COMPUTER PROGRAMMING LABORATORY

18CPL27

ATRIA INSTITUTE OF TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
BANGALORE-560024
2020

COMPUTER PROGRAMMING LABORATORY
SEMESTER – I/II

Subject Code	**18CPL17/27**	**CIE Marks**	**40**
Number of Contact Hours/Week	**3hrs**	**SEE Marks**	**60**
Total Number of Lab Contact Hours	**30**	**Exam Hours**	**3 Hrs**

Credits – 1

Descriptions (if any): The laboratory should be preceded or followed by a tutorial to explain the approach or algorithm being implemented / 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.

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

PART A

2. Develop a program to solve simple computational problems using arithmetic expressions and use of each operator leading to simulation of a commercial calculator. (No built-in math function)
3. Develop a program to compute the roots of a quadratic equation by accepting the coefficients. Print appropriate messages.
4. Develop a program to find the reverse of a positive integer and check for palindrome or not. Display appropriate messages
5. An electricity board charges the following rates for the use of electricity: for the first 200 units 80 paisa per unit: for the next 100 units 90 paisa 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 charge.
6. Introduce 1D Array manipulation and implement Binary search.
7. Implement using functions to check whether the given number is prime and display appropriate messages. (No built-in math function)

PART B

8. Develop a program to introduce 2D Array manipulation and implement Matrix multiplication and ensure the rules of multiplication are checked.
9. Develop a Program to compute Sin(x) using Taylor series approximation .Compare your result with the built- in Library function. Print both the results with appropriate messages.

10. Write functions to implement string operations such as compare, concatenate, string length. Convince the parameter passing techniques.
11. Develop a program to sort the given set of N numbers using Bubble sort.
12. Develop a program to find the square root of a given number N and execute for all possible inputs with appropriate messages. Note: Don't use library function sqrt(n).
13. Implement structures to read, write, compute average- marks and the students scoring above and below the average marks for a class of N students.
14. Develop a program using pointers to compute the sum, mean and standard deviation of all elements stored in an array of n real numbers.
15. Implement Recursive functions for Binary to Decimal Conversion.

Laboratory Outcomes: The student should be able to: Write algorithms, flowcharts and program for simple problems. Correct syntax and logical errors to execute a program. Write iterative and wherever possible recursive programs. Demonstrate use of functions, arrays, strings, structures and pointers in problem solving.

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

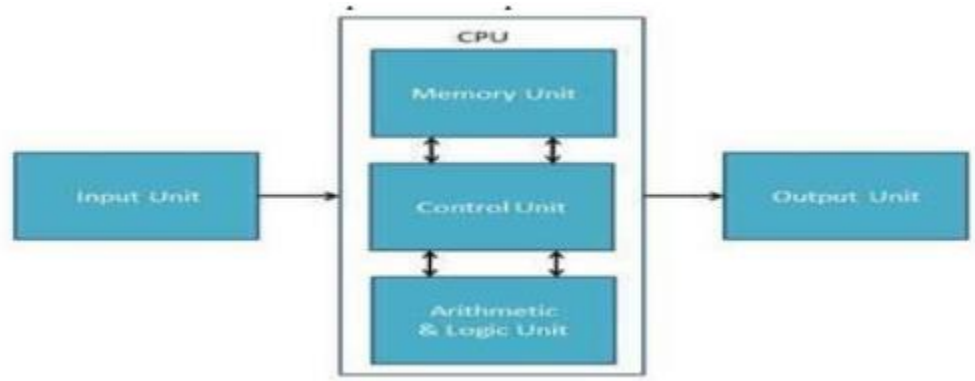
i. **Part A** – Procedure + Execution + Viva = 4 + 21 + 5 = 30 Marks

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

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

Computer is an advanced electronic device that takes raw data as an input from the user and processes it under the control of a set of instructions (called program), produces a result (output), and saves it for future use.

S.No.	Operation	Description
1	Take Input	The process of entering data and instructions into the computer system.
2	Store Data	Saving data and instructions so that they are available for processing as and when required.
3	Processing Data	Performing arithmetic, and logical operations on data in order to convert them into useful information.
4	Output Information	The process of producing useful information or results for the user, such as a printed report or visual display.
5	Control the workflow	Directs the manner and sequence in which all of the above operations are performe



Input Unit

This unit contains devices with the help of which we enter data into the computer. This unit creates a link between the user and the computer. The input devices translate the information into a form understandable by the computer.

CPU (Central Processing Unit)

CPU is considered as the brain of the computer. CPU performs all types of data processing operations. It stores data, intermediate results, and instructions (program). It controls the operation of all parts of the computer.

CPU itself has the following three components −

- ALU (Arithmetic Logic Unit)

- Memory Unit
- Control Unit
- Memory or Storage Unit

This unit can store instructions, data, and intermediate results. This unit supplies information to other units of the computer when needed. It is also known as internal storage unit or the main memory or primary storage or Random Access Memory (RAM).

Its size affects speed, power, and capability. Primary memory and secondary memory are two types of memory in the computer. Functions of the memory unit are –

o It stores all the data and the instructions required for processing.
o It stores intermediate results of processing.
o It stores the final results of processing before these results are released to an output device.
o All inputs and outputs are transmitted through the main memory.

Control Unit

This unit controls the operations of all parts of the computer but does not carry out any actual data processing operations.

Functions of this unit are –

o It is responsible for controlling the transfer of data and instructions among other units of a computer.
o It manages and coordinates all the units of the computer.
o It obtains the instructions from the memory, interprets them, and directs the operation of the computer.
o It communicates with Input/Output devices for transfer of data or results from storage.
o It does not process or store data.

ALU (Arithmetic Logic Unit) This unit consists of two subsections namely,

- **Arithmetic Section**
- **Logic Section**

- Arithmetic Section
  Function of arithmetic section is to perform arithmetic operations like addition, subtraction, multiplication, and division. All complex operations are done by making repetitive use of the above operations.

- Logic Section
  Function of logic section is to perform logic operations such as comparing, selecting, matching, and merging of data.

Output Unit The output unit consists of devices with the help of which we get the information from the computer. This unit is a link between the computer and the users. Output devices translate the computer's output into a form understandable by the users.

Following are some of the important input devices which are used in a computer – Keyboard Mouse Joy Stick Light pen Track Ball Scanner Graphic Tablet Microphone Magnetic Ink Card Reader(MICR) Optical Character Reader(OCR) Bar Code Reader Optical Mark Reader(OMR)

Following are some of the important output devices used in a computer Monitors Graphic Plotter

Printer Programming Environment
If we want to set up environment for the C programming language, we need the following two software tools available on computer, (a) Text Editor and (b) The C Compiler.

An **integrated development environment (IDE**) is a software application that provides comprehensive facilities to computer programmers for software development. An IDE normally consists of a source code editor, build automation tools, and a debugger. Most modern IDEs have intelligent code completion. Some IDEs, such as NetBeans and Eclipse, contain a compiler, interpreter, or both;

Naming the program files : A filename is a name used to uniquely identify a computer file stored in a file system. The filename can consist of letters, digits and special characters. The files that we create with editor are called the source files and they contain the program source codes. The source files for C programs are typically named with the extension ".c".

Syntax: filename.ext ic, Eg: hello.c

$ gedit hello.c

Storing, Compilation, Execution and Debugging : The program is created and named and stored in the disk. It can be referenced any time later by its filename.

Compilation is a process of converting source code in high level language to low level. $gcc hello.c

The Translation is done after examine each instruction for correctness. When there are no errors, the translated program is stored on another file with the name filename.o. This program is the object code. Eg- hello.o.

Execution is the process of loading the executable object code into computer memory and execute the instructions. $ ./a.out

Debugging is a systematic process of spotting and fixing the number of bugs, or defects, in a piece of software so that the software is behaving as expected. Let us look at a simple code that would print the words "Hello World" −

#include<stdio.h> int main() { /* my first program in C */ printf("Hello, World! \n"); return 0; }

Let us take a look at the various parts of the above program −
The first line of the program #include <stdio.h> is a preprocessor command, which tells a C compiler to include stdio.h file before going to actual compilation. The next line int main() is the main function where the program execution begins.

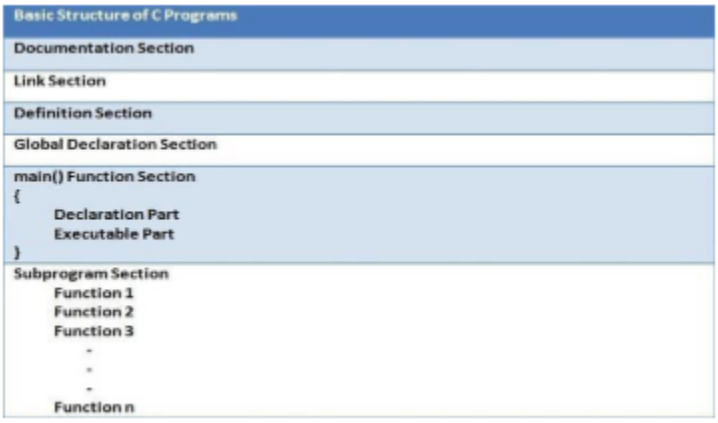
The next line /*...*/ will be ignored by the compiler and it has been put to add additional comments in the program. So such lines are called comments in the program.

The next line printf(...) is another function available in C which causes the message "Hello, World!" to be displayed on the screen.

The next line **return 0;** terminates the main() function and returns the value 0.

Compile and Execute C Program :Let us see how to save the source code in a file, and how to compile and run it.
Open a text editor and add the above-mentioned code.
Save the file as hello.c
Open a command prompt and go to the directory where you have saved the file.
Type gcc hello.c and press enter to compile your code
If there are no errors in your code, the command prompt will take you to the next line and would
generate a.out executable file.
Now, type a.out to execute your program. You will see the output "Hello World" printed on the screen.

```
$ gcc hello.c
$ ./a.out
Hello, World!
```



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)

Algorithm:

Step1: [Initialize]

start

Step2: [Input a simple arithmetic expression]

Read a,op,c

Step3: [Check for arithmetic operators using switch case with operator op]

switch(op)

3.1 case '+': res=a+b; goto step 4

3.2 case '-' : res=a-b; goto step 4

3.3 case '*': res=a*b; goto step 4

3.4 case '/':

if(b!=0)

res=a/b;

else

printf("division by zero is not possible");

goto step 5

end if

goto step 4

3.5 case '%': res=a%b; goto step 4

3.6 default: printf("illegal operator");

goto step 5

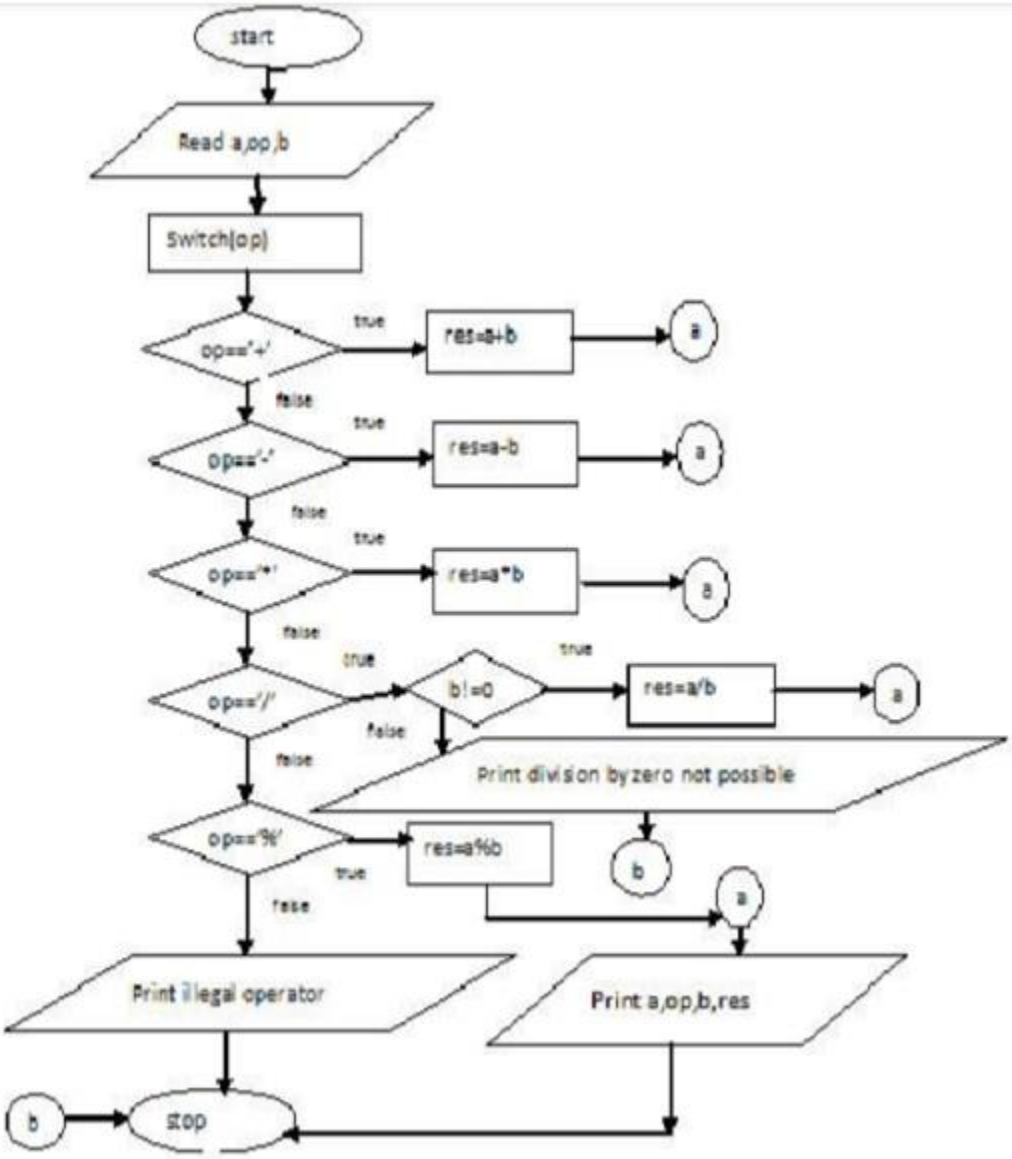
Step4:[Print the result res]

Print a,op,b,res

Step 5: [Finished]

Stop

Flowchart


start
Read a,op,b
Switch(op)
op=='+'
true
res=a+b
a
false
op=='-'
true
res=a-b
a
false
op=='*'
true
res=a*b
a
false
op=='/'
true
b!=0
true
res=a/b
a
false
Print division byzero not possible
b
false
op=='%'
true
res=a%b
a
false
Print illegal operator
Print a,op,b,res
b
stop

Program:

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
 int a, b, res;
char op;
printf("Enter a simple arithmetic expression:\n");
scanf("%d%c%d",&a,&op,&b);
switch(op)
{
case '+':res=a+b;
break;
case '-':res=a-b;
break; case '*':res=a*b;
break;
case '/':if(b!=0)
res=a/b;
else
{
printf("division by zero is not possible\n");
exit(0);
break;
case '%':res=a%b;
break;
default:printf("Illigal operator\n");
exit(0);
}
printf("\n %d%c%d=%d",a,op,b,res);
return 0;
}
```

Test Cases:

Test No	Input Parameters	Expected Output	Obtained Output
1	5+3	5+3=8	5+3=8
2	5-3	5-3=2	5-3=2
3	5*3	5*3=15	5*3=15
4	5/3	5/3=1	5/3=1
5	5%3	5%3=2	5%3=2
6	5/0	division by zero is not possible	division by zero is not possible
7	5_3	illegal operator	illegal operator

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

Algorithm:

Step1: [Initialize]

start

Step2: [Input the coefficients of quadratic equation]

Read a,b,c.

Step3: [Check for valid coefficients]

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

Then invalid coefficients. Go to step 6

Step4:[Compute discriminant value as d]

d=b*b-4*a*c

Step5: [Check for d value to compute roots]

5.1:if (d>0)

then

print real and distinct roots

rootl=(−b+√d)/(2*a)

root2=(−b−√d)/(2*a)

print root1,root2

go to step 6

5.2 :if (d==0) then

print real and equal roots root1 = root2= -b/(2*a)

print root1,root2

go to step 6

5.3:if (d<0) then

print real and imaginary real=-b/(2*a)

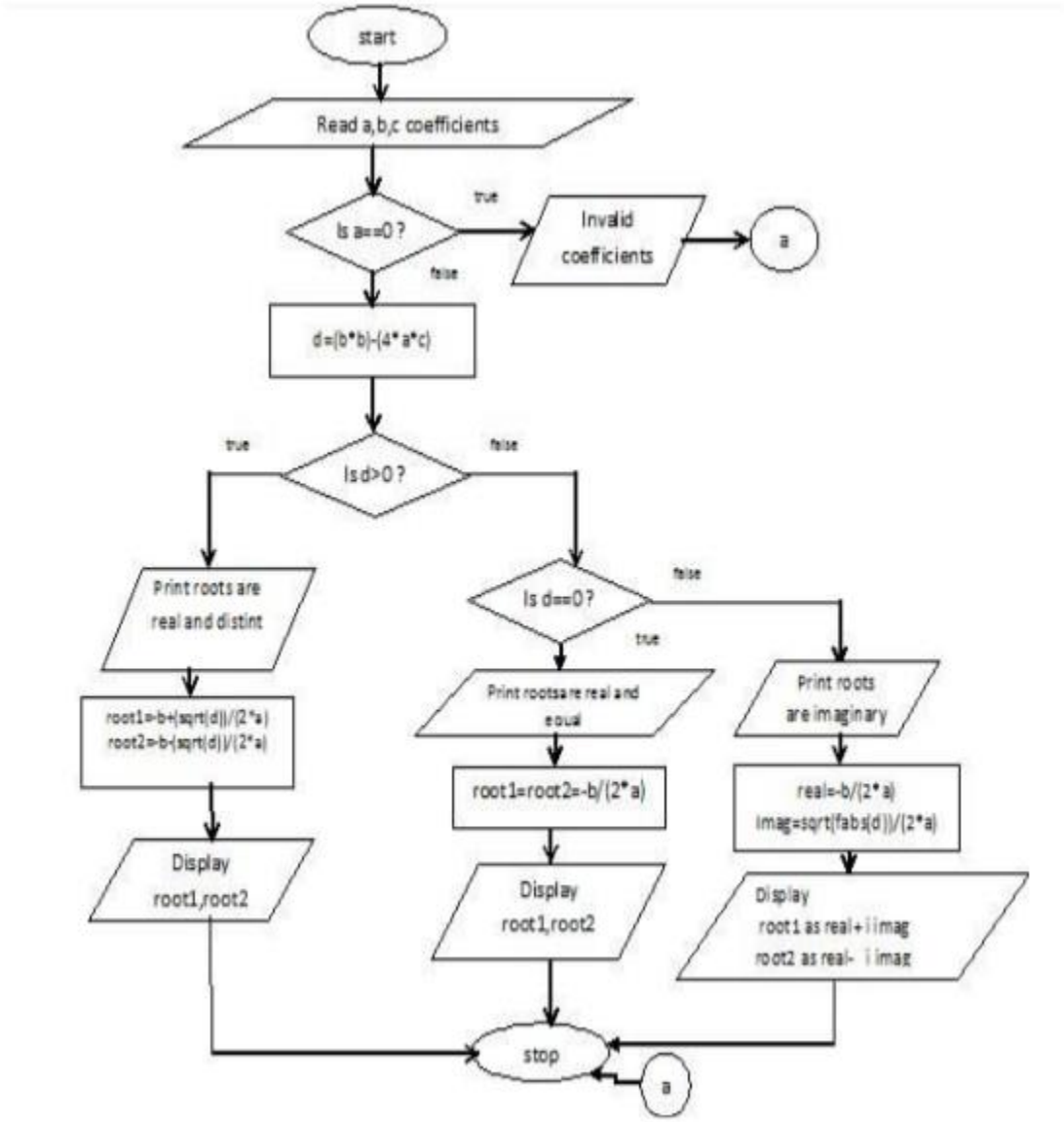
imag=(√fabs(d))/(2*a) print rootl as real+imag print root2 as real−Image

goto step 6

Step 6: [Finished]

Stop

Flowchart:


start
Read a,b,c coefficients
Is a==0 ?
true
Invalid coefficients
a
false
d=(b*b)-(4*a*c)
true
Is d>0 ?
false
Print roots are real and distint
root1=-b+(sqrt(d))/(2*a)
root2=-b-(sqrt(d))/(2*a)
Display root1,root2
Is d==0 ?
false
true
Print rootsare real and equal
root1=root2=-b/(2*a)
Display root1,root2
Print roots are imaginary
real=-b/(2*a)
Imag=sqrt(fabs(d))/(2*a)
Display root1 as real+iimag root2 as real- iimag
stop
a

Program:

```
 #include<stdio.h>
 #include<stdlib.h>
#include<math.h>
int main()
{
 float a, b, c, d,root1,root2,real,imag;
printf("enter the coefficients a, b, c:\n");
scanf("%f%f%f",&a,&b,&c);
if((a==0)||(b==0)||(c==0))
{ printf("Invalid coefficients\n");
 exit(0);
}
d=b*b-4*a*c;
 if(d>0)
{
 root1=(-b+sqrt(d))/(2.0*a);
 root2=(-b-sqrt(d))/(2.0*a);
printf("the roots are real and distinct...\n");
printf("root1=%.2f\nroot2=%.2f\n",root1,root2)
 }
 else if(d==0)
{
 root1=root2=b/(2.0*a);
printf("the roots are real and equal..\n");
printf("root1=%.2f\nroot2=%.2f\n",root1,root2);
 }
 Else
 {
 real=b/(2.0*a);
 imag=sqrt(fabs(d))/(2.0*a);
 printf("the roots are complex and imaginary..\n");
printf("root1=%.2f+i%.2f\nroot2=%.2f-i%.2f\n",real,imag,real,imag);
}
return 0;
}
```

Test case:

Test No	Input Parameters	Expected Output	Obtained Output
1	a=0, b=0, c=1	Invalid Coefficients	Invalid Coefficients
2	a=1, b=6, c=9	Roots are real and equal root1=-3, root2=-3	Roots are real and equal root1=-3, root2=-3
3	a=1, b=-5, c=3	Roots are real and distinct root1=4.30, root2=0.69	Roots are real and distinct root1=4.30, root2=0.69
4	a=1, b=4, c=7	Roots are complex and imaginary root1= -2+i1.73, root2= -2- i1.73	Roots are complex and imaginary root1= -2+i1.73, root2= -2- i1.73

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

Algorithm:

Step 1:[Initialize]

Start

Step 2: [Read num from the user]

Step 3: [Set number num to a variable n]

n=num

Step 4: [Iterate until n is not equal to 0]

If n value becomes 0, control comes out of the loop. So n's original value is lost. So, num value is stored in other variable n in step 3. In step 4 reverse of the number is calculated

While (n!=0)
rem=n%10
rev=rev*10+rem
n=n/10
End while

Step 5: [Print reverse number]

Print rev

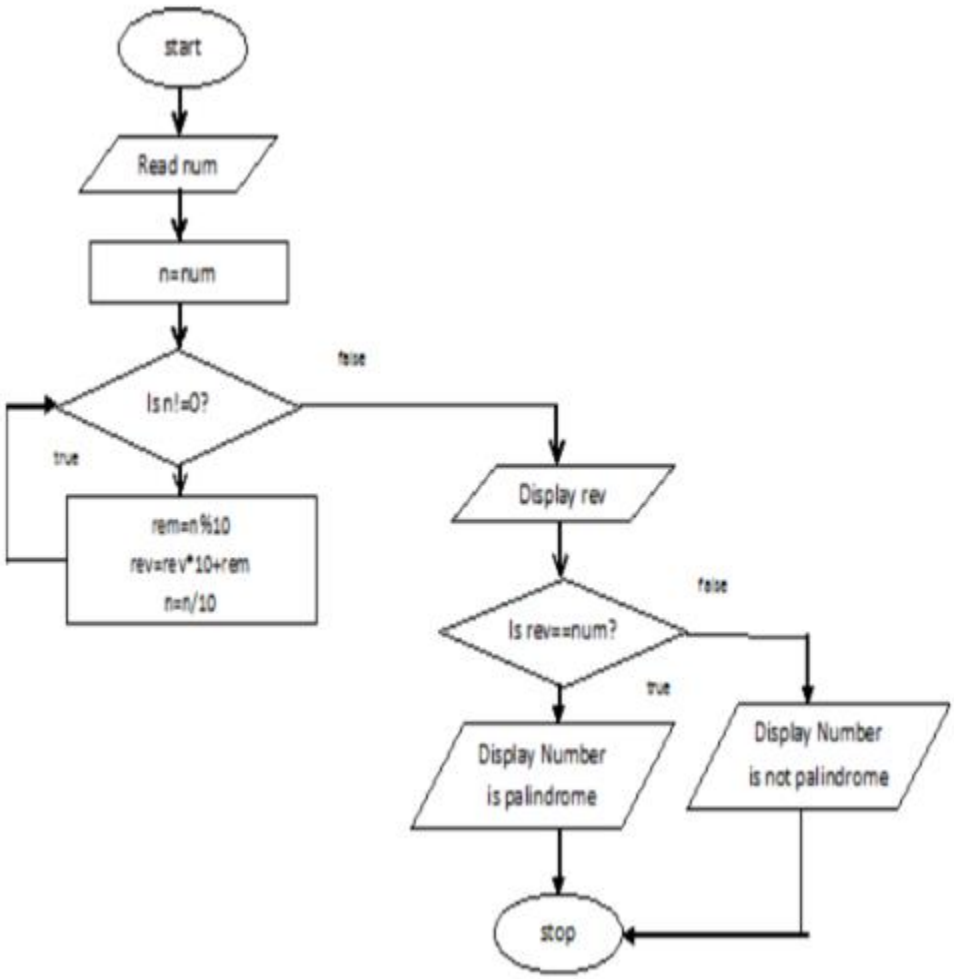
Step 6: [Check if original number and reverse number are the same. If it is, number is palindrome. otherwise, not palindrome]

if rev==num then
print "The number is palindrome"
else
print "The number is not palindrome"
end if

Step 7:[Finished]

Stop

Flowchart


start
Read num
n=num
Is n!=0?
false
true
rem=n%10
rev=rev*10+rem
n=n/10
Display rev
Is rev==num?
false
true
Display Number
is palindrome
Display Number
is not palindrome
stop

Program

```
#include<stdio.h>
int main()
{
int n, rev=0, rem=0,num;
printf("enter a number:\n");
scanf("%d",&num);
n = num;
while(n!=0)
{
rem=n%10;
rev=rev*10+rem;
 n=n/10;
}
printf("\n reversed num is %d",rev);
if(num==rev)
printf("\n %d is palindrome",num);
else
printf("\n%d is not palindrome",num);
return 0;
}
```

Test Cases

Test No	Input Parameters	Expected Output	Obtained Output
1.	N=1441	Reversed num is 1441 The number is palindrome	Reversed num is 1441 The number is palindrome
2	N=0	Reversed num is 0 The number is palindrome	Reversed num is 0 The number is palindrome
3	N=9987	Reversed num is 7899 The number is not palindrome	Reversed num is 7899 The number is not palindrome
4	N=-12221	Reversed num is -12221 The number is palindrome	Reversed num is -12221 The number is palindrome

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

Algorithm

Step 1: [Initialize]

Start

Step 2:[Read the name of the user and number of units of electricity consumed]

Read name,units

Step 3: [Use else if ladder, to satisfy the given condition: 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(units<=200)
rupees=units*0.80;
rupees=rupees+100;
else if(units<=300 && units>200)
rupees=200*0.80+(units-200)*0.90;
rupees=rupees+100;
else
rupees=200*0.80+100*0.90+(units-300)*1.00;
rupees=rupees+100;
end if
```

Step 4:[Use simple if to satisfy, the total amount is more than Rs 400, then an additional surcharge of 15% of the total amount is charged]

```
if(rupees>400)
rupees=rupees+0.15*rupees;
end if
```

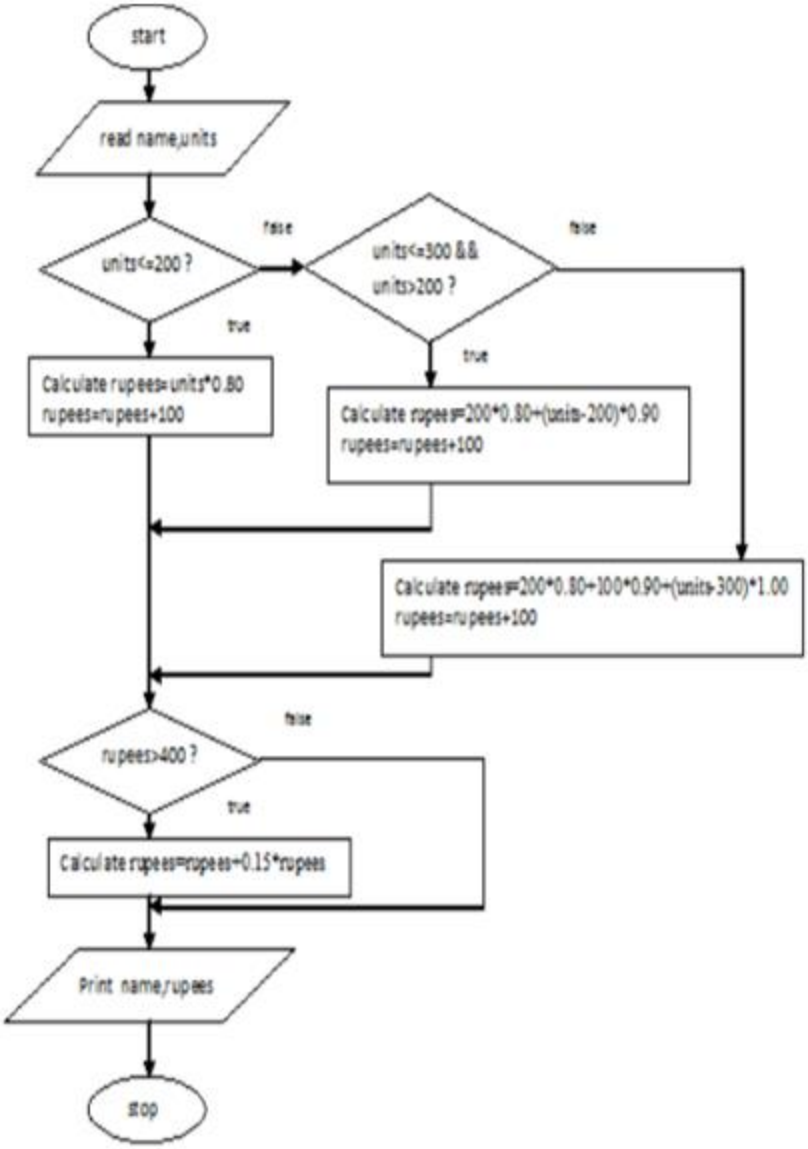
Step 5:[Print the name of user and amount to be paid]

Print name, rupees

Step 6:[Finished]

Stop

Flowchart:


start
read name,units
units<=200 ?
false
units<=300 &&
units>200 ?
false
true
true
Calculate rupees=units*0.80
rupees=rupees+100
Calculate rupees=200*0.80+(units-200)*0.90
rupees=rupees+100
Calculate rupees=200*0.80+100*0.90+(units-300)*1.00
rupees=rupees+100
rupees>400 ?
false
true
Calculate rupees=rupees+0.15*rupees
Print name,rupees
stop

Program

```
#include<stdio.h>
int main()
{
char name[20];
int units;
float rupees=0;
printf("\n enter the name of the user :");
gets(name);
printf("\n enter number of units consumed :");
scanf("%d",&units);
if(units<=200)
{
rupees=units*0.80;
rupees=rupees+100;
}
else if(units<=300 && units>200)
{
rupees=200*0.80+(units-200)*0.90;
rupees=rupees+100;
}
else
{
rupees=200*0.80+100*0.90+(units-300)*1.00;
rupees=rupees+100;
}
if(rupees>400)
rupees=rupees+0.15*rupees;
printf("%s has to pay rupees %f",name,rupees);
return 0;
}
```

Test cases:

Test No	Input Parameters	Expected Output	Obtained Output
1.	enter the name of the user :Sonia enter number of units consumed : 200	Sonia has to pay rupees 260	Sonia has to pay rupees 260
2	enter the name of the user :Sonia enter number of units consumed : 300	Sonia has to pay rupees 350	Sonia has to pay rupees 350
3	enter the name of the user :Sonia enter number of units consumed : 400	Sonia has to pay rupees 517.5	Sonia has to pay rupees 517.5

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

Algorithm

Step 1: [Initialize]

Start

Step 2:[Read number of numbers and the numbers:n and n numbers in ascending order in an array]

Read n, a[]

Step 3: [Read the number to be searched as key]

Read key

Step 4:[Set low and high]

low=0

high=n-1

Step 5:[Iterate using while loop until low<=high and !found]

while(low<=high && !found)

mid=(low+high)/2

if(a[mid]==key)

found=1

else if(a[mid]<key)

low=mid+1

else

high=mid-1

end if

end while

Step 6:[Finished]

Stop

Flowchart

Program:

```
#include<stdio.h>
#include<string.h>
int main()
{
int a[10],key;
int n,i,low,high,mid,found=0;
printf("enter the number of numbers to read\n");
scanf("%d",&n);
printf("enter the numbers in ascending order\n");
for(i=0;i<n;i++)
scanf("%d",&a[i]);
printf("enter the number to search\n");
scanf("%d",&key);
low=0;
high=n-1;
while(low<=high && !found)
{
mid=(low+high)/2;
if(a[mid]==key)
found=1;
else if(a[mid]<key)
low=mid+1;
else
high=mid-1;
}
if(found==1)
printf("\n number found at position:%d",mid+1);
else
printf("\n number not found");
return 0;
}
```

Test cases

Test No	Input Parameters	Expected Output	Obtained Output
1	enter the number of numbers to read 5 enter the numbers in ascending order 10 20 30 40 50 enter the number to search 50	number found at position 5	number found at position 5
2	enter the number of numbers to read 5 enter the numbers in ascending order 10 20 30 40 50 enter the number to search 5	number not found	number not found

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

Algorithm (calling function)

Step 1: [Initialize]
Start
Step 2: [Input a number]
Read num
Step 3: [Call the user defined function isprime, Use if-else to find prime or not]
if(isprime(num))
print the num is prime
else
print the num is not prime
end if
Step 6:[Finished]
Stop

Algorithm (called user defined function)

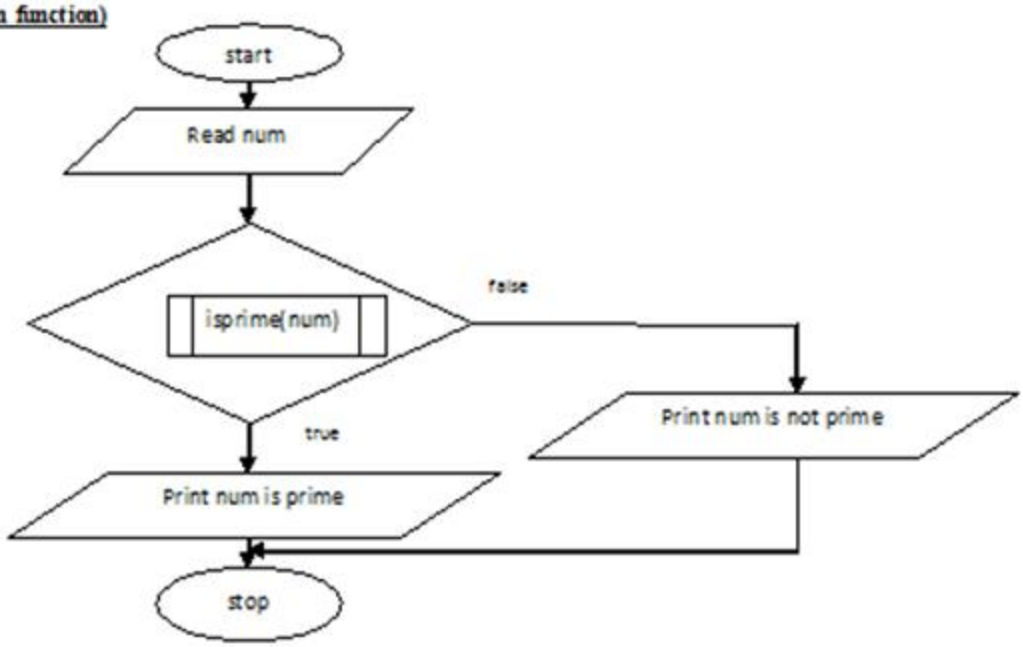
Step 1: [Initialize]
Start
Step 2: [check whether the number n passed from calling function is prime or not]

```
if(n==0||n==1)
        return 0;
for(i=2;i<=sqrt(n);i++)
{
        if(n%i==0)
        return 0;
}
return 1;
```

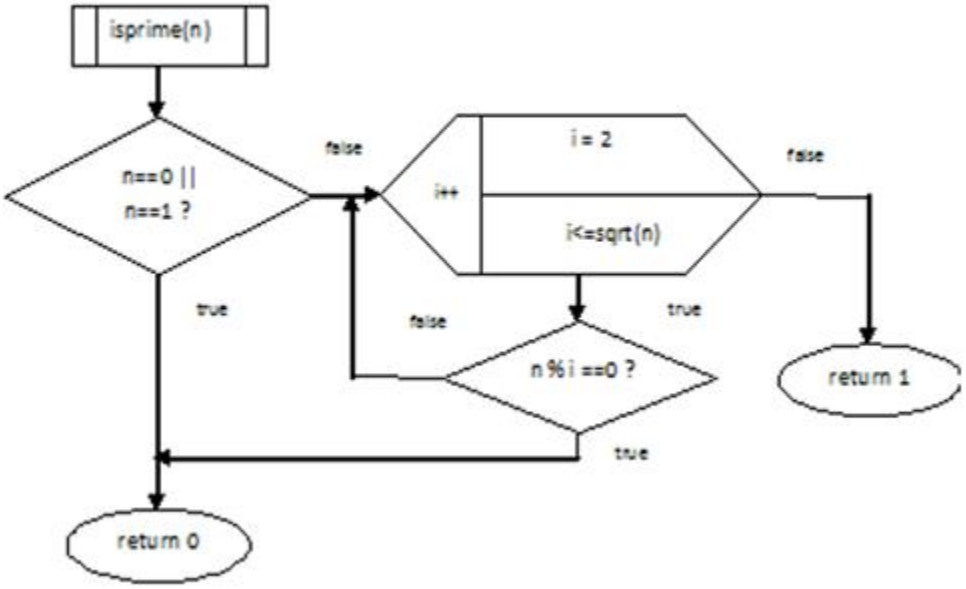
Step 3:[Return to the calling function by sending 1 if the number is prime. Otherwise, returns zero]
return to main

Flowchart

Flowchart: (main function)



Flowchart: (isprime user defined function)



Program

```
#include<stdio.h>
#include<math.h>
 int isprime(int n)
{
int i;
if(n==0||n==1)
return 0;
for(i=2;i<=sqrt(n);i++)
{
if(n%i==0)
return 0;
}
return 1;
}

int main()
{
int num;
printf("\n Enter a number to check for prime or not");
scanf("%d",&num);
if(isprime(num))
printf("\n %d is prime",num);
else
printf("\n %d is not prime",num);
return 0;
}
```

Test cases

Test No	Input Parameters	Expected Output	Obtained Output
1	Enter a number to check for prime or not 9	9 is not prime	9 is not prime
2	Enter a number to check for prime or not 7	7 is prime	7 is prime

PART B

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

Algorithm

Step 1: [Initialize]

start

Step 2:[Read the order and the matrices (2D array) from the user]

Read order of matrices a[ ] [ ] and b[ ][ ] i.e m*n and p*q

Read a[][] and b[][] arrays.

Step 3: [check for possibility of multiplication]

if n!=p

print Matrix multiplication not possible.

goto step 6

else

goto Step 4

end if

Step 4:[Multiply the two matrices a*b(nested for is used)]

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];

end for

end for

end for

Step 5:[Display the resultant C Matrix]

Print the input matrices a[ ][ ],b[ ][ ] and resultant matrix c[ ][ ]

Step 6:[Finished]

Stop

Flowchart

Program

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
int a[10][10],b[10][10],c[10][10];
int m,n,p,q,i,j,k;
printf("Enter the order of the matrix A\n");
scanf("%d%d",&m,&n);
printf("Enter the order of the matrix B\n");
scanf("%d%d",&p,&q);
if(n!=p)
{
printf("\n multiplication not possible");
```

```
exit(0);
}
else
{
printf("Enter the elements of matrix A...\n");
for(i=0;i<m;i++)
{
for(j=0;j<n;j++)
{
scanf("%d",&a[i][j]);
}
}
printf("Enter the elements of matrix B...\n");
for(i=0;i<p;i++)
{
for(j=0;j<q;j++)
{
scanf("%d",&b[i][j]);
}
}
printf("\n Matrix A \n");
for(i=0;i<m;i++)
{
printf("\n");
for(j=0;j<n;j++)
{
printf("%d\t",a[i][j]);
}
}
printf("\n Matrix B \n");
for(i=0;i<p;i++)
{
printf("\n");
for(j=0;j<q;j++)
{
printf("%d\t",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("\n Product of A and B matrices : MATRIX C\n");
for(i=0;i<m;i++)
{
printf("\n");
for(j=0;j<q;j++)
{
printf("%d\t",c[i][j]);
}
}
}
return 0;
}
```

Test cases

Test No	Input Parameters	Expected Output	Obtained Output
1	Enter the order of the matrix A 2 2 Enter the order of the matrix B 2 2 Enter the elements of matrix A... 1 2 3 4 Enter the elements of matrix B... 5 6 7 8	MATRIX A 1 2 3 4 MATRIX B 5 6 7 8 Product of A and B matrices : MATRIX C 19 22 43 50	MATRIX A 1 2 3 4 MATRIX B 5 6 7 8 Product of A and B matrices : MATRIX C 19 22 43 50
2	Enter the order of the matrix A 2 3 Enter the order of the matrix B 2 2 Enter the elements of matrix A... 1 2 3 4 5 6 Enter the elements of matrix B... 5 6 7 8	multiplication not possible	multiplication not possible

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

Algorithm

Step 1: [Initialize]

Start

Step 2: [Set PI=3.142]

PI=3.142

Step 3: [Reading degree]

read degree

Step 4:[Calculate x to convert degrees to radians]

x=degree*(PI/180)

Step 5:[Set the intial values of nume,deno,i]

nume=x;
deno=1;
i=2;

Step 6:[Iterate using do while loop till term>=0.000001]

do
term=nume/deno;
nume=-nume*x*x;
deno=deno*i*(i+1);
sum=sum+term;
i=i+2;
while(fabs(term)>=0.000001);

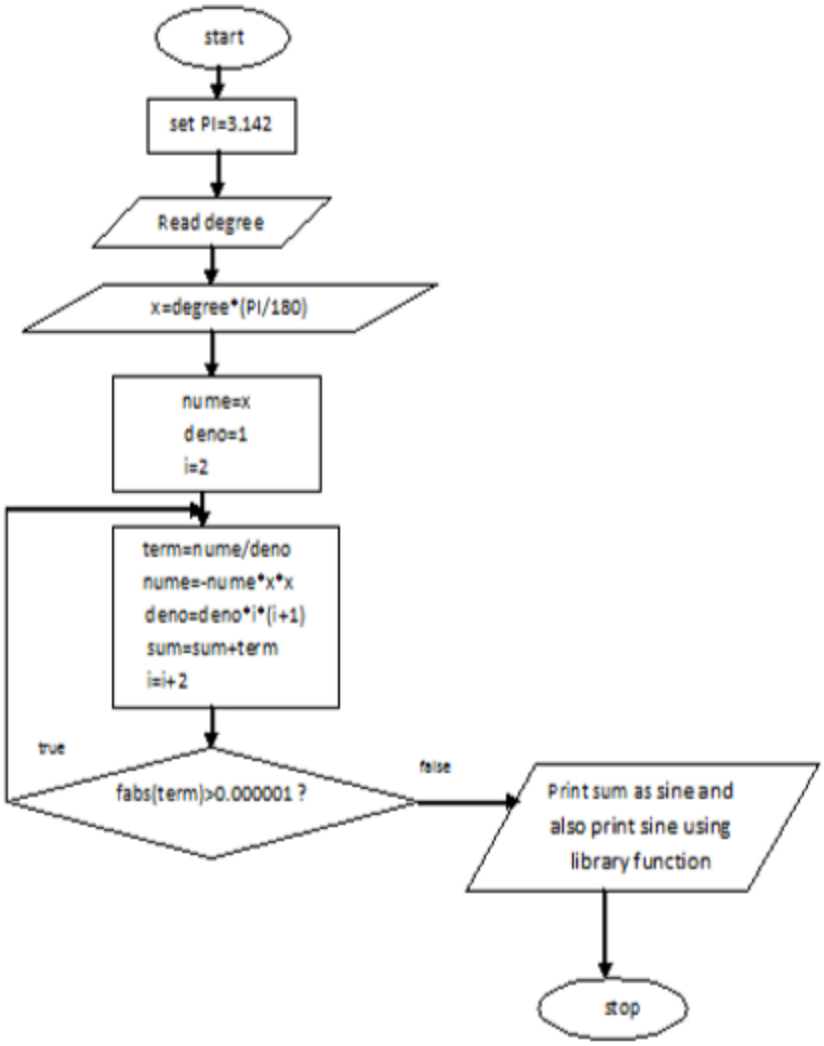
Step 7: [Display the result]

print degree and sum.
Also, print using built in function sin(x).

Step 8: [Finished]

Stop

Flowchart



Program

```
#include<stdio.h>
#include<math.h>
#define PI 3.142
int main()
{
int i,degree;
float x,sum=0,term,nume,deno;
printf("enter the value of degree");
scanf("%d",&degree);
x=degree * (PI/180);
nume=x;
deno=1;
i=2;
do
{
```

```
term=nume/deno;
nume=-nume*x*x;
deno=deno*i*(i+1);
sum=sum+term;  i=i+2;
}
while(fabs(term)>=0.00001);
printf("\nThe sine of %d is %.2f",degree,sum);
printf("\nThe sine function of %d is %.2f using library function",degree,sin(x));
return 0;
}
```

Test cases

Test No	Input Parameters	Expected Output	Obtained Output
1	Degree=0	The sine of 0 is 0 The sine function 0 is 0 using library function	The sine of 0 is 0 The sine function 0 is 0 using library function
2	Degree=60	The sine of 60 is 0.866093 The sine function 60 is 0.866093 using library function	The sine of 60 is 0.866093 The sine function 60 is 0.866093 using library function
3	Degree=-10	The sine of -10 is -0.173670 The sine of -10 is -0.173670 using library function	The sine of -10 is -0.173670 The sine of -10 is -0.173670 using library function

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

Algorithm

Step 1: [Initialize]

start

Step 2: [Input two source strings]

read source1,source2

Step 3: [Caculate length1 of source1 by calling the user defined function,strlength(); Repeat the same for length2 of source2]

length1=strlength(source1);

length2=strlength(source2);

Step 4: [Ouput length1,length2]

print length1,length2

Step 5: [Compare the two strings by calling the user defined function,strcompare()]

k=strcompare(source1,source2);

Step 6: [check k, to find the whether the strings are same or not]

if(k==0)

print Both strings are same

else

print strings are different

end if

Step 7: [Concatenate two strings by calling the user defined function,strconcat() and the concatenated string is stored in source1]

strconcat(source1,source2);

print source1

Step 8: [Finished]

Stop

Algorithm (user defined function - strlength())

Step 1: [Initialize]

Start

Step 2: [set i=0]

i=0

Step 3: [receive the source string as str, read character by character, count one by one until we reach NULL character]

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

i++

end while

Step 4: [return i to the calling function]

return i

Algorithm (user defined function - strcompare())

Step 1: [Initialize]

Start

Step 2: [set i=0]

i=0

Step 3: [Receive both the source strings as str1 and str2, read character by character until they match, if the matched character is a NULL then go out of while loop, when any unmatched character then go out of loop]

while(str1[i]==str2[j])
if(str1[i]=='\0')
break;
end if
i++;
j++;
end while

Step 4: [calculate k]

k=str1[i]-str2[j];

Step 5: [return i to the calling function]

return k

Algorithm (user defined function - strconcat())

Step 1: [Initialize]

Start

Step 2: [set i=0]

i=0

Step 3: [Receive both the source strings as str1 and str2, calculate length of str1 using strlength() as l]

l=strlength(str1)

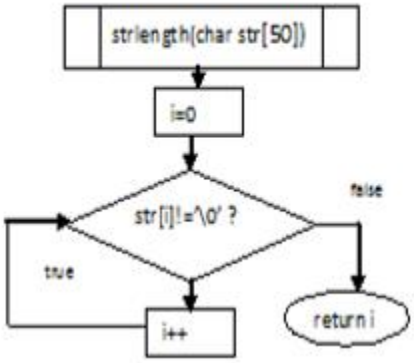
Step 4: [read str2 character by character and store it from the end of str1]

while(str2[i]!='\0')
str1[l+i]=str2[i]
i++
end while

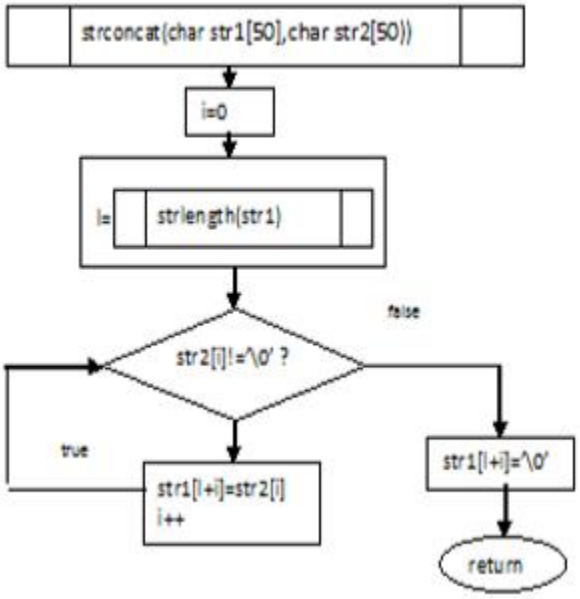
Step 5: [return to the calling function]

return

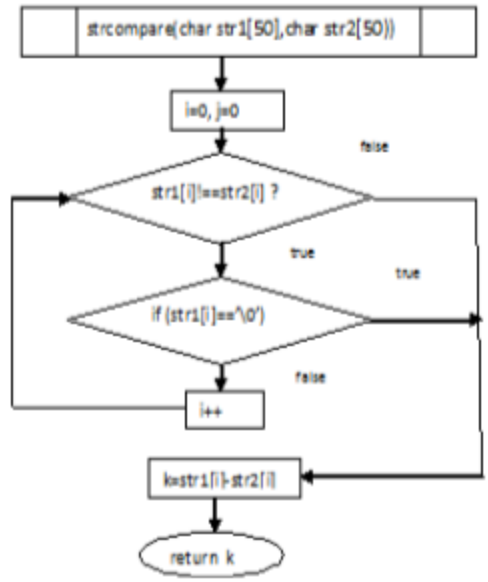
Flow Chart (strlength())



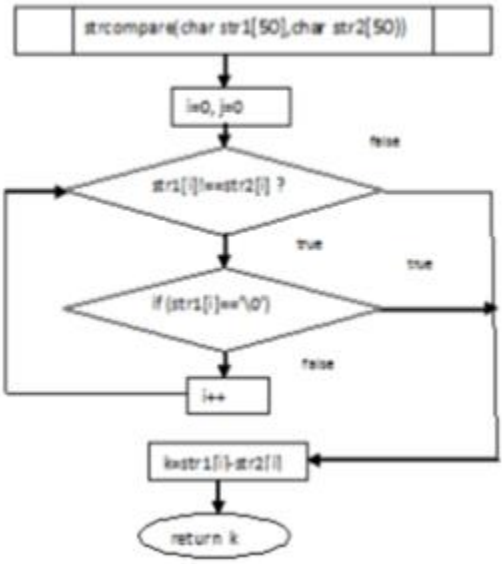
Flow Chart (strconcatenation())



Flow Chart (strcompare())



Flow Chart (strcompare())



Program :

```
#include<stdio.h>
 int strlength(char str1[50]);
void strconcat(char str1[50],char str2[50]);
int strcompare(char str1[50],char str2[50]);
int strlength(char str[50])
{
i=0;  while(str[i]!='\0')
i++;
return i;
```

```
}

void strconcat(char str1[50],char str2[50])
{
int i=0,l;
l=strlength(str1);
while(str2[i]!='\0')
{
str1[l+i]=str2[i];
i++;
}
str1[l+i]='\0';
}

int strcompare(char str1[50],char str2[50])
{
int i=0,j=0,k;
while(str1[i]==str2[i])
{
if(str1[i]=='\0')
break;
i++;
}
k=str1[i]-str2[i];
return k;
}
int main()
{
char source1[50],source2[50],dest[50];
int length1,length2,k;
printf("\n Enter the source string 1:");
gets(source1);
printf("\n Enter the source string 2:");
gets(source2);
length1=strlength(source1);
length2=strlength(source2);
printf("\n string length of string 1 is %d",length1);
printf("\n string length of string 1 is %d",length2);
k=strcompare(source1,source2);
if(k==0)  printf("\n Both string are same");
else
printf("\n Both string are different");
strconcat(source1,source2);
printf("\n concatenated string is ");
```

```
puts(source1);
return 0;
}
```

Test cases

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

Algorithm

Step 1: [Initialize]

start

Step 2: [Input number of elements]

read n

Step 3: [Input unsorted elements in an array]

Read elements in array a[]
for(i=0;i<n;i++)
read a[i]
end for

Step 4: [output unsorted elements in array]

Print elements in array a[]
for(i=0;i<n;i++)
print a[i]
end for

Step 5:[Iterate array a[] in two loops. Outer loop gives number of passes. Inner loop does n-I comparisions and swap task. In each pass, compare each pair of adjacent items. If former element is greater than the latter one, swap them.]

```
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
        end if
end for
end for
```

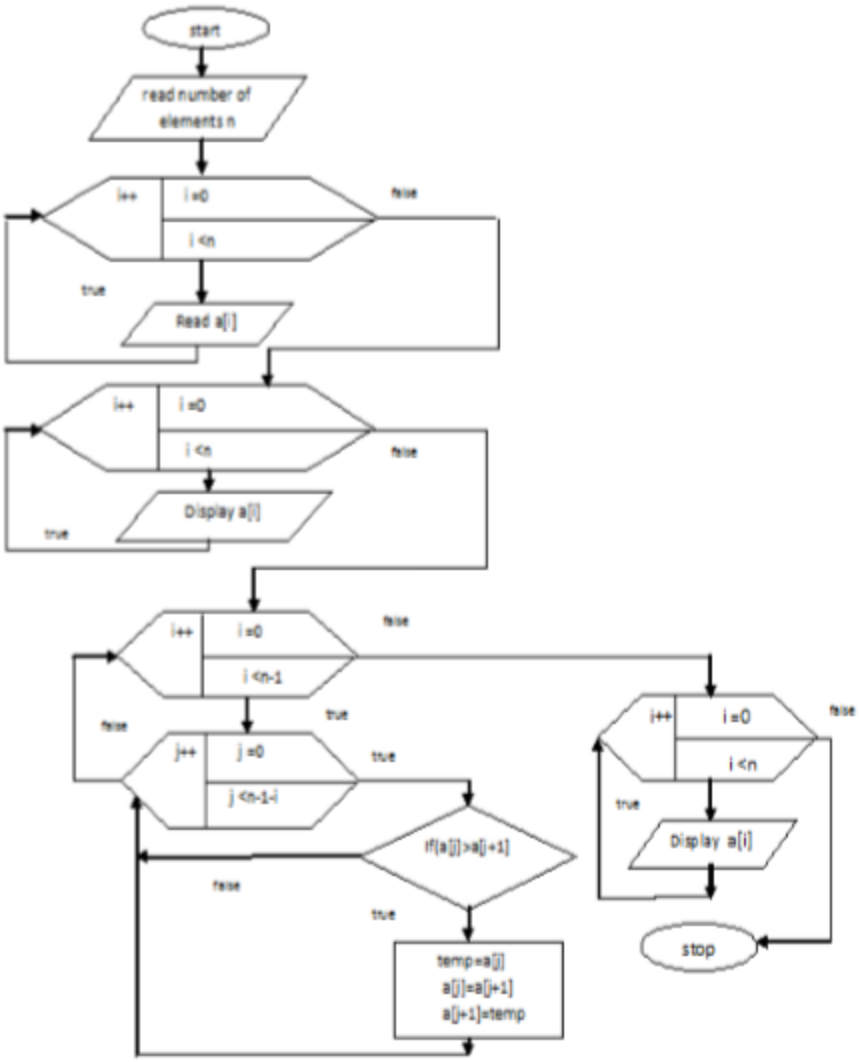
Step 6: [Display sorted elements in array]

Print elements in array a[]
for(i=0;i<n;i++)
print a[i]
end for

Step 7: [Finished]

Stop

Flowchart



Program

```
#include<stdio.h>
int main()
{
int n,i,j,a[10],temp;
printf("\n enter the number of elements");
scanf("%d",&n);
printf("Enter the array elements\n");
for(i=0;i<n;i++)
{
scanf("%d",&a[i]);
}
printf("Original elements are \n");
for(i=0;i<n;i++)
```

```
{
printf("%d\t",a[i]);
}
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;
}
}
printf("\nThe sorted elements are\n");
for(i=0;i<n;i++)
printf("%d\n",a[i]);
return 0;
}
```

Test case

Test No	Input Parameters	Expected Output	Obtained Output
1	Enter the number of elements 5 Enter the array elements 23 4 6 12 40	Original elements are 23 4 6 12 40 The sorted elements are 4 6 12 23 40	Original elements are 23 4 6 12 40 The sorted elements are 4 6 12 23 40
2	Enter the no.of elements 4 Enter the array elements -3 -9 12 6	Original elements are -3 -9 12 6 The sorted elements are -9 -3 6 12	Original elements are -3 -9 12 6 The sorted elements are -9 -3 6 12

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

Algorithm:

Step 1:[Initialize]

Start

Step 2: [Read a number from the user]

Read num

Step 3: [to find square root, call the user defined function square()]

3.1 if (num>=0) is true then

if num==0 then res=0

Else

if num==1then res=1

else

res=square(num), call the user defined function square()

end if

3.2 otherwise, print no square root for negative number, go to step 5

Step 4: [Print the square root]

Display "square root of the num is res"

Step 5: [Finished]

Stop

Algorithm (user defined function –square())

Step 1:[Initialize]

Start

Step2: [set s=n/2 as first guess]

S=n/2

Step 3: [Receive n from calling from calling function tofind the square root of n]

for(i=0;i<n;i++)

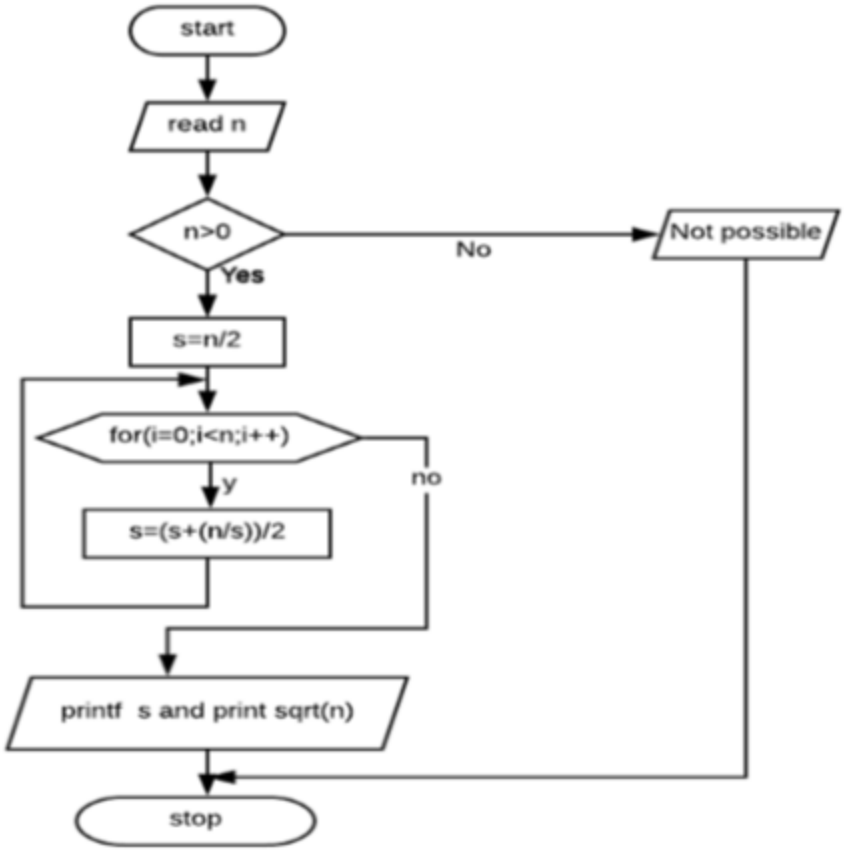
s=(s+n/s)/2

end for

Step 4: [Return s to main()]

return s

Flowchart



Program

```
#include<stdio.h>
#include<math.h>
int main()
{
float n,s;
int i;
printf("\n Enter the number to find the square root\n"):
scanf("%f",&n);
if(n>0)
{
s=n/2;
for(i=0;i<n;i++)
s=(s+(n/2))/2;
printf("Square root without using Library function is %f\n",s);
printf("Square root  using Library function is %f\n",sqrt(n));
```

```
}
else
printf("\n Not possible to find square root");
return 0;
}
```

Test case

Test No	Input Parameters	Expected Output	Obtained Output
1	Enter a number 0	The square root of 0.000000 is 0.000000	The square root of 0.000000 is 0.000000
2	Enter a number -1	No square root for negative number	No square root for negative number
3	Enter a number 49	The square root of 49.000000 is 7.000000	The square root of 49.000000 is 7.000000
4	Enter a number 50.5	The square root of 50.500000 is 7.106335	The square root of 50.500000 is 7.106335

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.

Algorithm

Step 1: [Initialize]

Start

Step 2:[Create a structure student]

Create a structure called student with fields rollno,name,marks,grade with suitable datatypes.

```
struct student
{
int rollno;
char name[20];
float marks;
};
```

Step 3:[Set found=0, sum=0]

found=0

Step 4:[Read the number of student details]

Read n

Step 5: [Read n student details. Iterate using a for loop for inputting rollno, name, marks of each student]

```
for(i=0;i<n;i++)
printf("\ nenter the %d student details",i+1);
printf("\n enter roll number:");
scanf("%d",&s[i].rollno);
printf("\ n enter student name:");
scanf("%s",s[i].name);
printf("\n enter the marks:");
scanf("%d",&s[i].marks);
end for
```

Step 6:[Display the n student details]

Print the rollno, name, marks of all students by iterating using a for loop.

Step 7:[Calculate sum of the marks of all students]

```
for(i=0;i<n;i++)
sum=sum+s[i].marks
end for
```

Step 8:[calculate average]

```
average=sum/n;
```

Step 9:[Display the names of students scoring above average]

```
for(i=0;i<n;i++)
if(s[i].marks>=average)
printf("%s\t",s[i].name)
end if
end for
```

Step 10:[Display the names of students scoring below average]

```
for(i=0;i<n;i++)
```

if(s[i].marks<average)
printf("%s\t",s[i].name)
end if
end for

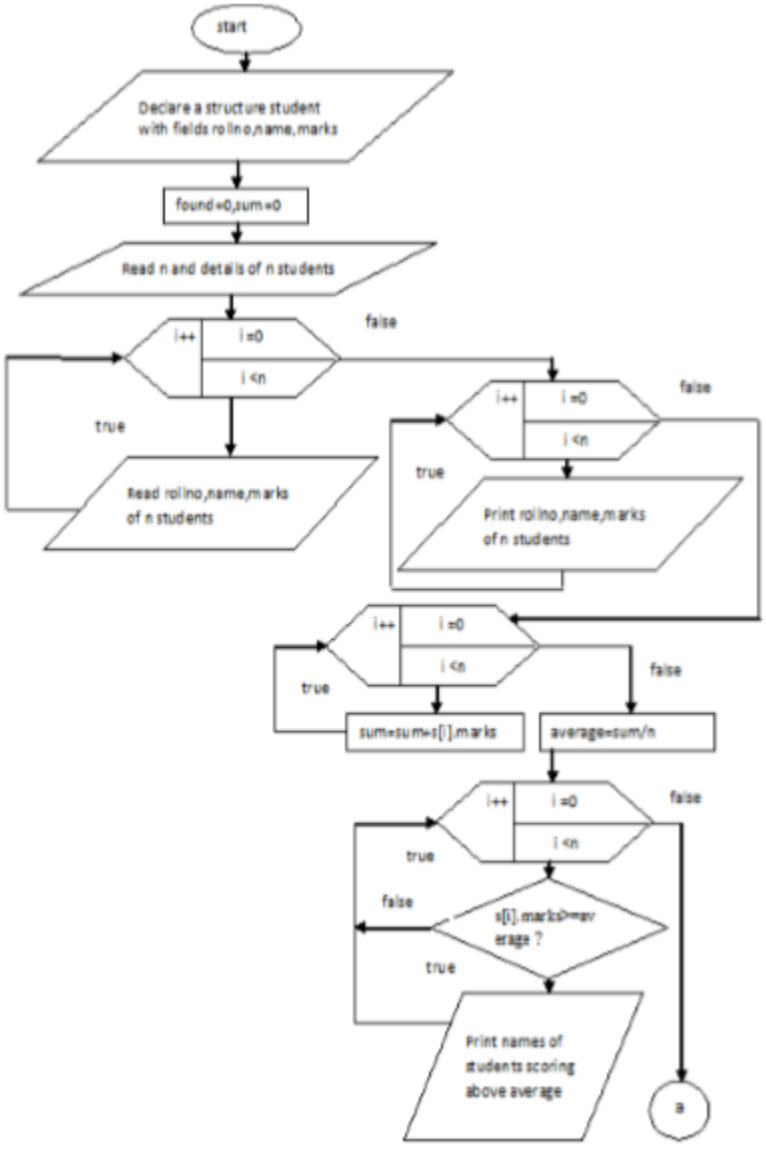
Step 11:[Display no students scored below average, if found==0]

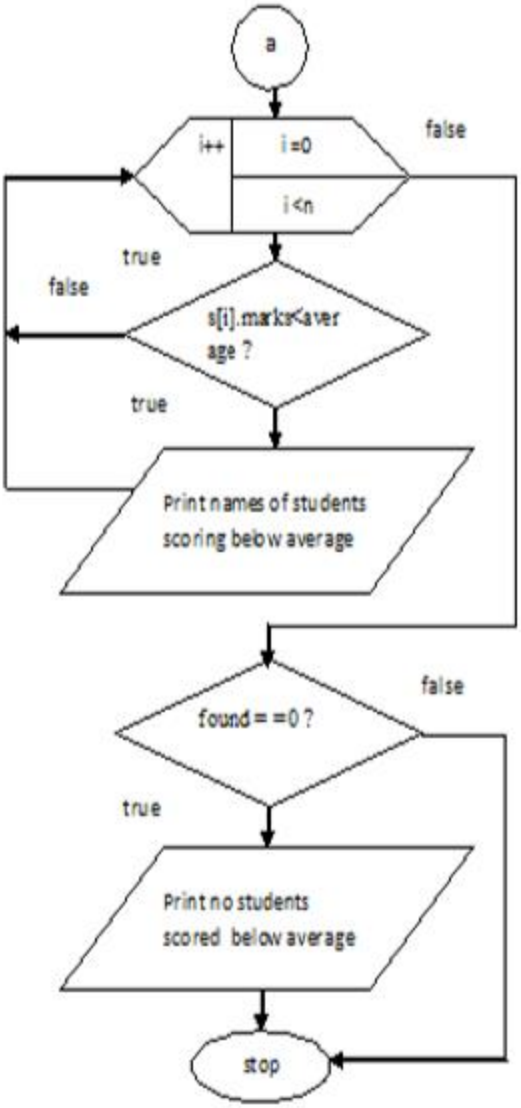
if(found==0)
printf("\n no students scored below average")
end if

Step 12:[Finished]

Stop

Flowchart





Program

```
#include<stdio.h>
#include<string.h>
struct student
{
int rollno;
char name[20];
int marks;
};
int main()
{
int i,n,found=0;
struct student s[10];
float sum=0,average;
printf("\nEnter the number of student details");
```

```
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("\nenter the %d student details",i+1);
printf("\n enter roll number:");
scanf("%d",&s[i].rollno);
printf("\n enter student name");
scanf("%s",s[i].name);
printf("\n enter the marks:");
scanf("%d",&s[i].marks);
}
printf("\nStudent details are\n");
printf("\nRollno\t\tName\t\tMarks\n");
for(i=0;i<n;i++)
printf("%d\t\t%s\t\t%d\n",s[i].rollno,s[i].name,s[i].marks);
for(i=0;i<n;i++)
sum=sum+s[i].marks;
average=sum/n;
printf("\nAVERAGE=%f",average);
printf("\n students scoring above average\n");
for(i=0;i<n;i++)
{
if(s[i].marks>=average)
{
printf("%s\t",s[i].name);
}
}
printf("\n students scoring below average\n");
for(i=0;i<n;i++)
{
if(s[i].marks<average)
{
printf("%s\t",s[i].name);
found=1;
}
}
if(found==0)
printf("\n no students scored below average");
return 0;
}
```

Test case

Test No	Input Parameters	Expected Output	Obtained Output
1	Enter the number of student details 4 enter the 1 student details enter roll number: 100 enter student name:austin enter the marks: 56 enter the 2 student details enter roll number: 200 enter student name:banu enter the marks: 67 enter the 3 student details enter roll number: 300 enter student name:charan enter the marks: 90 enter roll number: 400 enter student name:dina enter the marks: 69	Student details are Rollno Name Marks 100 austin 56.000 200 banu 67.000 300 charan 90.000 400 dina 69.000 Average=70.500000 Students scoring above average charan Students scoring below average austin banu dina	Student details are Rollno Name Marks 100 austin 56.000 200 banu 67.000 300 charan 90.000 400 dina 69.000 Average=70.500000 Students scoring above average charan Students scoring below average austin banu dina
2	Enter the number of student details 4 enter the 1 student details enter roll number: 100 enter student name:austin enter the marks: 100 enter the 2 student details enter roll number: 200 enter student name:banu enter the marks: 100 enter the 3 student details enter roll number: 300 enter student name:charan enter the marks: 100 enter roll number: 400 enter student name:dina enter the marks: 100	Student details are Rollno Name Marks 100 austin 100.000 200 banu 100.000 300 charan 100.000 400 dina 100.000 Average=100.000000 Students scoring above average Austin banu charan dina Students scoring below average No students scored below average	Student details are Rollno Name Marks 100 austin 100.000 200 banu 100.000 300 charan 100.000 400 dina 100.000 Average=100.000000 Students scoring above average Austin banu charan dina Students scoring below average No students scored below average

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

Algorithm

Step 1: [Initialize]

Start

Step 2:[Read the no of elements and array elements]

Read n

Read a[]

Step 3: [Set starting address of array to a pointer variable]

ptr=a

Step 4:[Iterate using a for loop to find sum using pointers]

for(i=0;i<n;i++)

sum=sum+*ptr ptr++

end for

Step 5:[Calculate mean]

mean=sum/n

Step 6: [Set starting address of array to a pointer variable]

ptr=a

Step 7:[Iterate using a for loop to find sumstd using pointers]

for(i=0;i<n;i++)

sumstd=sumstd+pow((*ptr-mean),2)

ptr++

end for

Step 8:[Calculate standard deviation]

std=sqrt(sumstd/n)

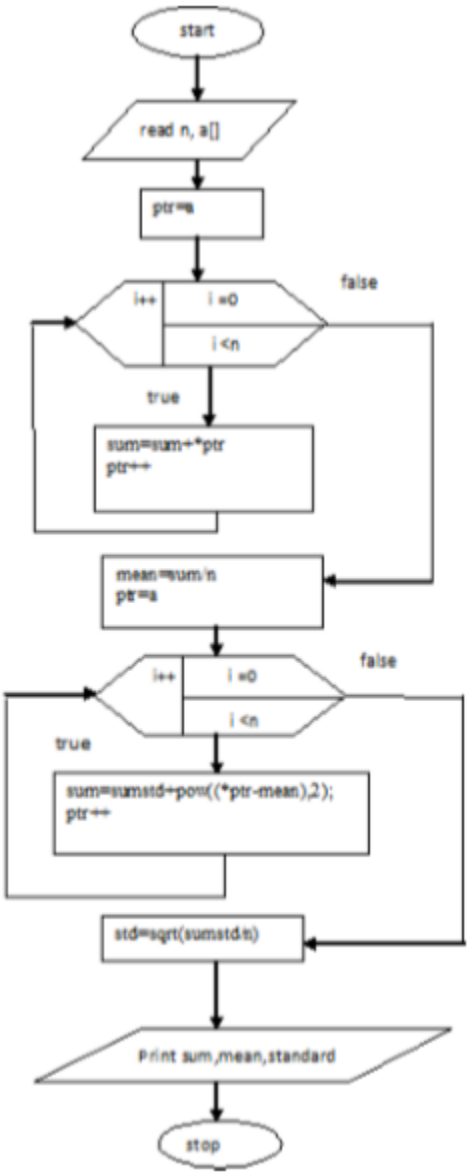
Step 9:[Display the result]

Print sum,mean,std

Step 10:[Finished]

Stop

Flowchart



Program

```
#include<stdio.h>
#include<math.h>
int main()
{
float a[10],*ptr,mean,std,sum=0,sumstd=0;
int n,i;
printf("\n Enter the number of elements");
scanf("%d",&n);
printf('\n Enter the array elements");
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=%f\n",sum);
printf("Mean=%f\n",mean);
printf("Standard Deviation=%f\n",std);
return 0;
}
```

Test case

Test No	Input Parameters	Expected Output	Obtained Output
1	Enter the number of elements 5 Enter the array elements 1 5 9 6 7	Sum= 28 Mean= 5.6 Standard Deviation= 2.09	Sum= 28 Mean= 5.6 Standard Deviation= 2.09
2	Enter the number of elements 4 Enter the array elements 2.3 1.1 4.5 2.78	Sum= 10.68 Mean= 2.67 Standard Deviation= 0.863	Sum= 10.68 Mean= 2.67 Standard Deviation= 0.863

15.Implement Recursive functions for Binary to Decimal Conversion.

Algorithm

Step 1: [Initialize]

Start

Step 2:[Set decimal=0]

Set decimal=0

Step 3:[Enter the binary input]

Read binary

Step 4: [Find the length of binary using strlen() and store in len]

len=strlen(binary)

Step 5:[call the user defined function recursion to find the decimal value for binary]

recursion(binary, &decimal, &len);

Step 6:[Display decimal value]

Print decimal

Step 7:[Finished]

Stop

Algorithm (user defined function - recursion())

Step 1: [Initialize]

Start

Step 2:[set static int num, i=0]

static int num, i = 0

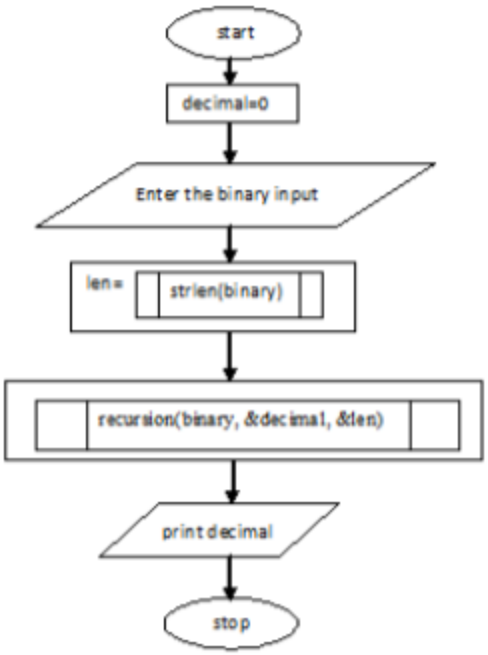
Step 3: [Until len becomes zero,do the following steps and call recursively, recursion again and again]

```
if (*len > 0)  *len = *len - 1;
num = *(binary + *len) - '0';
*decimal = *decimal + (num * pow(2, i++));
recursion(binary, decimal, len);
end if
```

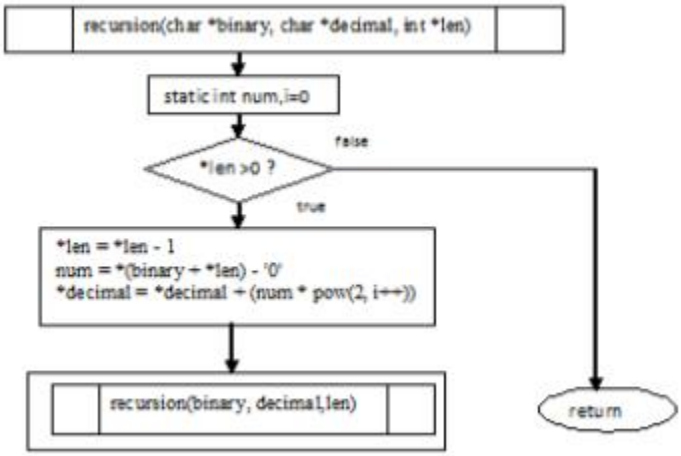
Step 4: [return to the calling function main()]

Return

Flowchart



Flow Chart (recursion())



Program

```
#include <stdio.h>
#include <string.h>
#include <math.h>
void recursion(char *binary, int *decimal, int *len)
{
static int num, i = 0;
if (*len > 0)
{
*len = *len - 1;
num = *(binary + *len) - '0';
```

```
*decimal = *decimal + (num * pow(2, i++));
recursion(binary, decimal, len);
}
return;
}
int main()
{
char binary[256];
decimal = 0, len;
printf("Enter the binary input:");
gets(binary);
len = strlen(binary);
recursion(binary, &decimal, &len);
printf("Decimal Value: %d\n", decimal);
return 0;
}
```

Test case

Test No	Input Parameters	Expected Output	Obtained Output
1	Enter the binary input:1001	Decimal value: 9	Decimal value: 9
2	Enter the binary input:1001001	Decimal value: 73	Decimal value: 73