LABORATORY PROGRAMS

PART-A

PROGRAM 1

Simulation of a Simple Calculator.

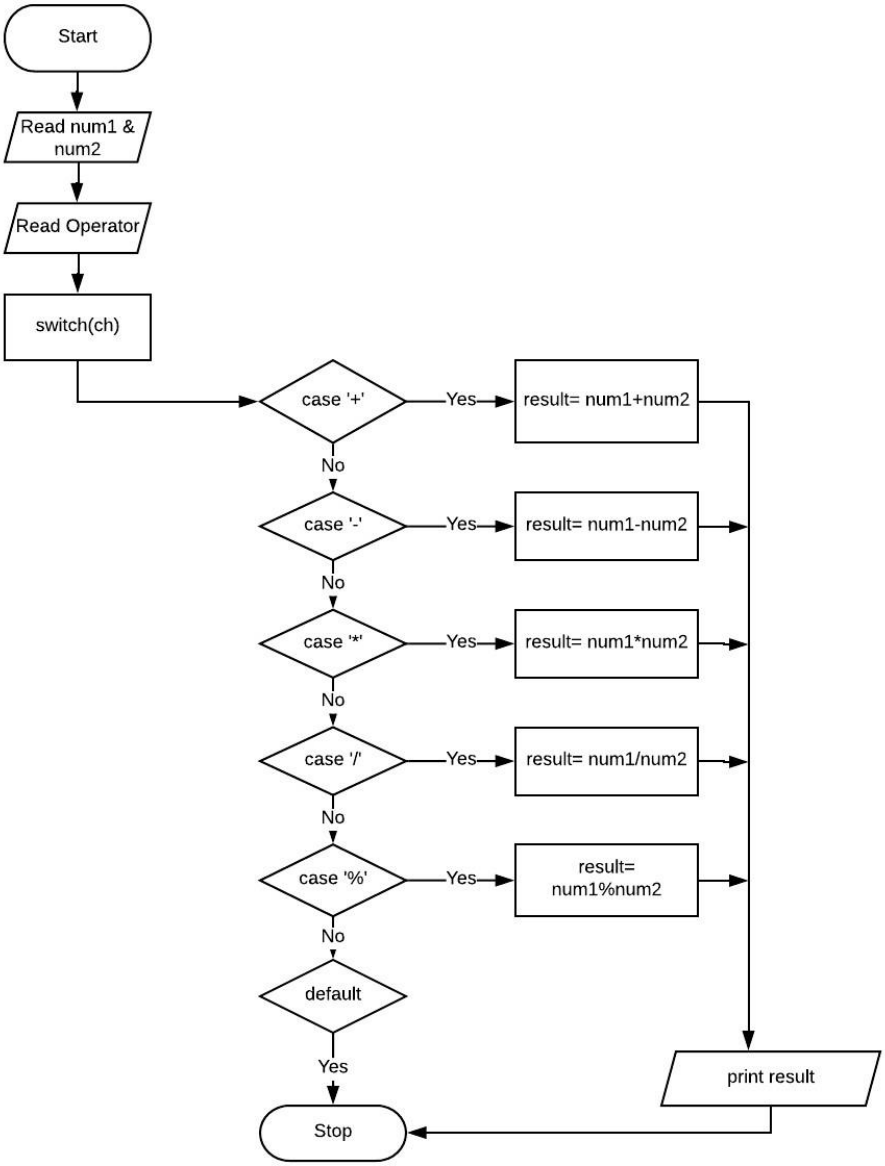
Procedure: This program takes an arithmetic operator +,-,*,/,% and two operands from the user and performs the calculation on the two operands depending upon the operator entered by the user.
Input: An operator and two operands.
Expected Output: Performs calculation and display result depending upon the operator.

ALGORITHM

Algorithm Calculator

Step 1: Start
Step 2: Read num1 and num2
Step 3: Enter the operator
Step 4: Evaluate operator wit case statements
    Step 4.1: case '+' : result = num1+num2
        goto step 6
        Print result
    Step 4.2: case '-' : result = num1-num2
        goto step 6
        Print result
    Step 4.3: case '*' : result = num1*num2
        goto step 6
        Print result
    Step 4.4: case '/' : result = (float)num1/(float)num2
        goto step 6
        Print result
    Step 4.5: case '%' : result = num1%num2
        goto step 6
        Print result
Step 5: Enter operator is invalid then
    Print "Invalid Operation"
Step 6: Print result
Step 7: Stop

FLOWCHART


Start
Read num1 & num2
Read Operator
switch(ch)
case '+'
Yes
result= num1+num2
No
case '-'
Yes
result= num1-num2
No
case '*'
Yes
result= num1*num2
No
case '/'
Yes
result= num1/num2
No
case '%'
Yes
result= num1%num2
No
default
Yes
print result
Stop

/* Program to design simple commercial calculator */

```
#include <stdio.h>
 void main()
{
   int num1,num2;      /*Declaration of variables*/
   float result=0;        /*Declaration of variables*/
   char ch;    //to store operator choice
   printf("Enter first number: ");
   scanf("%d",&num1);         /*read first number*/
   printf("Enter second number: ");
   scanf("%d",&num2);          //read second number
   printf("Choose operation to perform (+,-,*,/,%): ");
   scanf(" %c",&ch);             //read an operator
   switch(ch)             //searching for an operator case
   {
      case '+':result=num1+num2;
         break;
      case '-':result=num1-num2;
         break;
      case '*': result=num1*num2;
         break;
      case '/':result=(float)num1/(float)num2;
         break;
     case '%': result=num1%num2;
         break;
    default: printf("Invalid operation.\n");
   }
  printf("Result: %d %c %d = %f\n",num1,ch,num2,result);     //display output on screen

}
```

OUTPUT:

.....

Enter first number: 5

Enter second number: 2

Choose operation to perform (+,-,*,/,%): *

Result: 5 * 2 = 10.000000

PROGRAM 2

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

Procedure: The equation in the form $ax^2+bx+c=0$ is called quadratic equation. Read the coefficients a,b,c and calculate discriminant. Based on the discriminant value, calculate roots and print them with suitable messages.

Input: Three coefficients of quadratic equation $ax^2+bx+c=0$: a, b, c

Expected Output: This program computes all possible roots for a given set of coefficients with appropriate messages. The possible roots are: Real and Equal roots, Real and distinct roots, imaginary roots.

ALGORITHM

Algorithm: Quadratic Equation

[This algorithm takes three coefficients as input and compute the roots]

Step 1: [Start of the algorithm]

Start

Step 2: [Read the coefficients]

Read non zero coefficients a,b,c

Step 3: [calculate the discriminant]

d←b*b-4*a*c

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

if (d=0)

x1←-b/(2*a)

x2←-b/ (2*a)

Print “Roots are equal”

Print x1, x2

Go to step 7

Step 5: [check if roots are real and distinct]

If(d>0)

x1←(-b+sqrt (d)/ (2*a))

x2←(-b-sqrt (d)/ (2*a))

Print “Roots are real and distinct”

Print x1,x2

Go to step 7

Step 6: [check if roots are imaginary]

If(d<0)

x1←-b/(2*a)

x2←sqrt (fabs(d))/(2*a)

Print “ The roots are complex”

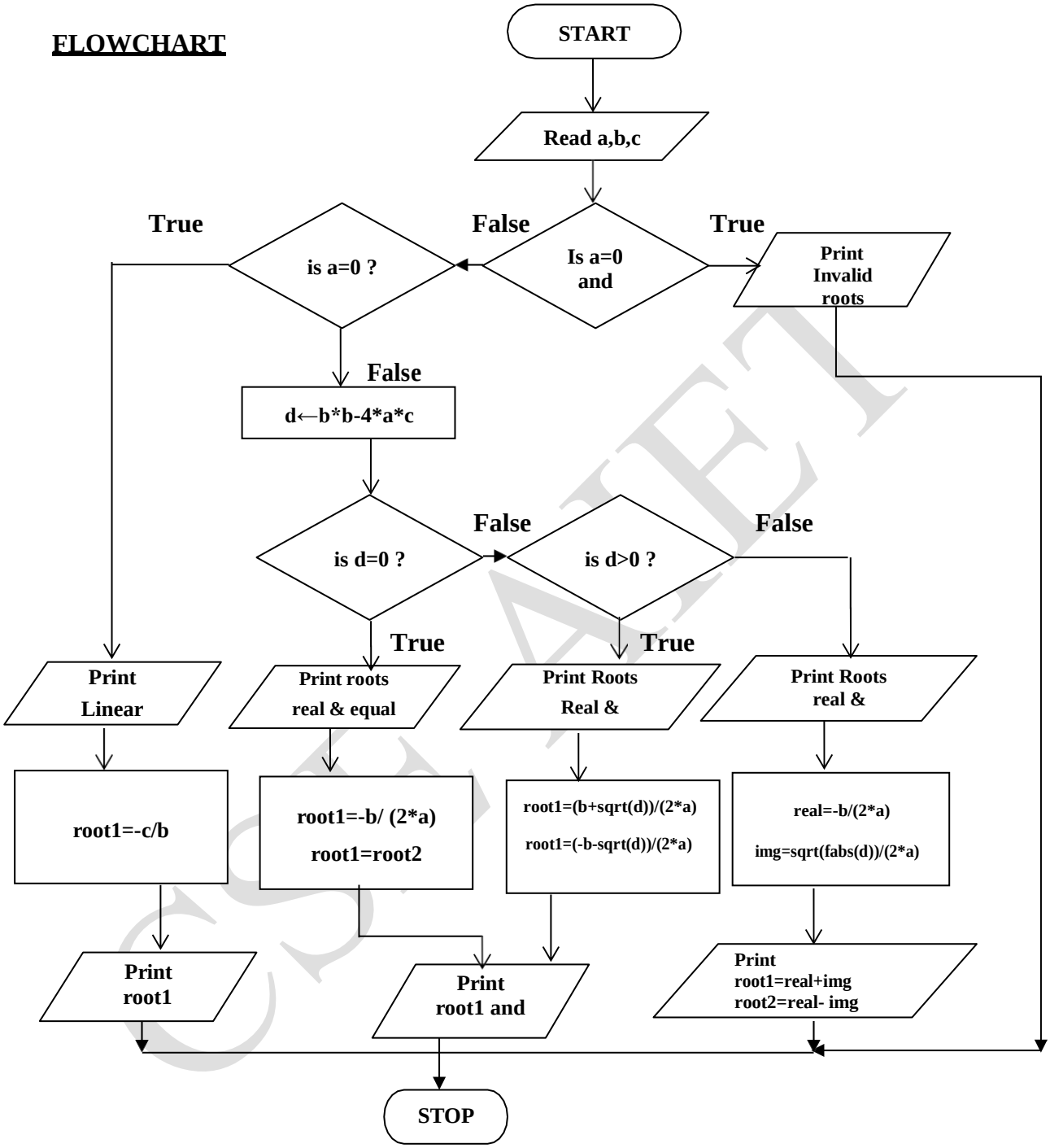
Print “Root1 ← “, x1+ix2

Print “ Root2←“, x1-ix2

Step 7: [terminate the algorithm]

Stop

FLOWCHART



Program:

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

void main()
{
        float a,b,c,d,rpart,ipart,root1,root2;
        clrscr();
        printf("Enter three co-efficient\n");
        scanf("%f%f%f",&a,&b,&c);

        if(a==0 && b==0)
        {
                printf(“Invalid inputs”);

        }
        else if(a==0)
        {
                printf(“Linear Equation\n”);
                root1=-c\b;
                printf(“Root=%f\n”,root1);
        }
        else
        {
                d=(b*b)-(4*a*c);
                if(d==0)
                {
                        printf("The roots real and equal\n");
                        root1= -b/(2*a);
                        root2=root1;
                        printf("The roots are root1=%.3f and root2=%.3f\n",root1,root2);
                }
                else if(d>0)
                {
                        printf("The roots are real and distinct\n");
                        root1=(-b+sqrt(d))/(2*a);
                        root2=(-b-sqrt(d))/(2*a);
                        printf("The roots are root1=%.3f and root2=%.3f\n",root1,root2);
                }
                else
                {
                        printf("The roots are imaginary\n");
                        rpart=-b/(2*a);
                        ipart=sqrt(fabs(d))/(2*a);
```

```
                    printf("The first root root1=%.3f+i%.3f\n",rpart,ipart);
                    printf("The second root root2=%.3f-i%.3f\n",rpart,ipart);
                }
        }
        getch();
}
```

OUTPUT:

.....

Run1:

Enter three co-efficient
1
2
3
The roots are imaginary
The first root root1=-1.000+i1.414
The second root root2=-1.000-i1.414

Run2:

Enter three co-efficients
1
5
2
The roots are real and distinct
The roots are root1=-0.438 and root2=-4.561

Run3:

Enter three co-efficients
1
2
1
The roots real and equal
The roots are root1=-1.000 and root2=-1.000

PROGRAM 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.

Input: Consumer name and number of units

Expected Output: This program checks all else if ladder conditions as per number of units consumed by consumer and display appropriate output

ALGORITHM

Algorithm power

Step 1: Start

Step 2: Read name & unit

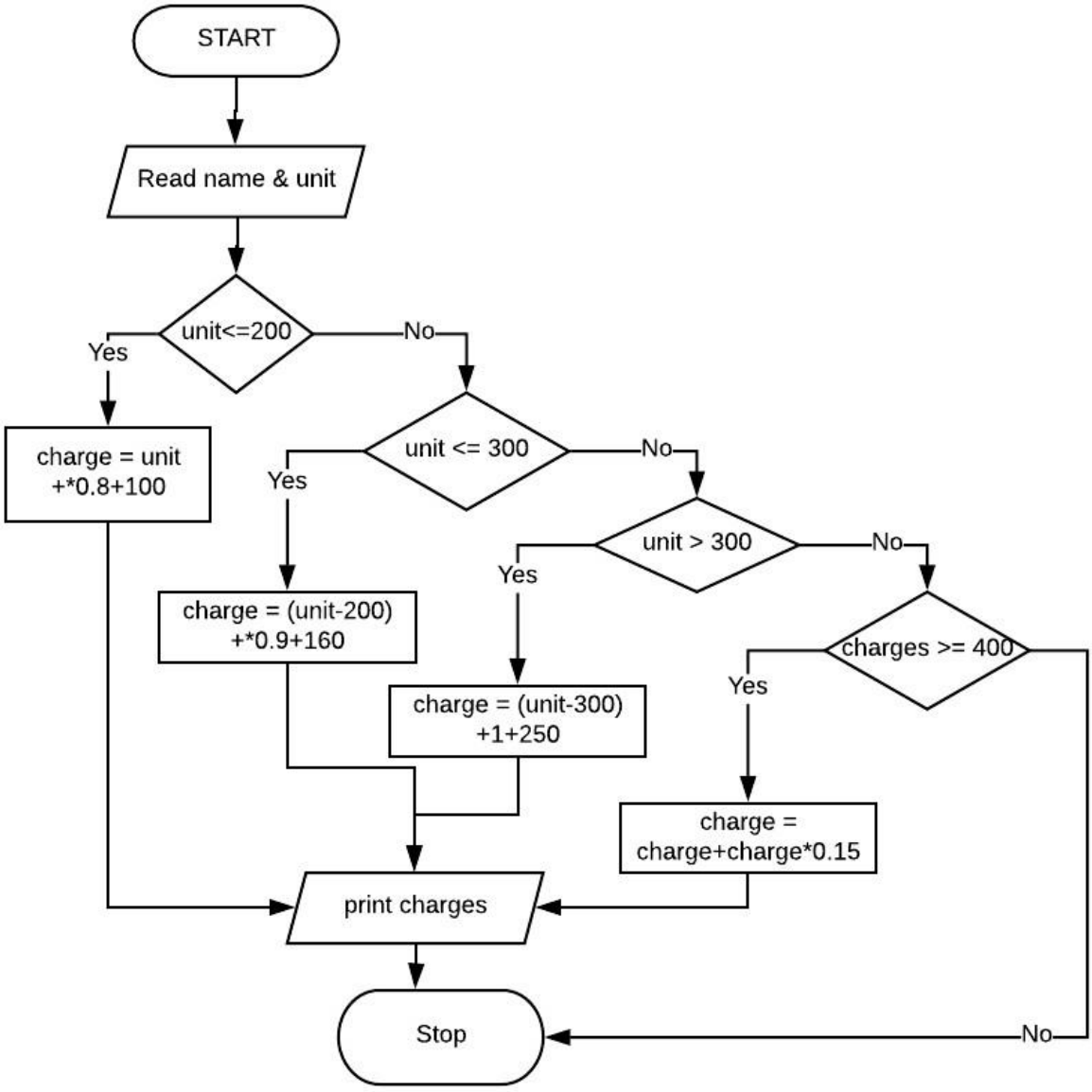
Step 3: if(unit <=200)

```
charge = unit *0.8+ 100
else if(unit<=300)
        charge=(unit-200) *0.90+160;
else if(unit>300)
        charge=(unit-300) *1+250;
if(charge>=400)
        charge=charge + charge*0.15;
```

Step 4: Display the charges

Step 5: Stop

FLOWCHART


START
Read name & unit
unit<=200
Yes
No
charge = unit
+*0.8+100
unit <= 300
Yes
No
charge = (unit-200)
+*0.9+160
unit > 300
Yes
No
charge = (unit-300)
+1+250
charges >= 400
Yes
No
charge =
charge+charge*0.15
print charges
Stop

```
#include <stdio.h>
void main()
{
char name[10];
float unit, charge;
printf("Enter your name and unit Consumed:\n");
scanf("%s%f", name, &unit);
if(unit<125)
charge=100;
else if(unit<=200)
charge=unit*.80;
else if(unit<=300)
charge=(unit-200)*0.90+160;
else
if(unit>300)
charge=(unit-300) *1+250;
if(charge>=400)
charge=charge + charge*0.15;
printf("Name: %s\n Charge: %.3f\n",name,charge);
getch();
}
```

OUTPUT:

Enter your name and unit Consumed:

ARUN

100

Name: ARUN

Charge: 100.000

PROGRAM 4

Implement Binary search on integers/ Names.

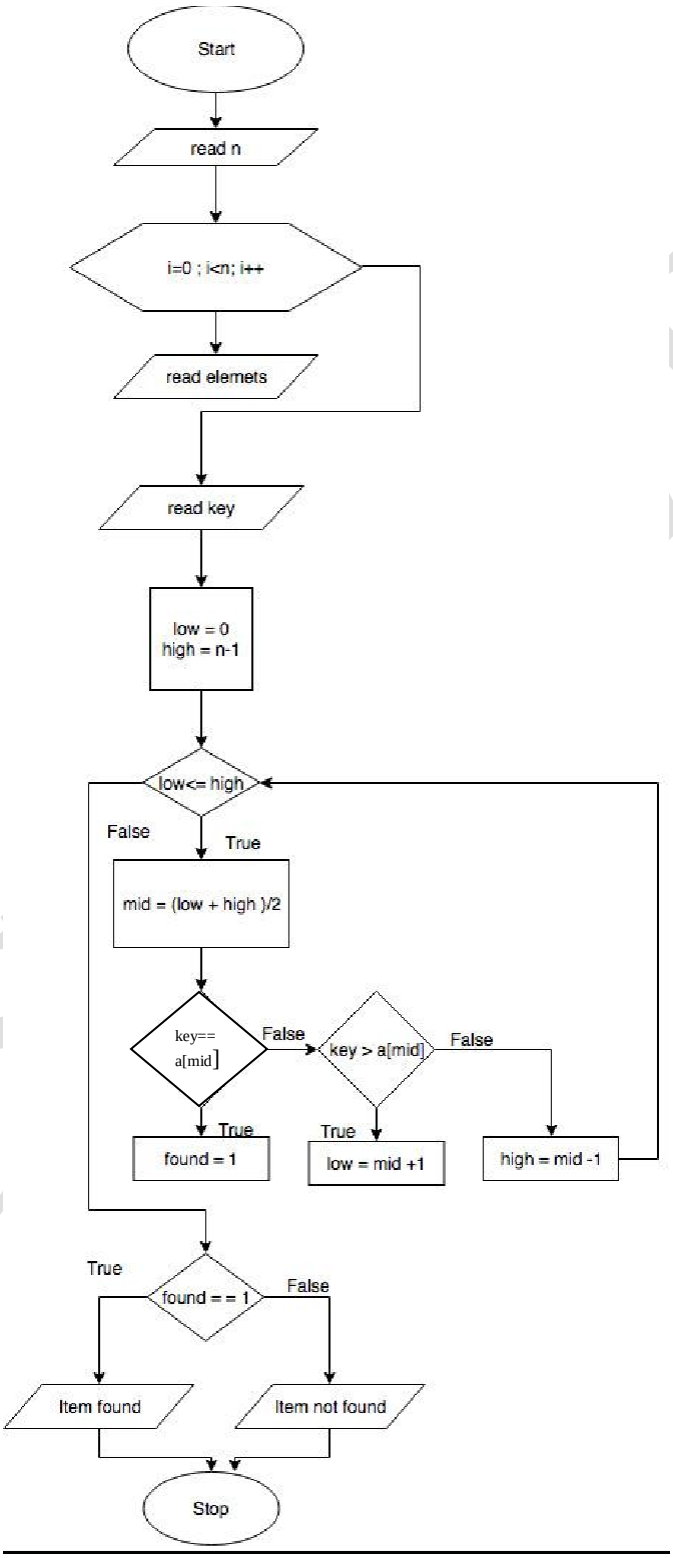
Procedure: Input N array elements and to find whether element is present or not.
Input: Array elements.
Expected Output: Successful search or unsuccessful search.

ALOGRITHM

Algorithm Binary Search

Step 1: Start
Step 2: Read n
Step 3: Repeat for i=0 to n-1
Read a[i]
Step 4: read key
Step 5: Initialize low =0 high = n-1
Step 6: Repeat through step 6 while (low <= high)
mid = (low+ high)/2
if(key==a[mid])
found=1
else if(key>a[mid])
low=mid+1
else
high=mid-1
end while
Step 7: if(found ==1)
print "Item found"
else
print" Item not found"
Step 8: Stop

FLOWCHART


Start
read n
i=0 ; i<n; i++
read elemets
read key
low = 0
high = n-1
low<= high
False
True
mid = (low + high)/2
key== a[mid]
False
key > a[mid]
False
True
True
True
found = 1
low = mid +1
high = mid -1
True
found = = 1
False
Item found
Item not found
Stop

```
#include<stdio.h>

void main()
 {
  int i,n,a[10],mid,low,high,key, found=0;

  printf("\n Enter the number of elements:\n");
  scanf("%d", &n);
  printf("Enter the array element in the ascending order\n");
  for(i=0;i<n;i++)
   {
    scanf("%d", &a[i]);
   }
  printf("\n Enter the key element to be searched\n");
  scanf("%d", &key);
  low=0;
  high=n-1;
  while(low<=high)
   {
          mid=(low+high)/2;
         if(key==a[mid])
         {
                 found=1;
                 exit(0);
         }
         else if(key>a[mid])
         {
                 low=mid+1;
         }
         else
         {
                 high=mid-1;
         }
   }
 if(found ==1)
         printf("Item found in position : %d",mid+1);
else
         printf("\n Item not found\n");
  getch();
 }
```

OUTPUT:

RUN 1:

```
Enter the number of elements:
5
Enter the array element in the ascending order
10
20
30
40
50
Enter the key element to be searched
30
Item found in position: 3
```

RUN 2:

```
Enter the number of elements:
5
Enter the array element in the ascending order
10
20
30
40
50
Enter the key element to be searched
70
Item not found
```

PROGRAM 5

Implement Matrix multiplication and validate the rules of multiplication.

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

ALGORITHM

Algorithm: Matrix_Multipication

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

Step 10: [Print the resultant matrix]
Repeat step 10 for i←0 to m-1
Repeat for j←0 to n-1
Print c[i][j]
Step 11: [terminate the algorithm]
Stop

Program:

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int m,n,p,q,i,j,k,a[10][10],b[10][10],c[10][10];
        clrscr();
        printf("Enter the size matrix A \n");
        scanf("%d%d",&m,&n);
        printf("Enter the size matrix B \n");
        scanf("%d%d",&p,&q);

        if(n!=p)
        {
                printf(“Matrix multiplication is not possible\n”);
        }
        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]);
                                }
                        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('A-matrix is\n");
                for(i=0;i<m;i++)
                {
                        for(j=0;j<n;j++)
                        {
                                printf("%d\t",a[i][j]);
                        }
                        printf("\n");
                }

                printf("B- matrix is \n");
                for(i=0;i<p;i++)
                {
                        for(j=0;j<q;j++)
                        {
                                printf("%d\t",b[i][j]);
                        }
                        printf("\n");
                }

                printf("The resultant matrix C is \n");
                for(i=0;i<m;i++)
                {
                        for(j=0;j<q;j++)
                        {
                                printf("%d\t",c[i][j]);
                        }
                        printf("\n");
                }
        }
        getch ();
}
```

OUTPUT (PASS 1)

.....

Enter the size of matrix A
2 3
Enter the size of matrix B
4 5
Matrix multiplication is not possible

OUTPUT (PASS 2)

Enter the size of matrix A
2 2

```
Enter the size of matrix B
2 2
Enter the elements of Matrix A
1       1
1       1
Enter the elements of Matrix B
1       1
1       1

Matrix-A is
        1       1
        1       1
Matrix-B is
        1       1
        1       1
The resultant matrix c is

        2       2
        2       2
```

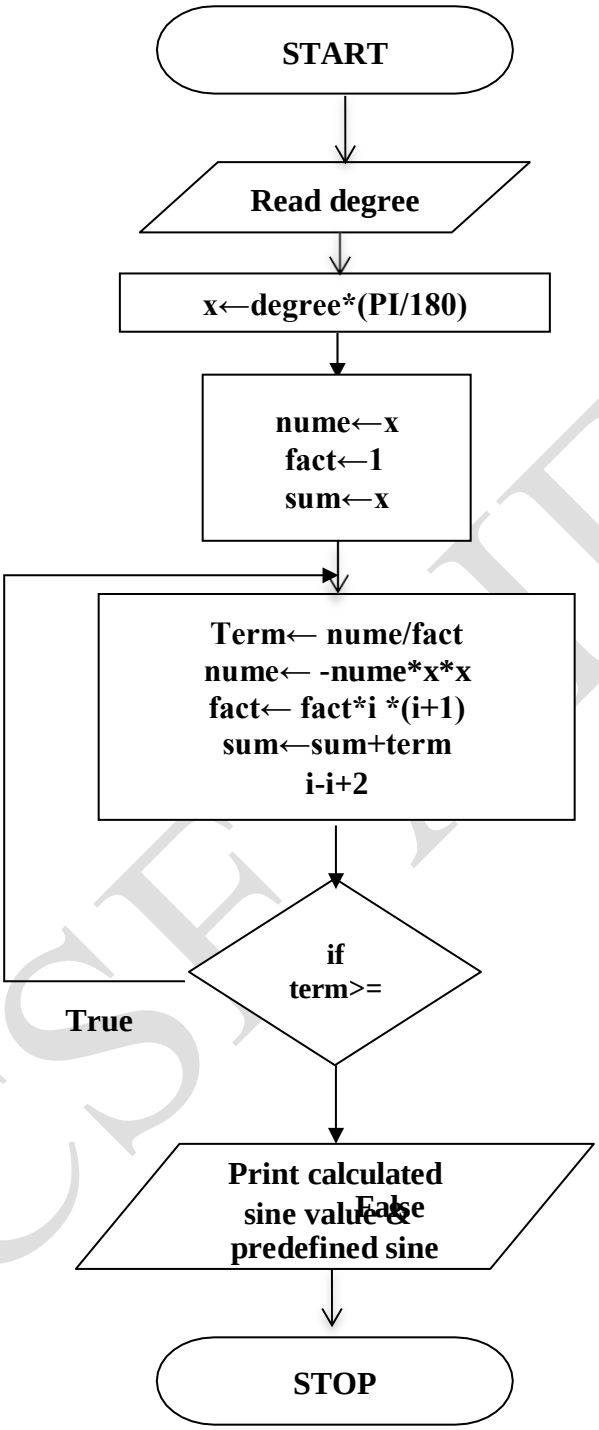
PROGRAM 6

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

ALGORITHM

Algorithm: Sine_series
Step 1:[Start the algorithm]
Start
Step 2:[Read the value of x]
Read x
Step 3:[Initialize the variables]
x1←x;
x←x*(3.142/180.0)
Step 4;[Read the value of no of terms]
Read n
Step 5:[Initialize the variables]
term←x;
sinx←term;
Step 6:[Repeat the following steps for n=1 to n terms
den←2*n*(2*n+1)
term←-term*x*x/den;
sum=sum+term;
Step 7:Print the result
sum of sine series=sum
sum using library function=sin(x)
Step 8: Stop

FLOWCHART



Program:

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#define PI 3.142

void main()
{
        int i,degree;
        float x,sum=0,term,nume,fact;
        clrscr();
        printf("Enter value of degree \n");
        scanf("%d",&degree);
        x=degree*(PI/180);
        nume=x;
        fact=1;
        i=2;
        do
        {
                term=nume/fact;
                nume=-nume*x*x;
                fact=fact*i*(i+1);
                sum=sum+term;
                i=i+2;
        }while(fabs(term)>=0.00001);
        printf("The sine of %d is %.3f",degree,sum);
        printf("Using inbuilt function sin(%d) is %.3f",degree,sin(x));
         getch();
}
```

OUTPUT:

.....

```
Enter value of degree
0
The sine of 0 is 0.000
Using inbuilt function sin (0) is 0

Enter value of degree
60
The sine of 60 is 0.866
Using inbuilt function sin(60) is 0.866

Enter value of degree
30
The sine of 30 is 0.500
Using inbuilt function sin(30) is 0.500
```

PROGRAM 7

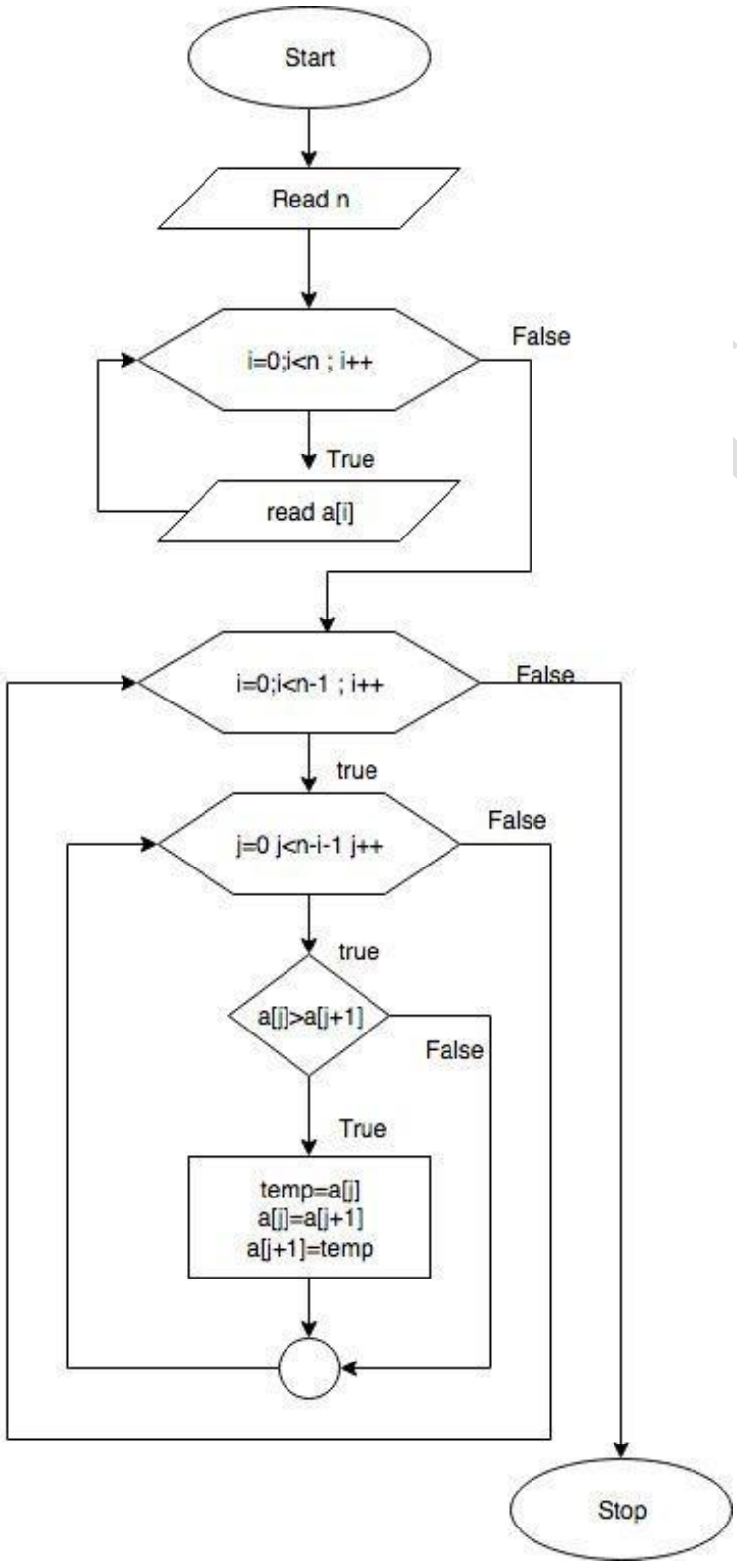
Sort the given set of N numbers using Bubble sort

ALGORITHM

Algorithm: Bubble sort

Step 1: [Start of the algorithm]
Start
Step 2: [Read the size of the array]
read n
Step 3: [Read the array elements]
Repeat for i=0 to n-1
read a[i]
Step 4: [Print the given array]
Repeat for i=0 to n-1
print a[i]
Step 5: Repeat through step 5 for i←1 to n-1
Repeat for j←0 to n-i
if(a[j]>a[j+1])
temp←a[j]
a[j] ←a[j+1]
a[j+1]←temp
Step 5: [Print the sorted array]
Repeat for i ←0 to n-1
print a[i]
Step 6 :[terminate the algorithm]
Stop

FLOWCHART


Start
Read n
i=0;i<n ; i++
False
True
read a[i]
i=0;i<n-1 ; i++
False
true
j=0 j<n-i-1 j++
False
true
a[j]>a[j+1]
False
True
temp=a[j]
a[j]=a[j+1]
a[j+1]=temp
Stop

Program:

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

void main()
{
        int a[100],n,i,j,temp;
        clrscr();

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

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

        printf("The Input array is\n");
        for(i=0;i<n;i++)
        {
                printf("%d\t",a[i]);
        }
        for(i=0;i<n-1;i++)
        {               for(j=0;j<n-i-1;j++)
                        {       if(a[j]>a[j+1])
                                {
                                        temp=a[j];
                                        a[j]=a[j+1];
                                        a[j+1]=temp;
                                }
                        }

        }

        printf("\nThe sorted array is\n");
        for(i=0;i<n;i++)
        printf("%d\t",a[i]);
        getch();
}
```

OUTPUT:

.....

Run1:

```
Enter the number of elements
4
Enter 4 elements of array
87
100
20
3

The input array is

87      100     20      3

The sorted array is

3       20      87      100
```

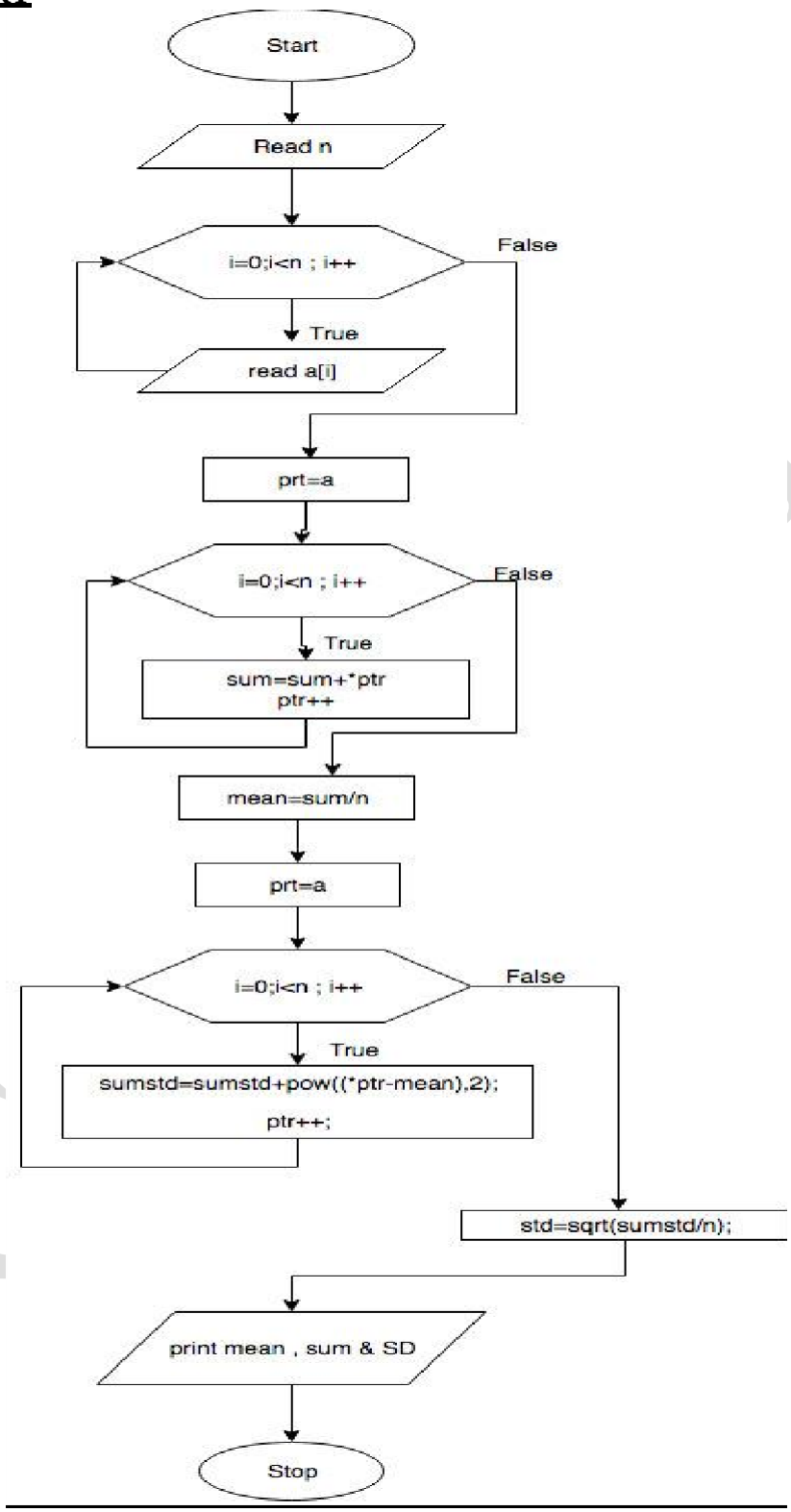
PROGRAM 10

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

ALGORITHM

Algorithm Mean
Step 1: Start
Step 2: read n
Step 3: read for i=0 to n-1
    Read a[i]
Step 4: prt=a
Step 5: repeat for i=0 to n-1
    sum = sum+*prt
    prt++
Step 6: mean=sum/n
Step 7: ptr=a
Step 8: repeat for i=0 to n-1
    sumstd=sumstd+pow((*ptr-mean),2)
        ptr++
Step 9: std=sqrt(stdsum/n)
Step 10: print mean sun and standard deviation
Step 11: Stop

FLOWCHART


Start
Read n
i=0;i<n ; i++
False
True
read a[i]
prt=a
i=0;i<n ; i++
False
True
sum=sum+*ptr
ptr++
mean=sum/n
prt=a
i=0;i<n ; i++
False
True
sumstd=sumstd+pow((*ptr-mean),2);
ptr++;
std=sqrt(sumstd/n);
print mean , sum & SD
Stop

Program:

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

void main()
{
        float a[10],*ptr,mean,std,sum=0,sumstd=0;
        int n,i;
        printf("Enter the number of elements\n");
        scanf("%d",&n);
        printf("Enter array elements\n");
        for(i=0;i<n;i++)
        {
                scanf("%f",&a[i]);
        }
        ptr=a;
        for(i=0;i<n;i++)
        {
                sum=sum+*ptr;
                ptr++;
        }
        mean=sum/n;
        ptr=a;
        for(i=0;i<n;i++)
        {
                sumstd=sumstd+pow((*ptr-mean),2);
                ptr++;
        }
        std=sqrt(sumstd/n);
        printf("Sum=%.3f\t",sum);
        printf("Mean=%.3f\t",mean);
        printf("Standard Deviation =%.3f\t",std)
        getch();
}
```

OUTPUT:

.....

```
Enter the number of elements
5
Enter array elements
1
2
3
4
5
Sum=15.000
Mean=3.000
Standard Deviation =1.414
```

```
RUN 2:
Enter the number of elements
4
Enter array elements
10.5
25.25
30.56
9.5
Sum=75.810
Mean=18.952
Standard Deviation =9.154
```

PROGRAM 8

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

ALGORITHM

Algorithm strings

Step 1: Start

Step 2: read string s1 & s2

Step 3: [call function strlength()]

i.e length =strlength

Step 4: Display length 1 and length 2
Step 5: [call function compare string]
if(compare_ string(s1,s2)==0)
print" equal strings"
else
print"unequl strings"
Step 6: call function concatenate
if(concatenate(s1,s2))
Step 7: print"concatenate string"
Step 8: Stop

Algorithm String_length()
Step 1: Start
Step 2: repeate step 2 through while(s1[i]!='\0')
i++
return i
end while
Step 3: Stop

Algorithm String_compare()
Step 1: Start
Step 2: repeatr step 2 trough while(s1[i]==s2[i])
if(s[i]==’\0’|| s2[i]==’\0’)
break
i++
end while

Step 3: if(s[i]==’\0’ && s2[i]==’\0’)
return 0
else
return 1
Step 4: Stop

Algorithm String_concatenate()
Step 1: Start
Step 2: Initilize i=0
Step 3: repeate step 3 through while (s1[i]!=’\0’)
i++
end while
Step 4: initialize j=0
Step 5: repeate step 5 through while(s2[i]1=’\0’)
s1[i]=s2[j]
i++ j++
end while
Step 6: s1[i]=’\0’
Step 7: Stop

PROGRAM

```
#include <stdio.h>
 int compare_strings(char [], char []);
void concatenate(char [], char []);
int string_length(char []);
 int main()
{
          char s1[1000], s2[1000];
         printf("Input a string1\n");
         gets(s1);
         printf("Input a string2\n");
          gets(s2);
          int length1 = string_length(s1);
          int length2 = string_length(s2);
         printf("Length of %s = %d\n", s1, length1);
          printf("Length of %s = %d\n", s2, length2);
          if (compare_strings(s1, s2) == 0)
          printf("Equal strings.\n");
```

```
        else
        printf("Unequal strings.\n");
       concatenate(s1, s2);
        printf("String obtained on concatenation: \"%s\"\n", s1);
        return 0;
}
/* string length function*/
 int string_length(char s1[])
{
        int c = 0;
        while (s1[c] != '\0')
         c++;
        return c;
}
/*string comparision*/
int compare_strings(char s1[], char s2[])
{
       int c = 0;
       while (s1[c] == s2[c])
       {
              if (s1[c] == '\0' || s2[c] == '\0')
              break;
              c++;
        }
        if (s1[c] == '\0' && s2[c] == '\0')
        return 0;
       else
        return 1;
}
/* string concatenation*/
void concatenate(char s1[], char s2[])
{
        int c, d;
        c = 0;
        while (s1[c] != '\0')
       {
              c++;
         }
        d = 0;
        while (s2[d] != '\0')
       {
             s1[c] = s2[d];
             d++;
             c++;
        }
        s1[c] = '\0';
}
```

PROGRAM 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.

ALGORITHM

Algorithm STUDENT

Step 1: Start
Step 2: create the structure student with fields usn, name and marks and structure variable s[10].
Step 3: Initialize the variables countav=0, countbv=0
Step 4: read the number of students 'n'
Step 5: read the value of usn name & marks for n number of students using structure variable s[i] for i=0 to n-1
Step 6: Display the details
Step 7: repeat for i=0 to n-1
    Compute the sum of students
Step 8: compute the average
Step 9: repeat for i=0 to n-1
    if(s[i].marks >= average)
    printf" total number of students above average"
    else
    printf" total number of students below average"
Step 10: Stop

PROGRAM

```
#include <stdio.h>
struct student
{
        char usn[50];
        char name[50];
        int marks;
} s[10];
void main()
{
        int i,n,countav=0,countbv=0;
        float sum,average;
        printf("Enter number of Students\n");
```

```
scanf("%d",&n);
printf("Enter information of students:\n");
// storing information
for(i=0; i<n;i++)
{
        printf("Enter USN: ");
        scanf("%s",s[i].usn);
        printf("Enter name: ");
        scanf("%s",s[i].name);
        printf("Enter marks: ");
        scanf("%d",&s[i].marks);
        printf("\n");
}
printf("Displaying Information:\n\n");
// displaying information
for(i=0; i<n; i++)
{
        printf("\nUSN: %s\n",s[i].usn);
        printf("Name: %s\n ", s[i].name);
        printf("Marks: %d",s[i].marks);
        printf("\n");
}
for(i=0;i<n;i++)
{
        sum=sum+s[i].marks;
}
average=sum/n;
printf("\nAverage marks: %f",average);
for(i=0;i<n;i++)
{
        if(s[i].marks>=average)
        countav++;
        else
        countbv++;
}
printf("\nTotal No of students above average= %d",countav);
printf("\nTotal No of students below average= %d",countbv);
}
```

OUTPUT:

.....

```
Enter number of Students
2
Enter USN: 11
Enter name: arun
Enter marks: 23

Enter USN: 12
Enter name: ram
Enter marks: 25

Displaying Information:
USN:11
Name: arun
Marks: 23
USN:12
Name: ram
Marks: 25

Average marks: 24

Total No of students above average= 1
Total No of students below average= 1
```