Program 1:

Design and implement the following:

a) **To read a *year* as an input and find whether it is leap year or not. (Also consider end of the centuries).**
b) **To find the largest of three numbers using *ternary operator***

1. a)

```
#include<stdio.h>

void main()

{

        int year;

        printf("Enter the Year \n");

        scanf("%d",&year);

        if((year%4==0)&&(year%100!=0)||(year%400==0))

        {

                printf ("%d is a leap year”, year);

        }

        else

        {

                printf ("%d is not a leap year”, year);

        }

getch();

}
```

OUTPUT:

...

Run 1

```
Enter the year

2012

2012 is leap year
```

Run 2

```
Enter the year
```

```
1900
1900 is not leap year
```

1. b)

```
# include <stdio.h>
 void main()
 {
 int a, b, c, big ;
 printf("Enter three numbers :") ;
 scanf("%d %d %d", &a, &b, &c) ;
 big = a > b ? (a > c ? a : c) : (b > c ? b : c) ;
 printf("\nThe biggest number is : %d", big) ;
 getch();
}
```

OUTPUT:

```
Enter three numbers:
9
3
2
The biggest number is :9
```

Program 1.a:

Algorithm:

Algorithm Leapyear [This algorithm takes year as input and finds whether it is a leap year or not]

Steps:

```
1. [Initialization]
        Start
2. [Input]
        Read year
3. [Check whether the given year is leap year or not and print the result]
        if((year%4==0)&&(year%100!=0)||(year%400==0)) then
                Print -It is a leap year
        else
                Print -It is not a leap year
        end if
4. [Finished]
        Stop
```

Program 1.b:

Algorithm:

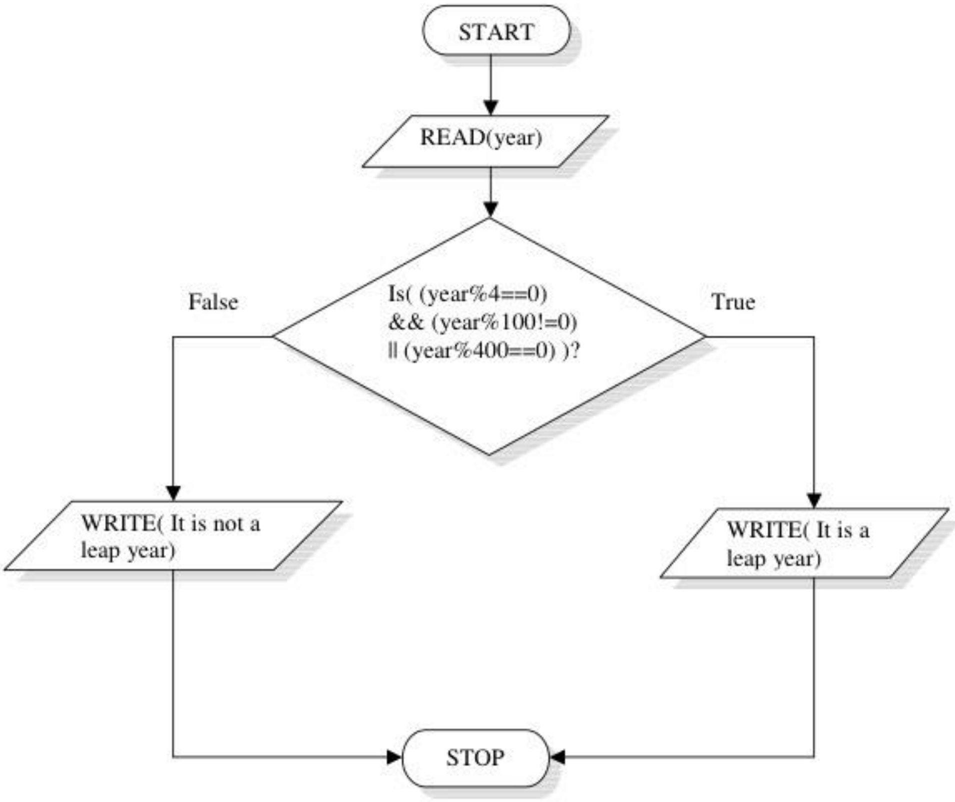
Algorithm Largest of three numbers [This algorithm accepts three numbers as input and displays the largest.]

Steps:

```
1. [Initialization]
        Start
2. [Input]
        Read three numbers
3. [Compute the largest of three numbers]
        big = a > b ? (a > c ? a : c) : (b > c ? b : c) ;
4. [Output]
        Print -the largest number
5. [Finished]
        Stop
```

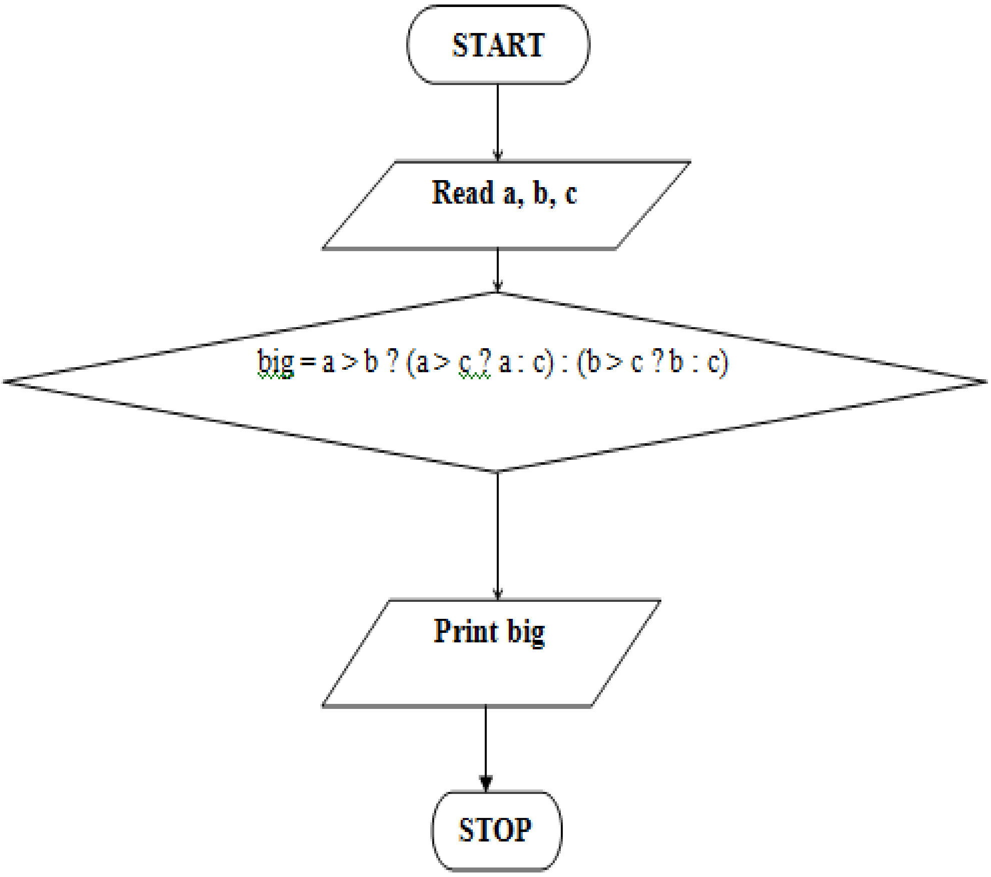
Program 1a)

Flowchart



Program 1b)

Flowchart



Program 2:

Design and develop a C program that accepts three coefficients (a, b, and c) of a Quadratic equation (ax2+bx+c=0) as input and compute all possible roots and print the possible roots for a given set of coefficients. Also print the messages in case of Zero valued coefficient/s.

```
#include<stdio.h>

#include<math.h>

void main()

{

        float a,b,c,d,rpart,ipart,root1,root2;

        clrscr();

        printf("enter three  co-efficients\n");

        scanf("%f%f%f",&a,&b,&c);

        if(a==0 && b==0)

        {

                printf(“invalid inputs”);

                getch();

        }

        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:

....

Run 1

```
Enter three co-efficients

1

2

3

The roots are imaginary

The first root root1=-1.000+i1.414

The second root root2=-1.000-i1.414
```

Run 2

```
Enter three co-efficients

1

-5

6

The roots are real and distinct

The roots are root1=3.000 and root2=2.000
```

Run 3

```
Enter three co-efficients

1

2

1

The roots real and equal

The roots are root1=-1.000 and root2=-1.000
```

Program 2:

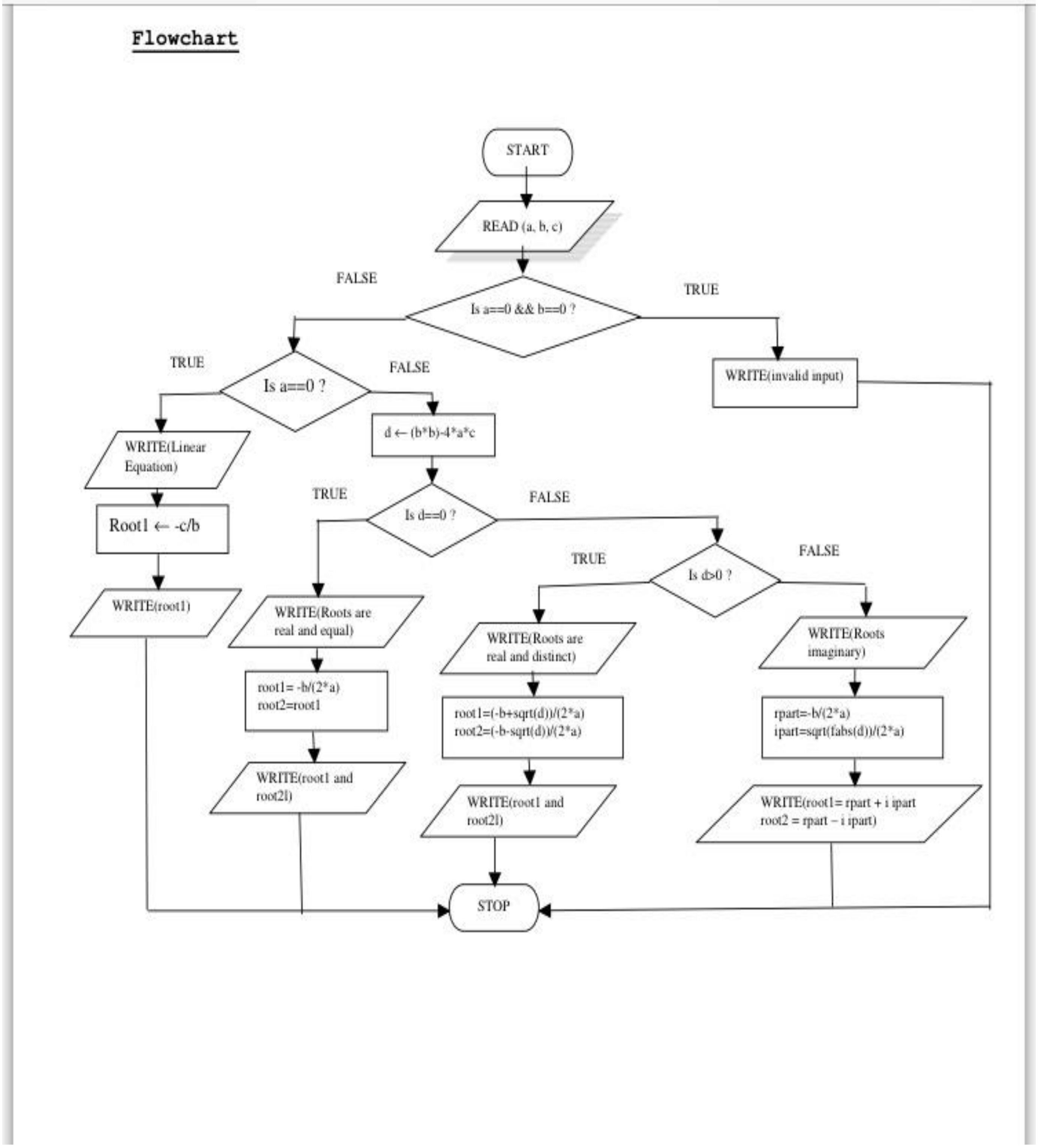
Algorithm Quadratic Equation [This algorithm takes three coefficients of quadratic equation and finds the roots for the same]

Steps:

```
1. [Initialization]
        Start
2. [Input]
        Read a, b, c
3. [For zero coefficients]
        if(a==0 && b==0) then
                Print- Invalid inputs
        end if
        else
         if(a==0) then
                Print -Linear Equation
                root1=-c\b;
        end if
else
4. [For non zero coefficients-Compute disc]
         d=(b*b)-(4*a*c);
5. [If d is zero- roots are real and equal]
        if(d==0) then
                root1= -b/(2*a);
                root2=root1;
        end if
   [If d is greater than zero-roots are real and distinct]
        if(d>0) then
                root1=(-b+sqrt(d))/(2*a);
                root2=(-b-sqrt(d))/(2*a);
          end if
```

```
 [If d is less than zero-roots are imaginary]
      if(d<0) then
            rpart=-b/(2*a);
            ipart=sqrt(fabs(d))/(2*a);
            root1=rpart+ipart;
            root2=rpart-ipart;
      end if
6. [Output]
      Print the roots
7. [Finished]
      Stop
```

Program 2)


Flowchart
START
READ (a, b, c)
FALSE
TRUE
Is a==0 && b==0 ?
WRITE(invalid input)
TRUE
Is a==0 ?
FALSE
WRITE(Linear Equation)
Root1 ← -c/b
WRITE(root1)
d ← (b*b)-4*a*c
TRUE
Is d==0 ?
FALSE
WRITE(Roots are real and equal)
root1= -b/(2*a)
root2=root1
WRITE(root1 and root2l)
TRUE
Is d>0 ?
FALSE
WRITE(Roots are real and distinct)
root1=(-b+sqrt(d))/(2*a)
root2=(-b-sqrt(d))/(2*a)
WRITE(root1 and root2l)
WRITE(Roots imaginary)
rpart=-b/(2*a)
ipart=sqrt(fabs(d))/(2*a)
WRITE(root1= rpart + i ipart
root2 = rpart – i ipart)
STOP

Program 3:

Design and develop an algorithm to find the reverse of an integer number NUM and check whether it is PALINDROME or NOT. Implement a C program for the developed algorithm that takes an integer number as input and output the reverse of the same with suitable messages. Ex: Num: 1234, Reverse: 4321, Not a Palindrome.

```
#include<stdio.h>
void main()
{
        int n,rev=0,temp,digit;
        printf("enter an integer number\n");
        scanf("%d",&n);
        temp=n;
        while(n!=0)
        {
                digit=n%10;
                rev=rev*10+digit;
                n=n/10;
        }
        printf(“givin number is :%d\n”,temp);
        printf(“it’s revers is:%d\n”,rev);
        if(rev==temp)
                printf("the number is a palindrome\n");
        else
                printf("the number  is not a palindrome\n");
getch();
}
```

OUTPUT:

..

Run 1

```
Enter an integer number

3443

Given number is: 3443

It’s reverse is 3443

The number is a palindrome
```

Run 2

```
Enter an integer number

5678

Given number is: 5678

It’s reverse is 8765

The number is not a palindrome
```

Program 3:

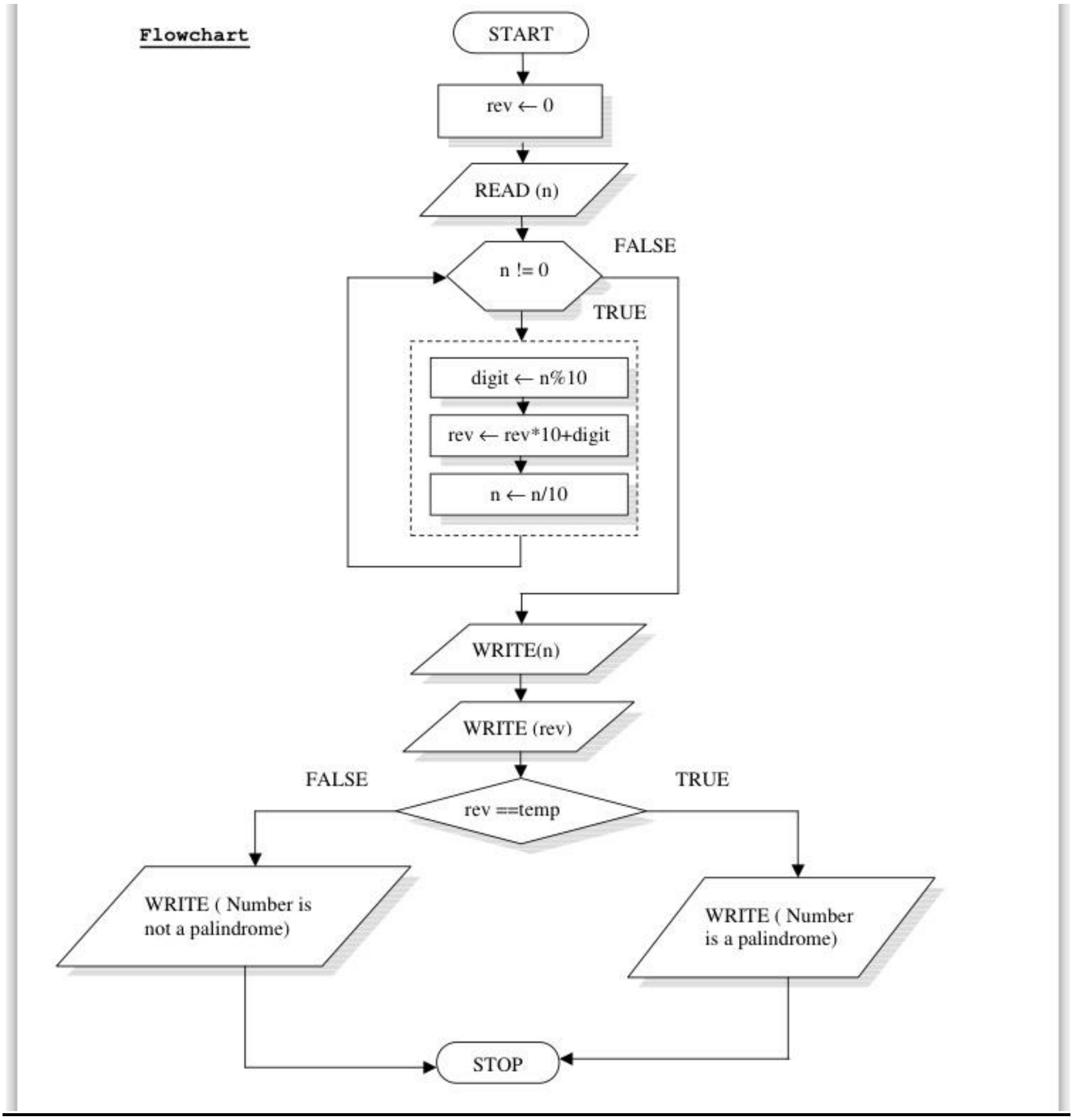
Algorithm:

Algorithm Palindrome [This algorithm takes an integer input and output the reverse for the same. It also checks whether it is palindrome or not.]

Steps:

```
1. [Initialization]
       Start
2. [Input]
       Read n
3. [Set number n to a variable temp]
         temp←n
4. [Iterate until n not equal to zero]
     while (n!=0) do
             digit←n%10
             rev=digit+10*rev
             n=n/10
     end while
5. [Print reverse of the number]
       Print- rev
6. [Check if original number and reverse number is same. If it is, number is
             palindrome. Otherwise, number is not palindrome]
       if (rev==temp) then
             Print -palindrome
       else
             Print -not a palindrome
       end if
7. [Finished]
       Stop.
```

Program 3)


Flowchart
START
rev ← 0
READ (n)
n != 0
FALSE
TRUE
digit ← n%10
rev ← rev*10+digit
n ← n/10
WRITE(n)
WRITE (rev)
FALSE
rev ==temp
TRUE
WRITE (Number is not a palindrome)
WRITE (Number is a palindrome)
STOP

Program 4:

Develop an algorithm, implement a C program that reads N integer numbers and arrange them in ascending order using *Bubble Sort.*

```
#include<stdio.h>

void main()

{

        int a[100], n, i , j, temp;

        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:

..

```
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 4:

Algorithm:

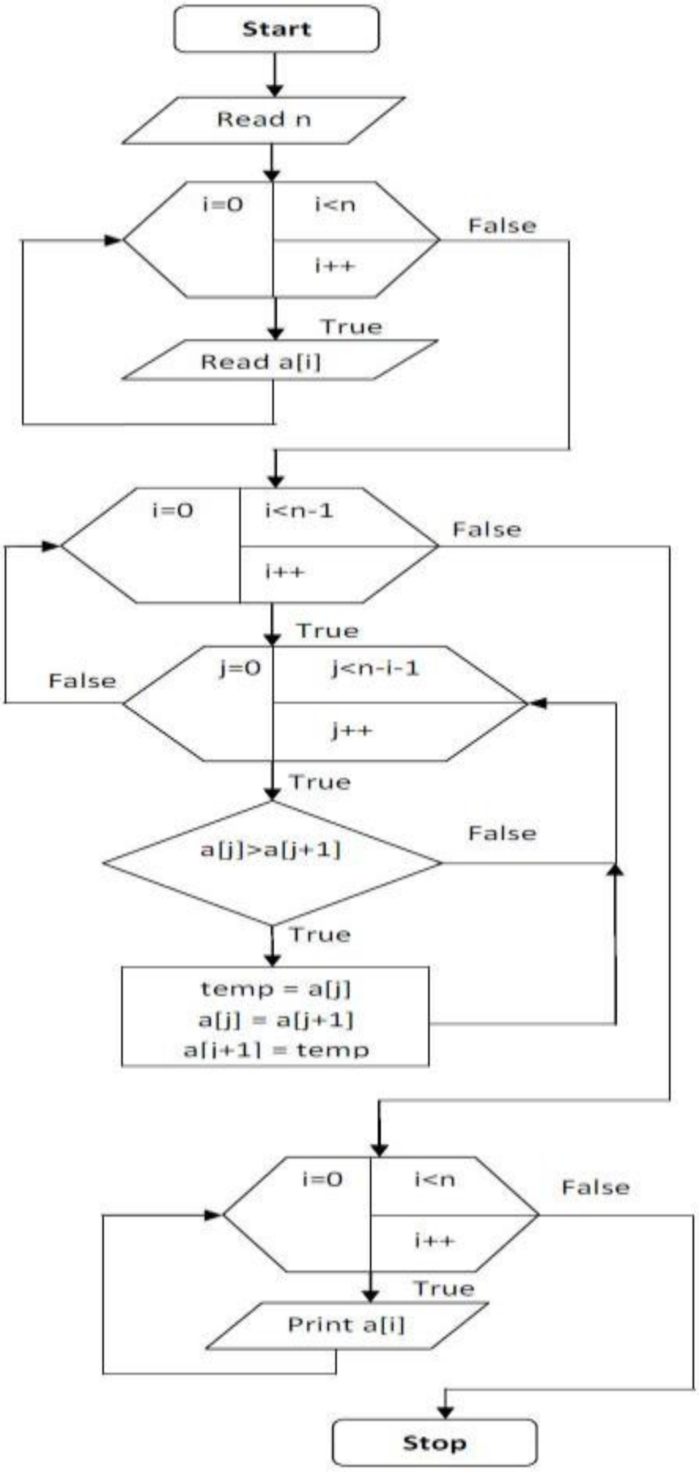
Algorithm Bubble Sort [This algorithm takes input as a list of unsorted numbers and arranges them in ascending order using Bubble Sort method]

Steps:

```
1. [Initialize]

            Start

2. [Input]

            Read n

3. [Input unsorted array]

            Read elements to array a[]

4. Print elements of array a[]

5. [Iterate array a[] in two loops. Outer loop gives number of pass. Inner loop does
   swap task. In each pass, compare each pair of adjacent items. If formal elements are
   greater than latter one, swap them.]

        for each value i in array a[i] to n do

            for each value j in array a[j] to n-1 do
                    [Compare each pair of adjacent elements]

                    if (a[j] > a[j+1])then

                    [Swap these elements using temp variable]

                        temp ← a[j]

                        a[j] ← a[j+1]

                        a[j+1]  ← temp

                    end if
             end for
         end for

6. [Output]
        Print array with sorted elements

7. [Finished]

        End.
```

Program 4)

Flowchart



Program 5:

Design and Develop a C program that reads N integer numbers and search a key element using Binary search technique and execute a C program to search a Name in a list of names using *Binary search* Technique.

```
#include<stdio.h>

#include<conio.h>

void main()

{

int a[100], n, i, key, low, mid, high;

clrscr();

printf("enter the number of elements\n");

scanf("%d",&n);

printf("enter %d elements in ascending order\n",n);

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

scanf("%d",&a[i]);                    //read n array elements in ascending order

printf("enter an element to search\n");

scanf("%d",&key);                           //read key element to search

low=0;

high=n-1;

mid= (low+high)/2;                          //compute mid element position

while(low<=high && key!=a[mid])

{

         if (key<a[mid])

                high=mid-1;

        else

                low=mid+1;

        mid= (low+high)/2;

}
if(a[mid]==key)

printf("%d is found at position %d\n",key,mid+1);

else

printf("key not found\n");

getch();

}
```

OUTPUT:

..

Run 1:

```
Enter the number of elements:
6
Enter 6 elements in ascending order
4 7 9 11 15 18
Enter an element to search
15
Key found at position 5
```

Run 2:

```
Enter the number of elements:
4
Enter 4 elements in ascending order
2 5 7 9
Enter an element t search
10
Key not found
```

Program 5:

Algorithm:

Algorithm Binary Search [This algorithm takes input as sorted numbers and searches for a key element in the given list of array]

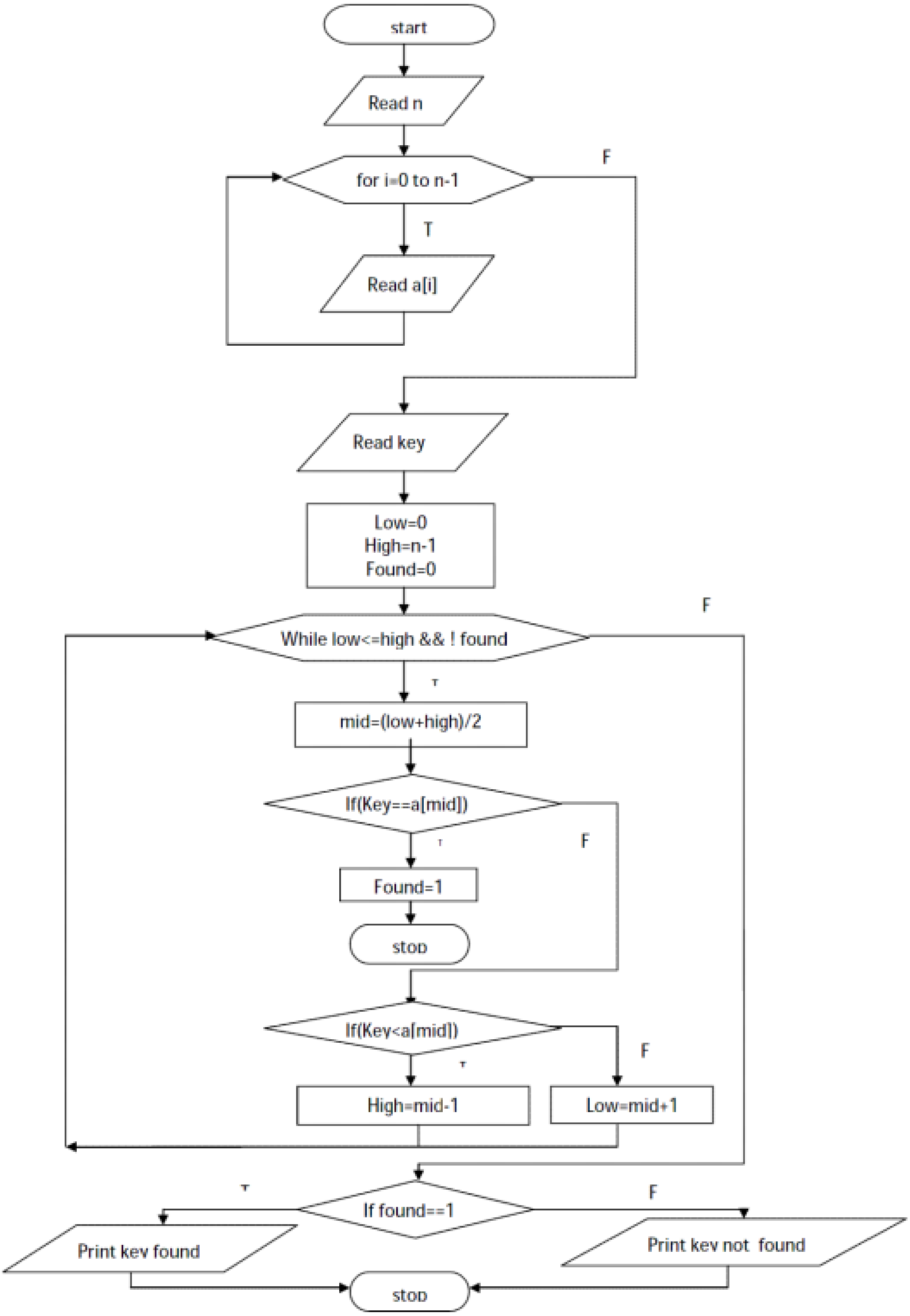
Steps:

```
1. [Initialize]
            Start
2. [Input]
            Read n
3. [Input sorted array]
            Read elements to array a[]
4. [Input the item to be searched]
            Read key
5. [Initialization]
            low=0
            high=n-1
6. [Search the key using binary search method]
        mid=(low+high)/2
        while (low<=high && key!=a[mid]) do
        {
            if (key<a[mid])
                high=mid-1
            else
                low=mid+1
            end if
            mid=(low+high)/2
        }
        end while
7. [Check the position of the key]
        if(a[mid]==key)
```

```
                Print “key found at mid+1 position”
        else
                Print “key is not found”
        end if
8. [Finished]
        End.
```

Program 5)

Flowchart



Program 6:

Implement a C program that reads two matrices A (m x n) and B (p x q) and Compute product of matrices A and B. Read matrix A and matrix B in row major order and in column major order respectively. Print both the input matrices and resultant matrix.

```
#include<stdio.h>

void main()

{

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

        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");

                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]);

                 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

...

Run 1

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

Run 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:

Algorithm:

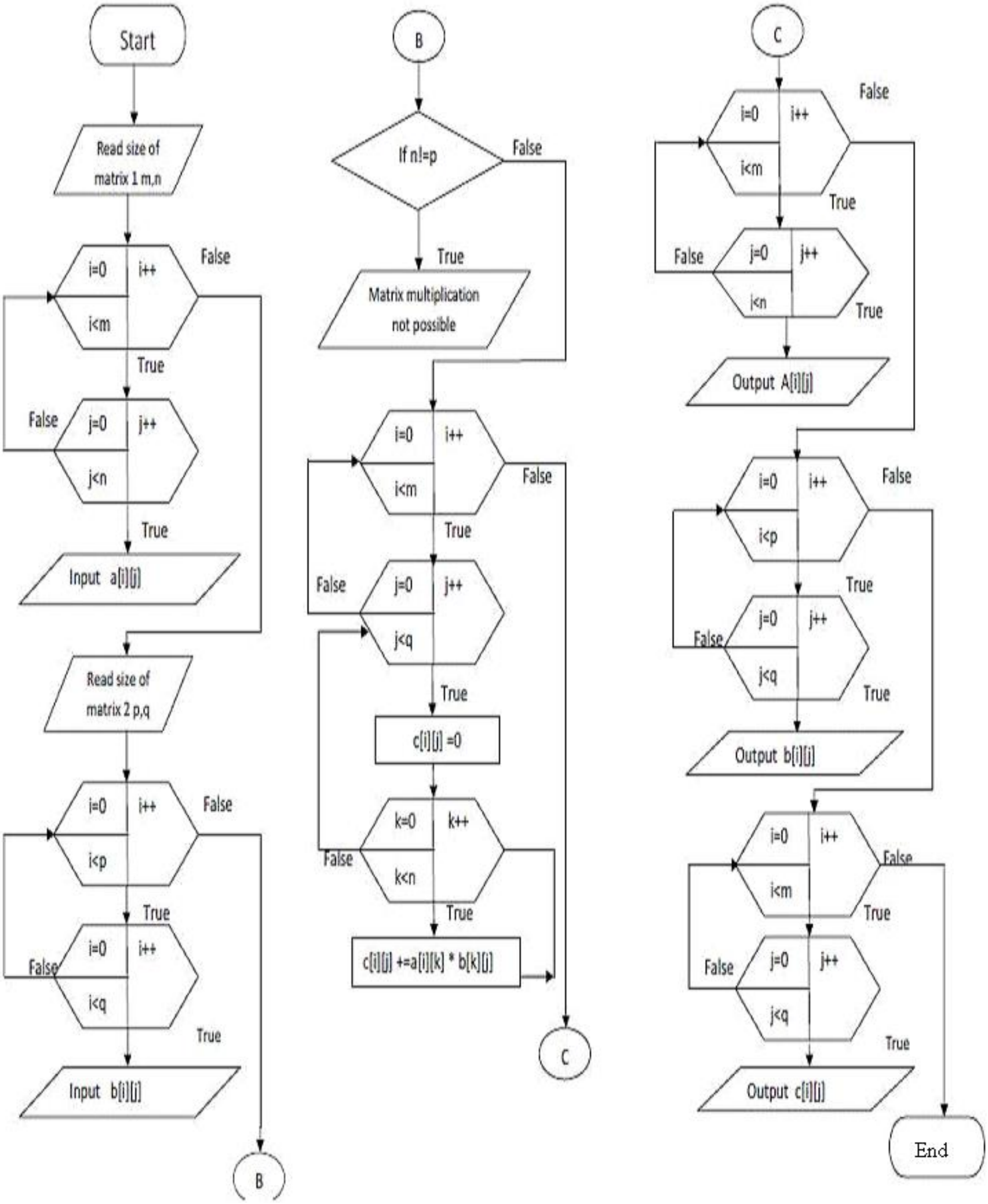
Algorithm Matrix Multiplication [This algorithm takes input as two matrices and displays the result as product of two matrices]

Steps:

```
1. [Initialization]
        Start
2. [Input]
        Read Order of Matrix A as m (rows) and N (columns)
        Matrix B as p (rows) and q (columns)
4. [Check for compatibility]
   if(n!=p) then
        Print Matrix multiplication is not possible goto step 8
   end if
   else
5. [Input elements of matrix A and matrix B]
        Read the elements of matrix A and matrix B by using loops
6. [Calculate product of two matrices]
         c[i][j]=c[i][j]+a[i][k]*b[k][j];
7. [Output]
        Print the resultant Matrix C
8. [Finished]
        Stop
```

Program 6)

Flowchart



Program 7:

Design and develop a C program to implement the following operations. Display the results after every operation. (Do not Use library function)

I. **STRING S1 = “Dayananda”**
II. **STRING S2 = “Sagar”**
III. **STRING S3 = “Dayananda Sagar”**

```
#include<stdio.h>

#include<conio.h>

void main()

{

 char STR1[100],STR2[100],STR3[100];

 int i=0,j=0,k=0;

 clrscr( );

printf("Enter the String 1\n");

gets(STR1);

 printf("Enter the String 2\n");

 gets(STR2);

 while(STR1[i]!='\0')

            {

                        STR3[k]=STR1[i];

                        k++;

                        i++;

              }

STR3[k++]=' ';

while(STR2[j]!='\0')

              {
                  STR3[k]=STR2[j];
                  k++;
                  j++;
              }
STR3[count]='\0';
printf("\n String STR1=\t");
puts(STR1);
printf("\n String STR2=\t");
puts(STR2);
printf("\n String STR3=\t");
puts(STR3);
getch();
}
```

OUTPUT

..

Run1:

```
String S1 =         Dayananda
String S2 =         Sagar
String S3 =         Dayananda Sagar
```

Run2:

```
String S1 =         Engineering
String S2 =         Department
String S3 =         Engineering Department
```

<u>Program 7:</u>

<u>Algorithm:</u>

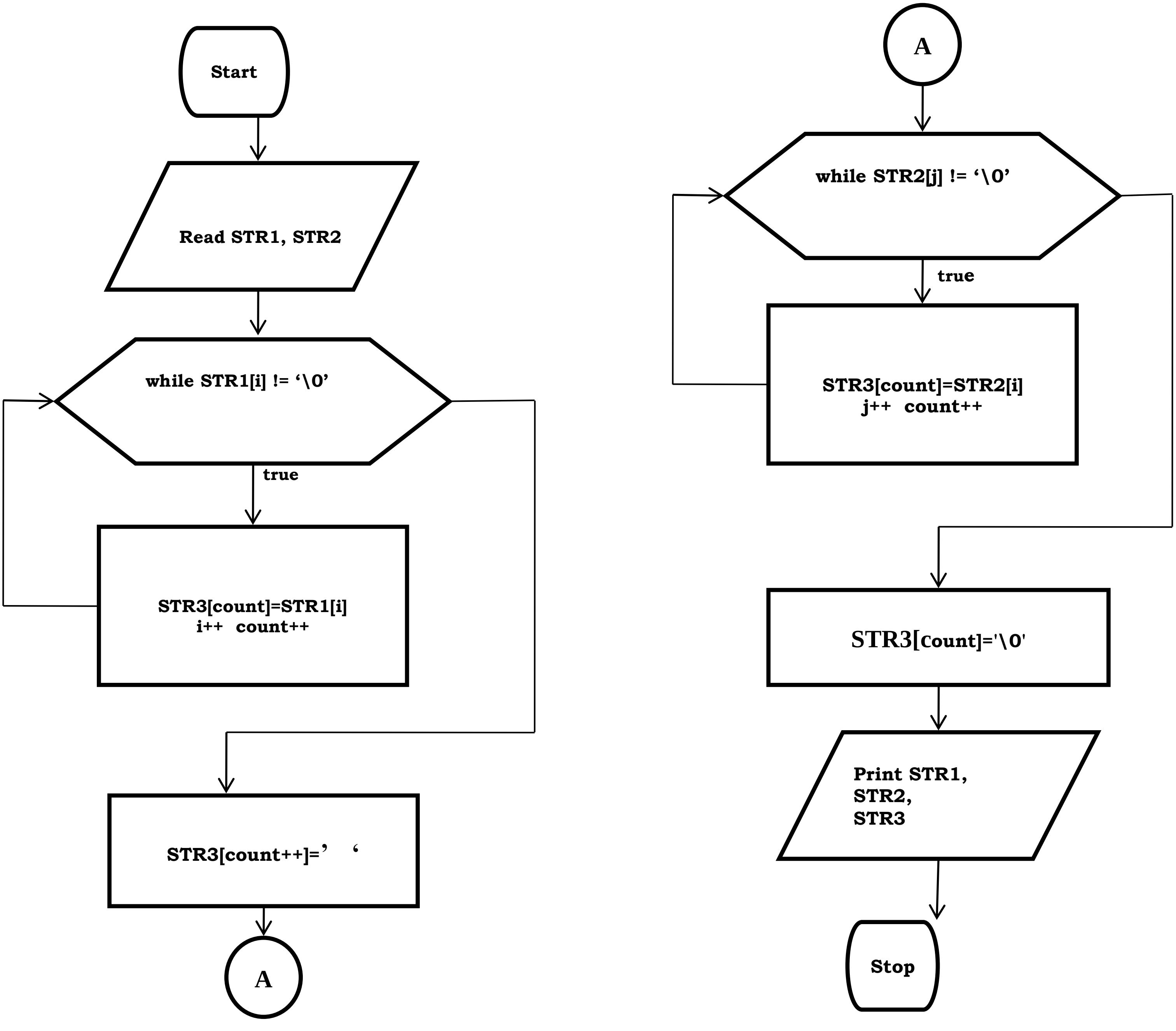
Algorithm String Operations [This algorithm takes input as two strings “Dayananda” and “Sagar” and displays the output as “Dayananda Sagar”]

Steps:

```
1. [Initialization]

        Start

2. [Input]

        Read STR1, STR2

3. [Initialization of indices]

        i=0, j=0, k=0

4. [Copy the string STR1 to STR3]
        while (STR1[ i ] != ‘\0’) do
            {
                STR3[k]=STR1[ i ]
                 k ++
                 i++
            }
         end while
5. [Insert space between 2 strings]
         STR3[k++]=’  ‘
6. [Copy the string STR2 to STR3]
          while (STR2[ j ] != ‘\0’)  do
            {
                STR3[k]     =   STR2[ j ]
                 k++
                j++
             }

          end while

7. [Terminate the String with end of Character]
             STR3[k] = ‘\0’
8. [Output]
                 Print-STR1, STR2, STR3

8. [Finished]

        Stop.
```

Program 7)

Flowchart



Program 8:

Develop a C function isprime(num) that accepts an integer argument and returns 1 if the argument is prime, 0 otherwise. Write a C program that invokes this function to generate prime numbers between the given ranges.

```
#include<stdio.h>

#include<conio.h>


int isprime(int); //function prototype


void main()

{

         int n1,n2,r,i;

        clrscr();

        printf("Enter the range n1 and n2 to find prime numbers\n");

        scanf("%d%d",&n1,&n2);  // read the inputs

        printf("The prime numbers between %d and %d are\n",n1,n2);


        for(i=n1;i<=n2;i++)

         {

                r=isprime(i);  //function call

                if(r==1)

                        printf("%d\t",i);

        }

        getch();

        }


        int isprime(int  x)         // function definition
```

```
{
  int i,c=0;
          for(i=1;i<=x;i++)
                    {
                              if(x%i==0)           //check number is completely divisible
                              c++;
                    }
  if(c==2)        //check number is divisible by 1 and itself
  return 1;
  else
  return 0;
 }
```

OUTPUT

…………………………………………………………

Run1:

```
Enter the range n1 and n2 to find prime numbers
10  20
The prime numbers between 10 and 20 are
11 13 17 19
```

Run2:

```
Enter the range n1 and n2 to find prime numbers
1 10
The prime numbers between 10 and 20 are
2  3 5 7
```

<u>Program 8:</u>

<u>Algorithm:</u>

Algorithm Prime Numbers [This algorithm accepts two integer values as ranges and prints the prime numbers within the given range]

Steps:

```
1. [Initialization]

        Start

2. [Input]

        Range n1 and Range n2

3. [Call the function and print the prime number]

        for (i=n1;i<=n2;i++) do

            {

                r=isprime(i);  //function call

                if (r==1) then

                    Print –i  //prime number

                end if

            }

        end for
4. [Finished]

        Stop
```

Algorithm for isprime(x)

Steps:

```
1. [Initialization]

        Start

2. [Initialize the count variable]

        Initialize count=0;


3. [Check the divisibility of the numbers]
```

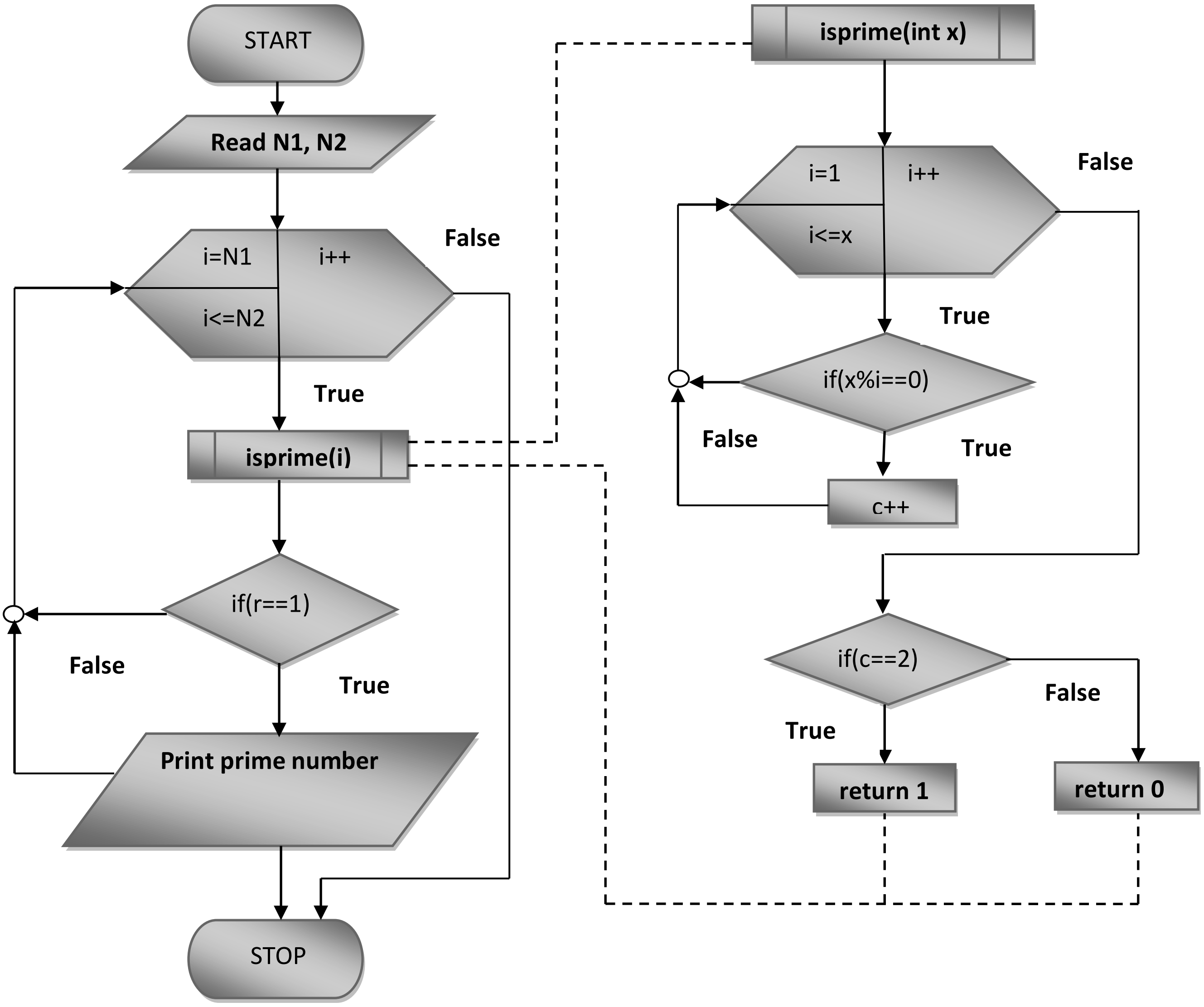
```
    for(i=1;i<=x;i++) do
        {
            if(x%i==0)  then        //check number is completely divisible
            c++;
            end if
        }
    end for
    if(c==2)   then          //check number is divisible by 1 and itself
        return(1);
    end if
    else
        return(0);
```

4. [Finished]

Stop

Program 8)

Flowchart



Program 9:

Design and develop a C program to create a structure called Employee to maintain a record of details using an array of structures with four fields (Emp_name, Emp_id, Emp_sal, Emp_age). Assume appropriate data type for each field. Print the Employee details in Tabular Format.

```
#include<stdio.h>

#include<conio.h>

        //Definition of an Employee structure with the necessary fields

struct employee

{

          int eid,eage;

          char ename[20];

          float esal;

}



        //typed definition gives aliasing for "struct employee" as "emp"

typedef struct employee emp;



void main()

        {

           emp e[50]; //e - Array of structure employee

           int n,i;  // n - for number of employees and i - a loop iterator

           clrscr();

            // Read the number of Employees


           printf("\n Enter the number of Employees: ");

           scanf("%d",&n);

           // Read the details of n -  employees
```

```
  for(i=0;i<n;i++)
  {
        printf("\n Enter the details of Employee : %d\n",i+1);
        printf("\n Enter eid:");
        scanf("%d",&e[i].eid);
        printf("\nEnter ename:");
        scanf("%s",e[i].ename);
        printf("\nEnter eage:");
        scanf("%d",&e[i].eage);
        printf("\nEnter esal:");
        scanf("%f",&e[i].esal);
  }

  //Display the details of n = eployees with proper formatting

  printf("\n**************************************************\n");
  printf("\nDetails of %d Employees are as follows\n",n);
  printf("\n**************************************************\n");
  printf("\nEmpid\tEmpname\tEmpage\tEmpsal\n");
  printf("\n------------------------------------------------\n");



 for(i=0;i<n;i++)
  {
        printf("\n%d\t%s\t%d\t%f",e[i].eid,e[i].ename,e[i].eage,e[i].esal);
```

```
       }
      printf("\n----------------------------------------------\n");


      getch();


    }
```

OUTPUT

..

```
Enter the number of Employees: 3

Enter the details of Employee : 1

Enter eid: 111

Enter ename: Akash

Enter eage: 21

Enter esal: 20000


Enter the details of Employee : 2

Enter eid: 222

Enter ename: Banu

Enter eage: 22

Enter esal: 30000


Enter the details of Employee : 3

Enter eid: 333

Enter ename: Chaithra

Enter eage: 23

Enter esal: 32000


******************************************************************
```

```
Details of 3 Employees are as follows
*************************************************************************

      Empid      Empname                  Empage        Empsal
      ---------------------------------------------------------------------------------
      111        Akash                    21            20000.000000
      222        Banu                     22            30000.000000
      333        Chaithra                 23            32000.000000
      ----------------------------------------------------------------------------------
```

Program 9:

Algorithm:

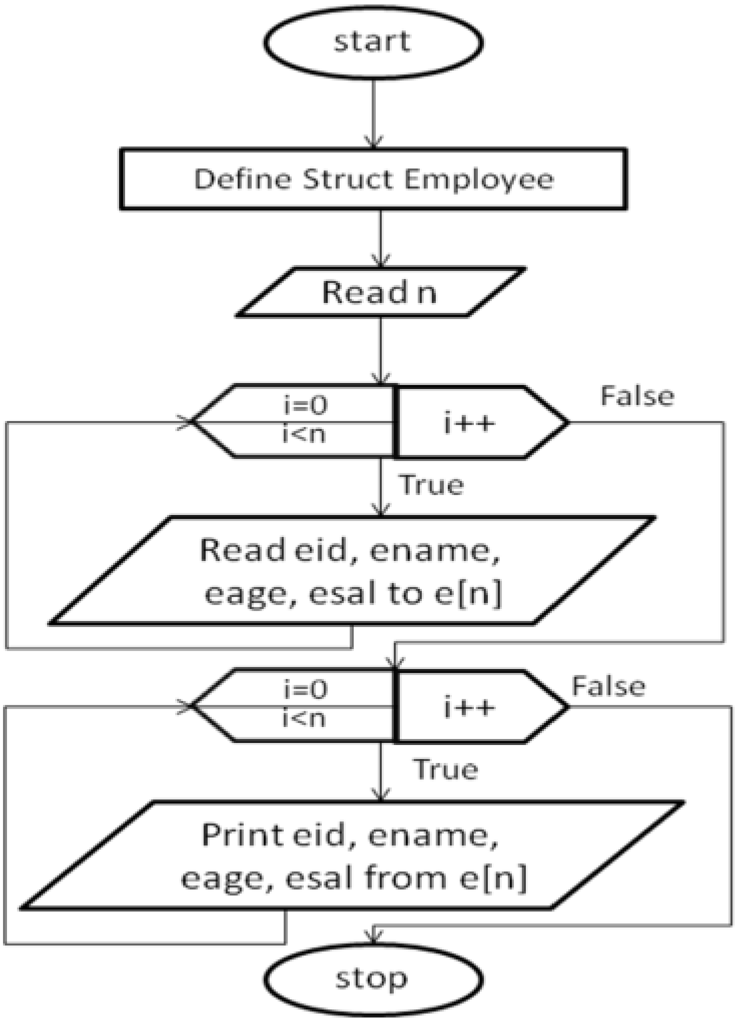
Algorithm Prime Numbers [This algorithm accepts two integer values as the range and print the prime numbers within the given range]

Steps:

```
1. [Initialization]
        Start
2. [Definition of the structure template]
3. [Create aliasing name for the structure using typedef. //for ease of use]
4. [Declare the array of structure variable as per the need]
5. [Read the number of Employees]
            Input n
6. [Input the details of employees]
        for(i=0;i<n;i++) do
            Read the details of each employee (eid, ename, eage, esal)
            and store the same in the array of structure created
        end for
7. [Print the header details of employees]
        for(i=0;i<n;i++) do
            Print the details of each employee (eid, ename, eage, esal) in
            a formatted way using the array of structure in which the
             details were stored.
        end for
8. [Finished]
        Stop
```

Program 9)

Flowchart


start
Define Struct Employee
Read n
i=0
i<n
i++
False
True
Read eid, ename, eage, esal to e[n]
i=0
i<n
i++
False
True
Print eid, ename, eage, esal from e[n]
stop

Program 10:

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.

```
#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");     //Input array elements

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

        scanf("%f",&a[i]);

ptr=a;

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

        {

                sum=sum+*ptr;         //compute sum

                ptr++;

        }

mean=sum/n;                     //compute mean

ptr=a;

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

        {
                sumstd=sumstd+pow((*ptr-mean),2);      //compute standard deviation

                ptr++;

        }

std=sqrt(sumstd/n);

printf("Sum=%.3f\n",sum);

printf("Mean=%.3f\n",mean);

printf("Standard Deviation =%.3f\n",std)

}
```

OUTPUT

...

Run 1:

```
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 10:

Algorithm:

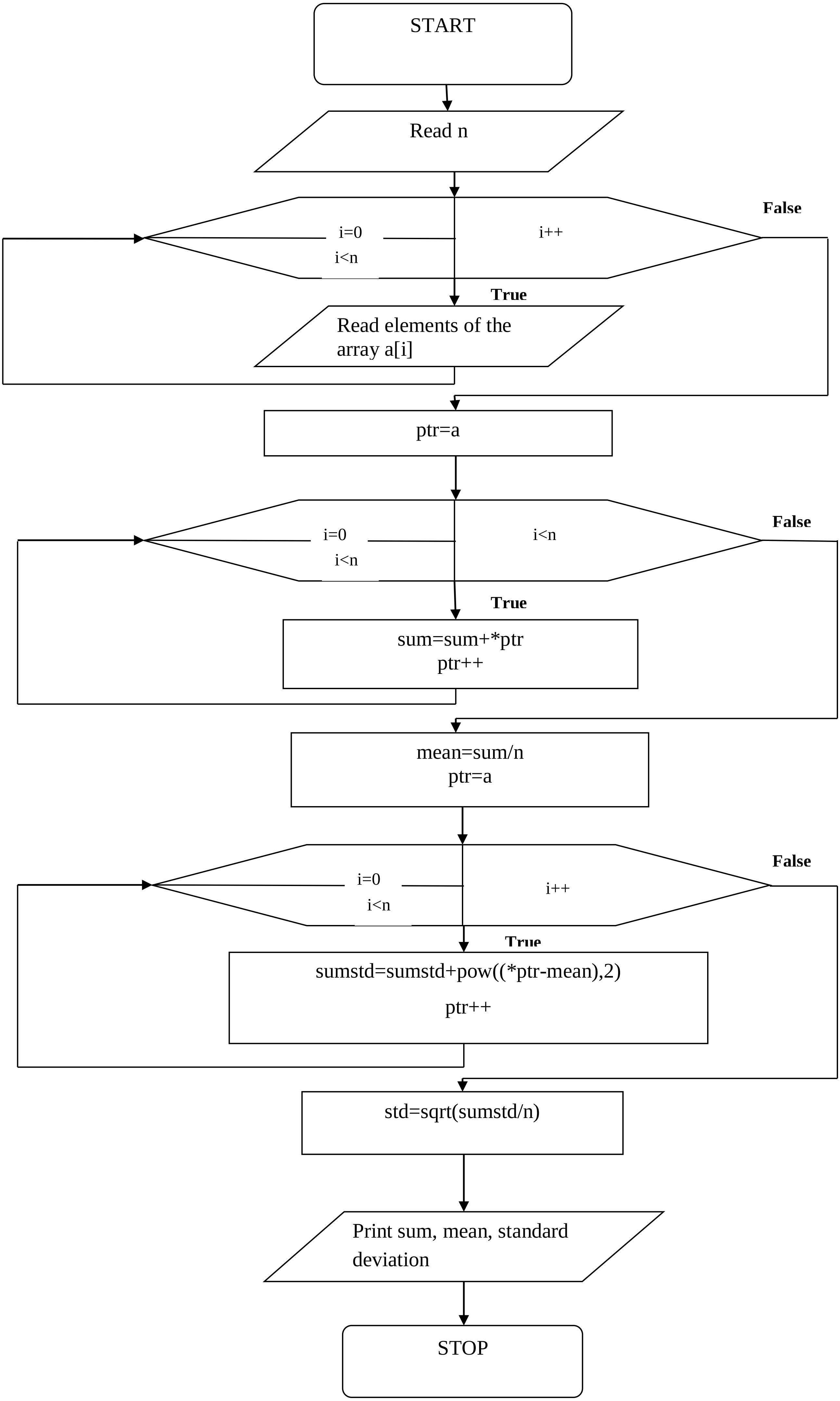
Algorithm-Computation of sum, mean, standard deviation of array elements [This algorithm takes array elements as input and outputs sum, mean, standard deviation for the given elements]

Steps:

```
1. [Initialization]
      Start
2. [Input]
      Read the array elements of an array a[]
3. [Set pointer to the array element 'a' of an array a[] ]
      ptr←a
4. [Compute sum of the array elements]
      sum=sum+*ptr
5. [Compute mean of the array elements]
      mean=sum/n
6. [Compute standard deviation of the array elements]
      sumstd=sumstd+pow((*ptr-mean),2)
      std=sqrt(sumstd/n);
7. [Output]
Print the values of sum, mean and standard deviation for the given array elements
8. [Finished]
      End.
```

Program 10)

Flowchart


START
Read n
i=0
i<n
i++
False
True
Read elements of the array a[i]
ptr=a
i=0
i<n
i<n
False
True
sum=sum+*ptr
ptr++
mean=sum/n
ptr=a
i=0
i<n
i++
False
True
sumstd=sumstd+pow((*ptr-mean),2)
ptr++
std=sqrt(sumstd/n)
Print sum, mean, standard deviation
STOP