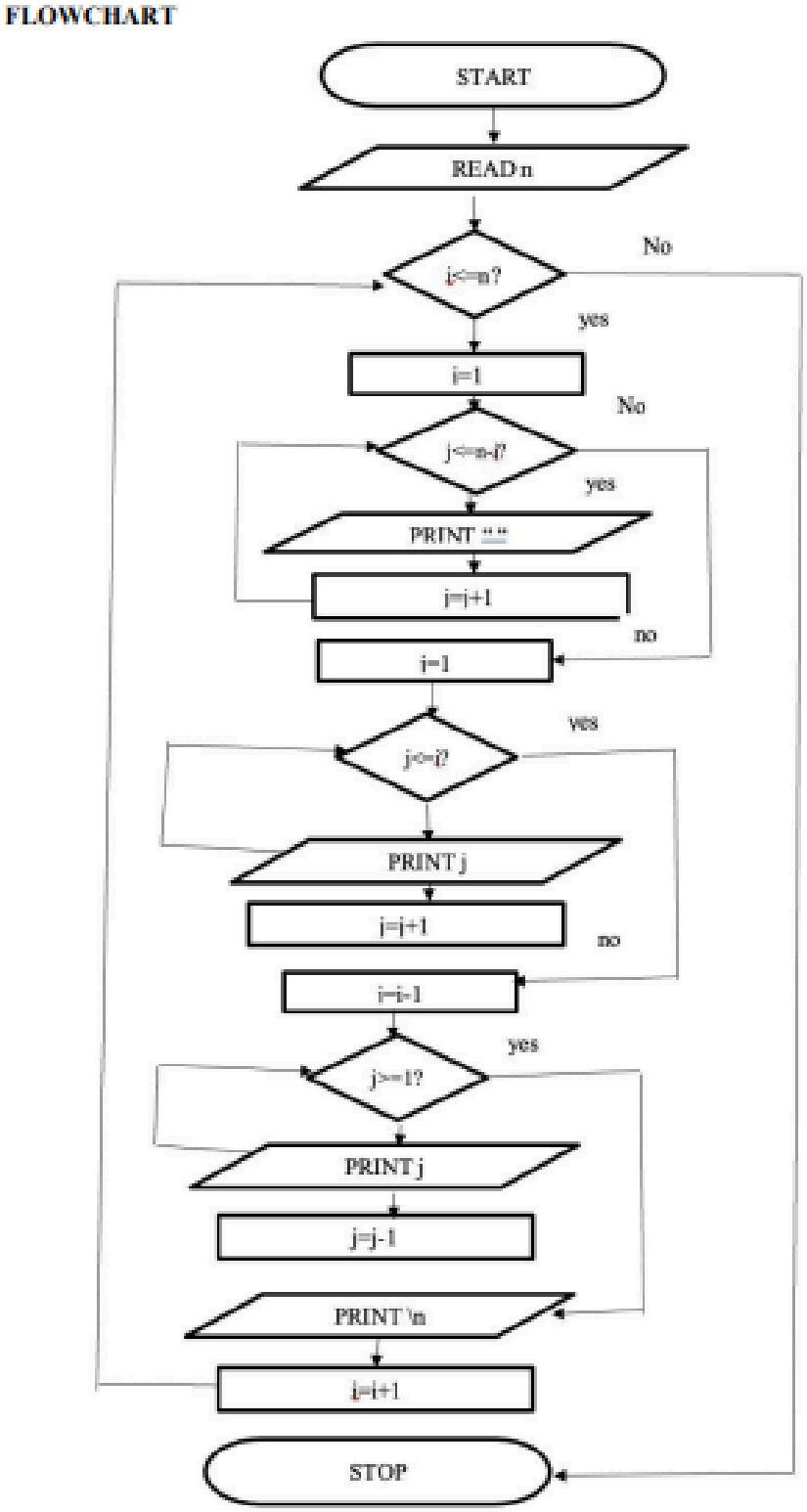
Step 9: Repeat

- Repeat steps 5 to 8 for the next rows by continuing the loop for i until all rows are printed.

Step 10: End

- Once the outer loop is complete, the program ends.

Flowchart:


FLOWCHART
START
READ n
i<=n?
No
yes
i=1
No
j<=n-i?
yes
PRINT " "
j=j+1
no
i=1
yes
j<=i?
PRINT j
j=j+1
no
i=i-1
yes
j>=1?
PRINT j
j=j-1
PRINT '\n
i=i+1
STOP

Program Code:

```
#include <stdio.h>
void main()
 { int i,j,n;
printf("Input number of rows : ");
 scanf("%d",&n);
for(i=1; i<=n;i++)
{
for(j=1;j<=n-i;j++)
{
printf(" ");
 }
for(j=1;j<=i;j++)
{
printf("%d",j);
} for(j=i-1;j>=1;j--)
{
printf("%d",j);
} printf("\n");
}
}
```

OUTPUT:

```
STDIN
5
```

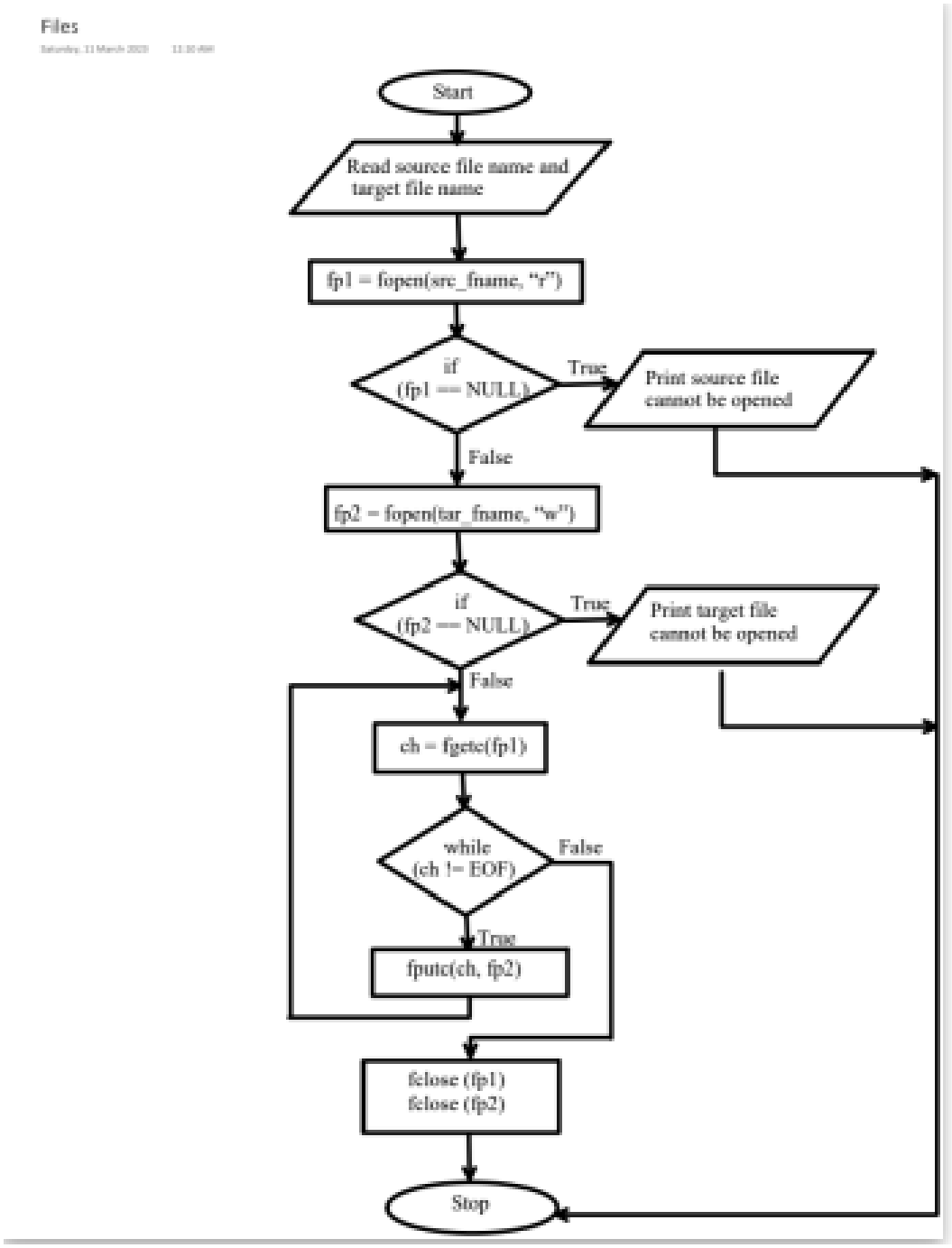
```
Output:

enter number of terms
          1
        1 2 1
      1 2 3 2 1
    1 2 3 4 3 2 1
  1 2 3 4 5 4 3 2 1
```

12: Write a C program to copy a text file to another, read both the input file name and target file name.

AIM:
To develop a c program to copy a text file to another, read both the input filename and target filename

Algorithm:
Input: Source file name and Target file name
Output: Contents of the source file is copied to the target file
Step 1: Start
Step 2: Read the source file name and target file name
Step 3: Open the source file in read mode
Step 4: if fp1 == NULL go to step 11, else go to step 5
Step 5: Open the target file in write mode
Step 6: if fp2 == NULL go to step 11, else go to step 7
Step 7: ch = fgetc(fp1)
Step 8: while ch != EOF (End of File) go to step 9, else go to step 11
Step 9: write the character ch to target file - fputc(ch, fp2), go to step 7
Step 10: Close the source file and target file
Step 11: Stop

Flowchart:


Files
Start
Read source file name and target file name
fp1 = fopen(src_fname, "r")
if (fp1 == NULL)
True
Print source file cannot be opened
False
fp2 = fopen(tar_fname, "w")
if (fp2 == NULL)
True
Print target file cannot be opened
False
ch = fgetc(fp1)
while (ch != EOF)
False
True
fputc(ch, fp2)
fclose (fp1)
fclose (fp2)
Stop

Program:

```
#include<stdio.h>
int main()
{
        char src_fname[20], tar_fname[20], ch;
        printf("Enter input file name and target file name :");
        scanf("%s%s", src_fname, tar_fname);
        FILE *fp1,*fp2;
        fp1 = fopen(src_fname, "r");
        if (fp1 == NULL)
        {
                printf("Unable to open file - %s in Read mode\n",src_fname);
                return 1;
        }
        fp2 = fopen(tar_fname, "w");
        if (fp2 == NULL)
        {
                printf("Unable to open file - %s in write mode\n", tar_fname);
                return 2;
        }
        while ((ch = fgetc(fp1)) != EOF)
        {
                        fputc(ch, fp2);
        }
        printf("File copied successfully\n");
        fclose(fp1);
        fclose(fp2);
}
```

Output

Prerequisite: create two text
files 1.txt Hello world
2.txt----?
Enter the filename to open for
reading 1.txt
Enter the filename to open for
writing 2.txt
File copied successfully

Content Beyond Syllabus Programs

1) Write a C program to print prime numbers within a range using functions.

```
#include <stdio.h>
int checkPrimeNumber(int n);
int main() {

  int n1, n2, i, flag;

  printf("Enter lower value and upper value: ");
  scanf("%d %d", &n1, &n2);

  // swap n1 and n2 if n1 > n2
  if (n1 > n2) {
    n1 = n1 + n2;
    n2 = n1 - n2;
    n1 = n1 - n2;
  }
  printf("Prime numbers between %d and %d are: ", n1, n2);
  for (i = n1 + 1; i < n2; ++i) {

    // flag will be equal to 1 if i is prime
    flag = checkPrimeNumber(i);

    if (flag == 1) {
      printf("%d ", i);
    }
  }
   return 0;
}

// user-defined function to check prime number
int checkPrimeNumber(int n) {
  int j, flag = 1;

  for (j = 2; j <= n / 2; ++j) {

    if (n % j == 0) {
      flag = 0;
```

```
      break;
    }
  }
  return flag;
}
```

Sample Output:

```
Enter the lower value and upper value : 2
10
The Prime numbers from 2 and 10 are:
3, 5, 7
```

2) Write a C program to find GCD and LCM using recursion.

```
#include <stdio.h>
int gcd(int x, int y);  //function prototype

int main()
{
  int num1, num2, hcf, lcm;

  printf("Enter two integer Values:\n");
  scanf("%d %d", &num1, &num2);

  hcf = gcd(num1, num2);
  printf("GCD: %d", hcf);
  printf("\nLCM: %d", (num1 * num2) / hcf);
  return 0;
}
//recursive function
int gcd(int x, int y)
{
  if (y == 0)   //recursion termination condition
  {
    return x;
  }
  else
  {
    return gcd(y, x % y);   //calls itself
  }
```

}

Sample Output:

Enter two integer values:
10
25
GCD:5
LCM:50
LCM of 24 and 36 = 72

Viva Questions

1. What is an algorithm?
2. What is flowchart?
3. Explain the function of switch statement?
4. What is a data type?
5. What is a variable?
6. Difference between case and default?
7. What are input and output operations?
8. Explain the purpose of header files.
9. What is printf statement?
10. What is scanf statement?
11. Explain the different cases in quadratic equation program.
12. What is clrscr ( )?
13. What is "\n" (slash n)?
14. What is math.h ?
15. What are decision making conditions of C language?
16. Write the syntax of " if "statement?
17. Difference between if and switch statements?
18. Write an unconditional control statement in C.
19. Write a flowchart for ' if' conditional construct?
20. Difference between "=" & "= =".
21. What is a keyword?
22. Explain the syntax of "scanf".
23. What is modulus (%) operation?
24. Explain the syntax of if-else statement.
25. What are Looping control statements?
26. Explain while loop.
27. What is the difference between while and for loops?
28. What are the Entry controlled and Exit controlled loops in C ?
29. Write the flowchart for 'while' loop.
30. Write the flowchart for 'do while' loop

31. Explain the function of if-else ladder.
32. Explain simple if condition.
33. Explain how the binary search program works.
34. Which are the different types of searching technique?
35. What is the difference between linear search and binary search?
36. What are advantages and disadvantages of binary search?
37. Why are arrays needed?
38. Why is the return type used?
39. What is function prototype?
40. What is the default value returned by a function.
41. What is RAM?
42. What is ROM?
43. What is PROM?
44. What is EPROM?
45. What is EEPROM?
46. Differentiate between Volatile and non-volatile memory.
47. What are the input and output devices?
48. What is the need for cache memory?
49. Differentiate between main memory and secondary memory.
50. What is hardware?
51. Mention the different types of hardware components?
52. What are the different types of keyboards?
53. Differentiate between serial and parallel keyboard.
54. How do you classify printers?
55. Differentiate between serial and parallel printer.
56. Differentiate between impact and non-impact printers.
57. Differentiate between line and page printers.
58. Which printer do you select when high quality output is to be produced?
59. What is meant by softcopy output?
60. What is meant by hardcopy output?
61. What is printer buffer?
62. What is meant by tracks?
63. What is meant by sectors?
64. What is CD-ROM?
65. What are the different types of mouse?
66. What are the main parts of a floppy disk?
67. Differentiate between magnetic disks and optical disks
68. What is programming?
69. What is software?
70. What are the different types of software?
71. What are the different types of programming languages?
72. Differentiate between interpreter and compiler.
73. Differentiate between loader and linker.
74. Differentiate between Application software and System software.

75. What is an operating system?
76. What are language processors?
77. Mention some operating systems.
78. Differentiate between compiler and assembler.
79. What is translator?
80. What is meant by interpretation?
81. What is meant by source program?
82. What is meant by object program?
83. Give examples for High Level Languages.
84. Give examples for Assembly Level Languages.
85. Give examples for General purpose HLL.
86. Give examples for Specific purpose HLL.
87. Differentiate between Analog and Digital computers.
88. Differentiate between microcomputer & minicomputer.
89. Differentiate between internal & external DOS commands.
90. How do you classify the computers based on the size & capability?
91. How do you classify the computers based on principle of working?
92. What is computer network?
93. How do you classify the computer networks?
94. What is Batch processing?
95. What is Time Sharing?
96. Differentiate between LAN and WAN.
97. What are the different types of DOS?
98. What is DOS?
99. What is Real Time Systems?
100. What are the advantages of computer network?
101. What is BCPL?
102. Who developed BCPL?
103. Who developed C language?
104. How do you make comments in C program?
105. How the name C is derived?
106. What is K & R C?
107. What is preprocessor statement?
108. Differentiate between constant and variable.
109. What is data type?
110. Name the basic data types of C.
111. Describe at least five different format specifiers
112. Differentiate between string constant and character constant.
113. What are signed values?
114. What is the range of integer, char, float for a 16-bit computer?
115. What is a statement?
116. What is a keyword?
117. Differentiate between keywords and identifiers.
118. What is the need for an escape sequences?

119. Define and explain scanf () function?
120. What are the maximum and minimum possible ranges of values for long andshort type?
121. What exactly is a variable scope, local variable and a global variable?
122. What is a symbolic constant?
123. What are the rules for constructing real constants?
124. How do you classify C operators?
125. What is the use of modulus operator?
126. What is meant by mixed mode operation?
127. What are bitwise operators?
128. What is unary operator?
129. What is binary operator
130. What is operator precedence?
131. What is typecasting?
132. What is enum used for and state its format?
133. What is a conditional / ternary operator?
134. What is need for type conversion?
135. Differentiate between pre-increment/decrement & post-increment/decrement.
136. Differentiate between Unformatted and formatted I/O statements.
137. How do you classify the control statements?
138. Differentiate between while and do-while loop.
139. Differentiate between break and continue.
140. Is it possible to write a c program without semicolons?
141. When do you prefer forloop statement?
142. What is looping?
143. What is an array?
144. Give the classification of arrays?
145. State some significant uses of arrays in C programming?
146. Differentiate between anarray and an ordinary variable.
147. Array variable is also called as .
148. What are character arrays?
149.What is the difference between character array and string in C? 150.When do you usetwo-dimensional character array?
151. Name the different string handling functions?
152. What is meant by modularization?
153. Differentiate between standard functions & user-defined functions?
154. Differentiate between arguments and parameters.
155. Differentiate between local and global variables.
156. Name the different methods of parameter passing?
157. How does the function definition differ from function declaration?
158. What isrecursive function?

****