# C PROGRAMMING LAB

**AURORA'S TECHNOLOGICAL AND RESEARCH INSTITUTE**
**DEPARTMENT OF CSE**

**EXPERIMENT LIST**

**CLASS:** I B.TECH CSE,ECE,EEE,ME,CE **SUBJECT:** C PROGRAMMING (R13)

**Recommended Systems/Software Requirements:**

- Intel based desktop PC
- ANSI C Compiler with Supporting Editors

**Week 1** : **Page No.5 - 8**
**a)** Write a C program to find the sum of individual digits of a positive integer.
**b)** A Fibonacci sequence is defined as follows: the first and second terms in the sequence are 0 and 1. Subsequent terms are found by adding the preceding two terms in the sequence. Write a C program to generate the first n terms of the sequence.
**c)** Write a C program to generate all the prime numbers between 1 and n, where n is a value supplied by the user.

**Week 2**: **Page No. 9 - 13**
**a)** Write a C program to calculate the following Sum:
Sum=$1-x^2/2! +x^4/4!-x^6/6!+x^8/8!-x^{10}/10!$
**b)** Write a C program toe find the roots of a quadratic equation.

**Week 3 :** **Page No.14 - 19**
**a)** The total distance traveled by vehicle in't' seconds is given by distance = $ut+1/2at^2$ where 'u' and 'a' are the initial velocity (m/sec.) and acceleration (m/sec$^2$). Write C program to find the distance traveled at regular intervals of time given the values of 'u' and 'a'. The program should provide the flexibility to the user to select his own time intervals and repeat the calculations for different values of 'u' and 'a'.
**b)** Write a C program, which takes two integer operands and one operator form the user, performs the operation and then prints the result. (Consider the operators +,-,*, /, % and use Switch Statement)

**Week 4:** **Page No.20 - 26**
**a)** Write C programs that use both recursive and non-recursive functions
i) To find the factorial of a given integer.
ii) To find the GCD (greatest common divisor) of two given integers.

**Week 5 :** **Page No.27 - 41**
**a)** Write a C program to find both the largest and smallest number in a list of integers.
**b)** Write a C program that uses functions to perform the following:
i) Addition of Two Matrices
ii) Multiplication of Two Matrices

**Week 6 :** **Page No.42 - 49**
**a)** Write a C program that uses functions to perform the following operations:
i) To insert a sub-string in to given main string from a given position.
ii) To delete n Characters from a given position in a given string.
**b)** Write a C program to determine if the given string is a palindrome or not

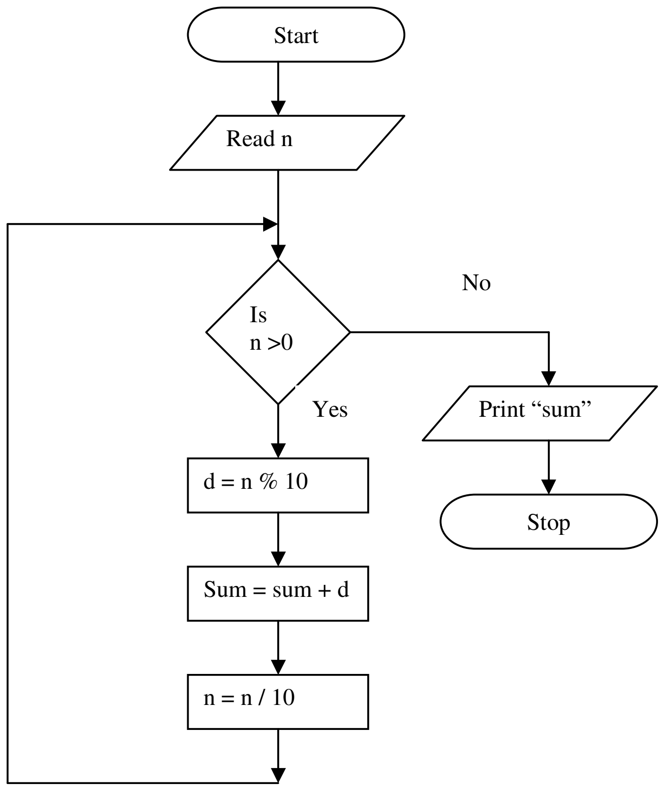
## WEEK-1

**A) AIM:** Write a C program to find the sum of individual digits of a positive integer.

**Algorithm:**

```
1. Read the number n
2. Initialize sum ← 0
3. while n > 0
4.   d ← n%10
5.   sum ← sum+d
6.   n ← n/10
7. print sum.
```

**Flow chart:**



**Program:**

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int n, sum=0,d;
    clrscr();
printf("Enter any integer:");
scanf("%d", &n);
while(n>0)
{
    d=n%10;
    sum=sum+d;
    n=n/10;
}
Printf("sum of individual digits is %d",sum);
getch();
}
```

**Result:**

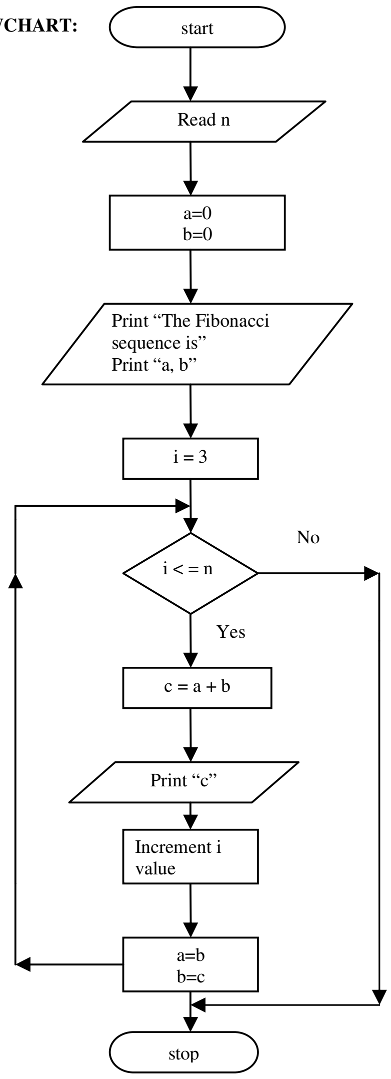
```
Enter any integer: 1234
Sum of individual digits is: 10
```

**B) AIM:** A Fibonacci sequence is defined as follows: the first and second terms in the sequence are 0 and 1. Subsequent terms are found by adding the preceding two terms in the sequence. Write a C program to generate the first n terms of the sequence.

**Algorithm:**

```
1. Read the number of terms n
2. Initialize a ← 0, b ← 1
3. print a and b values
4. for i ← 3 to n
      a. increment the i value
      b. c ← a+b
      c. print c value
      d. a ← b
      e. b ← c
```

**FLOWCHART:**



**Program:**

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int a=0,b=1,c,n,i;
    clrscr();
    printf("Enter no. of terms:");
    scanf("%d", &n);
    printf("The Fibonacci sequence
is:");
    printf("%d%d", a,b);
    for(i=3;i<=n;i++)
    {
            c=a+b;
            printf("%d",c);
            a=b;
            b=c;
    }
getch();
}
```

**Result:**

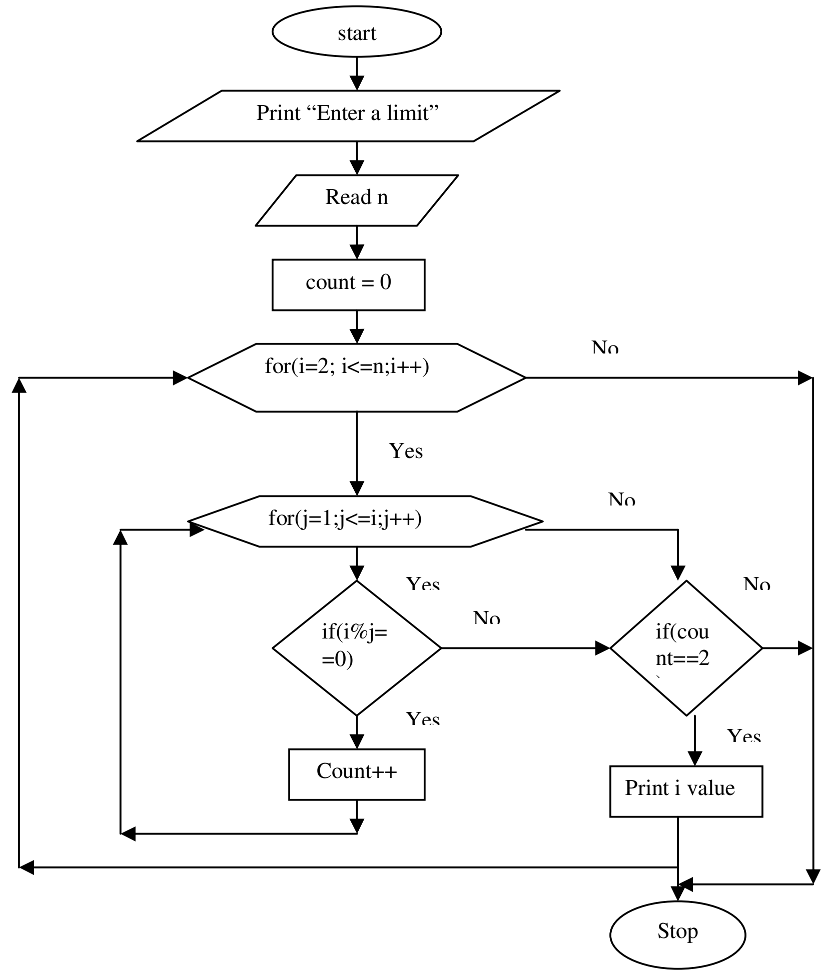
```
Enter no of items: 5
The Fibonacci sequence is
0 1 1 2 3
```

**C) AIM:** Write a C program to generate all the prime numbers between 1 and n is a value supplied by the user.

**Algorithm:**

```
1. Read n value
2. Initialize count ← 0
3. for i ← 2 to n
      a. for j ← 1 to i
      b. if i mod j is equal to 0
      c. then increment count
      d. if count is equal to 2
      e. then print i value.
```

**Flow chart:**



**Program:**

```
#incloude<stdio.h>
#Include<conio.h>
void main()
{
    int i, j, n, count=0;
    clrscr();
    printf("Enter the limit:");
    scanf("%d", &n);
    printf("The prime numbers are:");
    for(i=2;i<=n;i++)
    {
            for(j=1;j<=i;j++)
            {
                    if(i%j==0)
                    count++;
            }
            if(count==2)
            printf("%d\t", i);
    }
getch();
}
```

**Result:**

```
Enter the limit: 4
The prime numbers are:
2 3 5 7
```

## WEEK-2

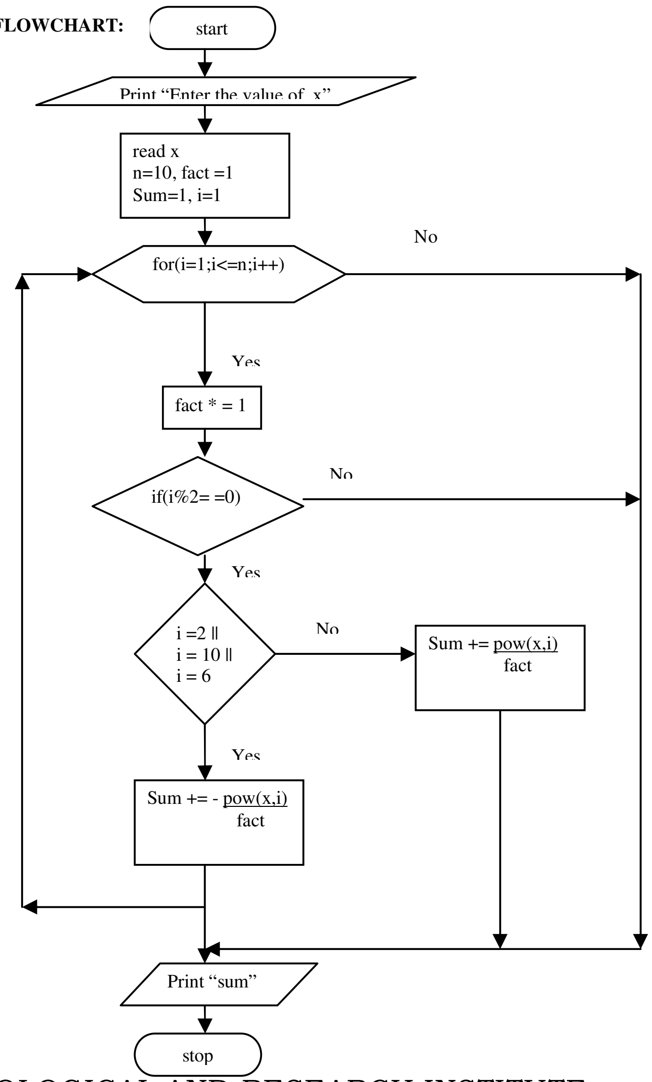
**A) AIM:** Write a C program to calculate the following sum:

Sum=1-x^2/2!+x^4/4!-x^6/6!+x^8/8!-x^10/10!

**Algorithm:**

```
1. Read the x value
2. Initialize fact ← 1, sum ← 1 and n ← 10
3. for i ← 1 to n
      a. fact ← fact*i
      b. if i mod 2 is equal to 0
      c. then if  i  equal to 2 or i equal to 10 or i equal to 6
      d. then sum+= -pow(x,i)/fact
      e. else  sum+=pow(x,i)/fact
4. print sum
```

**FLOWCHART:**



**Program:**

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
void main()
{
    int i,n=10,x;
    long int fact=1;
    float sum=1;
    printf("Enter the x value:");
    scanf("%d", &x);
    for(i=1;i<=n;i++)
    {
            fact=fact*i;
            if(i%2==0)
            {
                    if(i==2||i=10||i==6)
                    sum+= -pow(x,i)/fact;
                    else
                    sum+=pow(x,i)/fact;
            }
    }
Printf("sum is %f", sum);
}
```

**Result:**

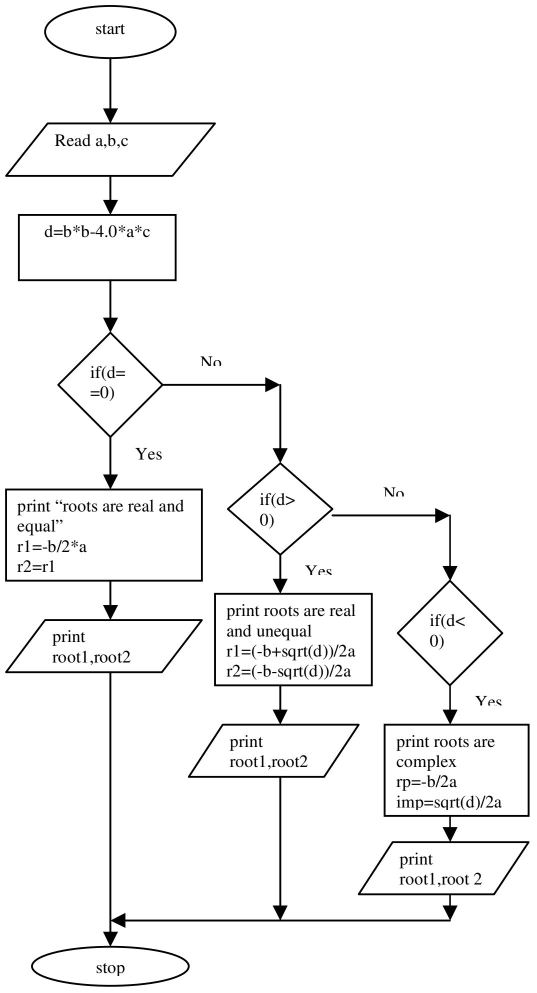
```
Enter x value: 2
Sum is: 0
```

**B)** **AIM:** Write a C program to find the roots of a quadratic equation.

**Algorithm:**

```
1. Read a,b,c values
2. Initialize d ← b*b-4*a*c
3. if d==0
        a. then print "roots are real and equal"
        b. r1← -b/2*a,r2 ← r1
4. else if d>0
        a. then print "roots are real and distinct"
        b. r1← (-b+sqrt(d))/2*a, r2← (-b-sqrt(d))/2*a
5. else if d<0
        a. then print "roots are complex"
        b. rp← -b/2a, imp← sqrt(d)/2*a
        c. print r1 and r2.
```

**FLOWCHART**


start
Read a,b,c
d=b*b-4.0*a*c
if(d= =0)
No
Yes
print "roots are real and equal" r1=-b/2*a r2=r1
print root1,root2
if(d> 0)
No
Yes
print roots are real and unequal r1=(-b+sqrt(d))/2a r2=(-b-sqrt(d))/2a
print root1,root2
if(d< 0)
Yes
print roots are complex rp=-b/2a imp=sqrt(d)/2a
print root1,root 2
stop

**<u>Program</u>:**

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
void main()
{
    float a,b,c,d,r1,r2,imp,rp;
    clrscr();
    printf("Enter a,b,c:");
    scanf("%f%f%f",&a,&b,&c);
    d=b*b-4.0*a*c;
    if(d= =0)
    {
            Printf("roots are real and equal");
            r1=-b/2*a;
            r2=r1;
            printf("root1=%f",r1);
            printf("root2=%f",r2);
    }
else if(d>0)
    {
            Printf("roots are real and unequal");
            r1=(-b+sqrt(d))/2*a;
```

```
                r2=(-b-sqrt(d))/2*a;
                printf("root1=%f",r1);
                printf("root2=%f",r2);
        }
    else if(d<0)
        {
                d=-d;
                printf("roots are complex");
                rp=-b/2*a;
                imp=sqrt(d)/2*a;
                printf("root1=%f+i%f",rp,imp);
                printf("root2=%f-i%f",rp,imp);
    }
    getch();
    }
```

**Result:**

```
Enter a,b & c:  1 5 3
Roots are real & unequal
```

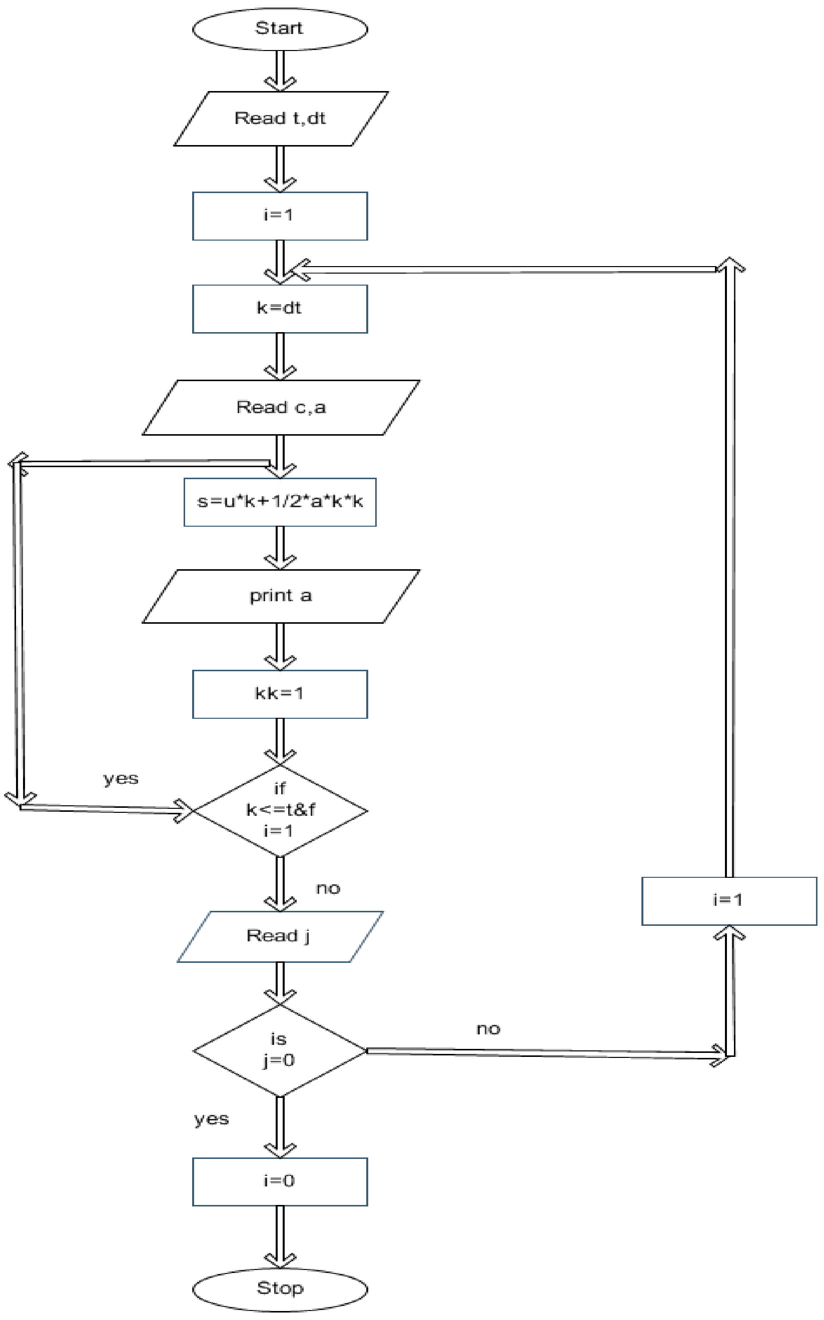
## WEEK-3

**A)** The total distance travelled by vehicle in 't' seconds is given by distance = ut+1/2at2 where 'u' and 'a' are the initial velocity (m/sec.) and acceleration (m/sec2). Write C program to find the distance travelled at regular intervals of time given the values of 'u' and 'a'. The program should provide the flexibility to the user to select his own time intervals and repeat the calculations for different values of 'u' and 'a'.

**Algorithm:**

```
Step 1:Start

Step2 : Read t ,dt

Step 3: Set i to 1

Step 4:Set k to dt

Step 5: Read u,a

Step 6: set s to u*k+0.5*d*k*k

Step 7: Write s

Step 8: If(k<=t) and i=1 then
          Begin
          Step 8.1 go to step 6
  And
           Else
                    Begin
          Step 8.2 :read
          Step 8.3 :if(j=0) then
          Begin
                    Step 8.3.1:Set I to 0
          End
                    Else
                              Begin
                                        Step 8.3.2: Set I to 1
                                        Step 8.3.3: go to step 4
          End
Step 9: Stop

Step 10: End
```

**Flowchart:**


Start
Read t,dt
i=1
k=dt
Read c,a
s=u*k+1/2*a*k*k
print a
kk=1
yes
if k<=t&f i=1
no
Read j
is j=0
no
i=1
yes
i=0
Stop

**Program:**

```
#include<stdio.h>
main()
{
        int a,u,t,t1,t2,i;
        float s;
        clrscr();
        printf("ENTER THE VALUES OF a,u,t,t1,t2:");
        scanf("%d%d%d%d%d",&a,&u,&t,&t1,&t2);
        for(i=t1;i<=t2;i=i+t) //  performing the looping operation for time intervals
        {
                s=(u*i)+(0.5*a*i*i);  // calculate the total distance
                printf("\n\nthe distance travelled in %d seconds is %f ",i,s);
        }
getch();
}
```

**Input/Output:**

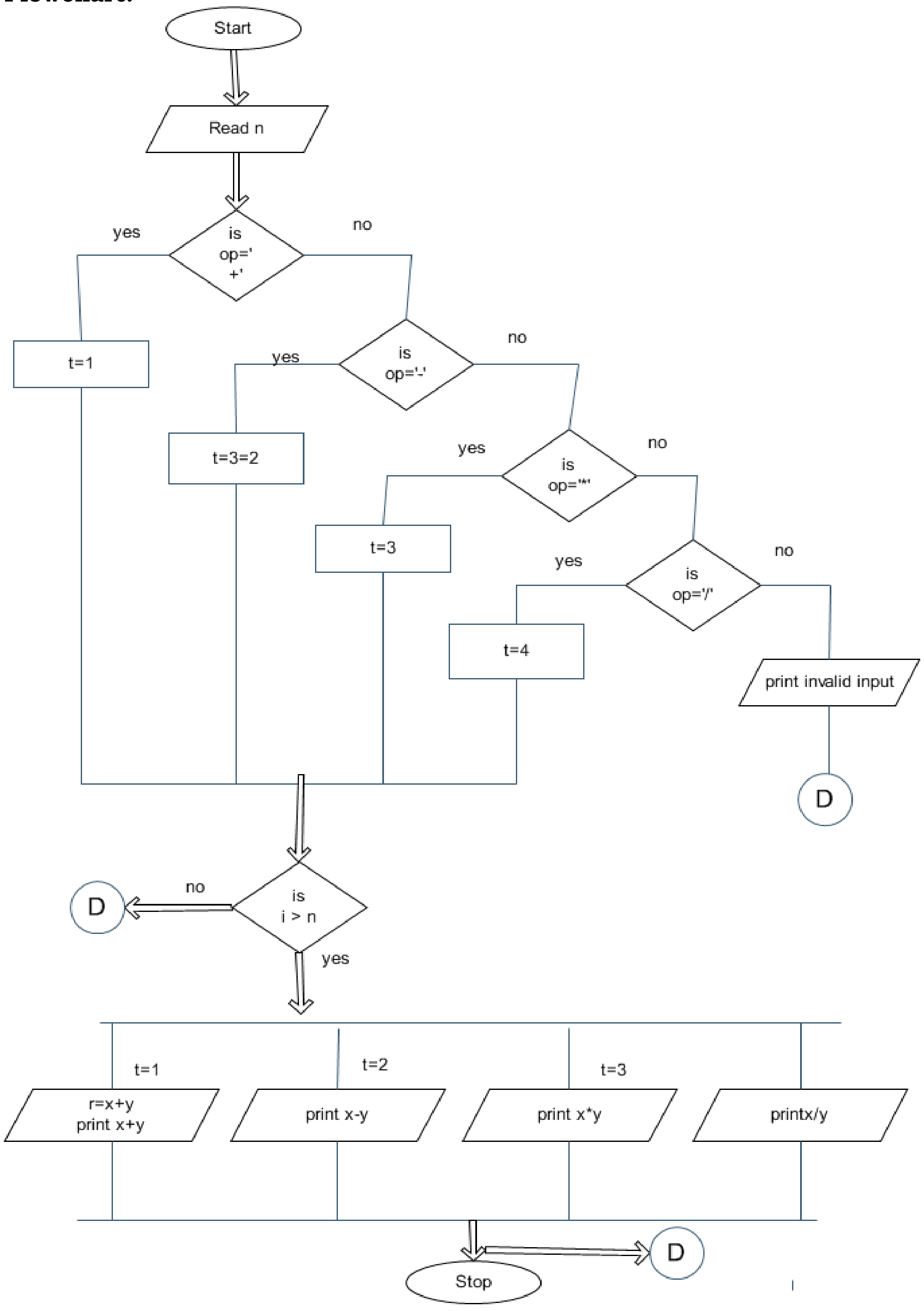
```
 1.ENTER THE VALUES OF a,u,t,t1,t2:
    1
    2
    3
    1
    5
     the distance travelled in 1 seconds is 2.500000

      the distance travelled in 4 seconds is 16.000000
2.ENTER THE VALUES OF a,u,t,t1,t2:0
   1
   2
   3
   4
    the distance travelled in 3 seconds is 3.000000
```

**B) AIM:** Two integer operands and one operator form user, performs the operation and then prints the result.
(Consider the operators +,-,*, /, % and use Switch Statement)

**Algorithm:**

```
Step 1: Start

Step 2: Read the values of a,b and operator

Step 3: if the operator is '+' then
        R=a+b
        Go to step 8
        Break

Step 4: Else if the operator is '-' then
        R=a-b
        Go to step 8

Step 5: Else if the operator is '*' then
        R=a*b
        Go to step 8

Step 6: Else if the operator is '/' then
        R=a/b
        Go to step 8

Step 7: Else if the operator is '%' then
        R=a%b
        Go to step 8

Step 8: write  R

Step 9:End
```

**Flowchart:**


Start
Read n
is op='+'
yes
no
t=1
is op='-'
t=3=2
is op='*'
t=3
is op='/'
t=4
print invalid input
D
is i > n
no
yes
t=1
t=2
t=3
r=x+y print x+y
print x-y
print x*y
printx/y
Stop

**Program:**

```
#include<stdio.h>
main()
{
        char op;
        float a,b,c;
        clrscr();
        printf("enter two operands:");
        scanf("%d%d",&a,&b);
        printf("enter an operator:");
        scanf(" %c",&op);
        switch(op) // used to select particular case from the user
        {
        case '+':printf("sum of two numbers %2d %2d is:     %d",a,b,a+b);
                    break;
        case '-':printf("subtraction of two numbers %2d %2d is:
                      %d",a,b,a-b);
                    break;
        case '*':printf("product of two numbers %2d %2d is:
                       %d",a,b,a*b);
                     break;
         case '/':printf("quotient of two numbers %2d %2d is:
                       %d",a,b,a/b);
                      break;
          case '%':printf("reminder of two numbers %2d %2d is:
                        %d",a,b,c);
                        break;
           default:printf("please enter correct operator");
                        break;
          }
  getch();
}
```

**Result:**

```
1.enter two operands:2 3
    enter an operator:+
    sum of two numbers  2  3 is: 5

2.enter two operands:3 4
   enter an operator: -
     subtraction of two numbers  3  4 is: -1

3.enter two operands:3 5
    enter an operator:*
     product of two numbers  3  5 is: 15
```

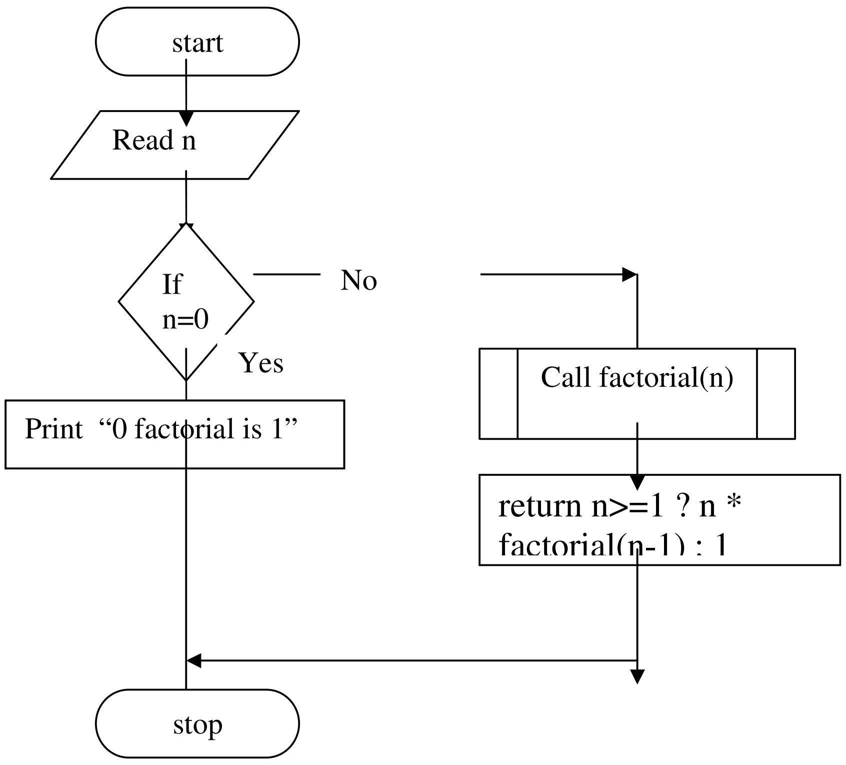
**WEEK-4**

**A) AIM:** Write a C program to find the factorial of a given integer by using recursive and non-recursive functions.

**i)Recursive Algorithm:**

1. Define the recursive function
2. Read the number n
3. if n is equal to 0
4. then print "factorial of 0 is 1"
5. else call the recursive function
6. print the factorial value.

**Flow chart:**



**Program:**

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

unsigned int factorial(int n);

void main()
{
  int n,i;
  long int fact;
  clrscr();
  printf("Enter the number: ");
  scanf("%d",&n);

  if(n==0)
    printf("Factorial of 0 is 1\n");
  else
       printf("Factorial of %d Using Recursive Function is      %d\n",n,factorial(n));

   getch();
}

/* Recursive Function*/
unsigned int factorial(int n)
{
    return n>=1 ? n * factorial(n-1) : 1;
}
```

**Result:**

```
Enter number: 5
Factorial of 5 using recursive function is: 120
```

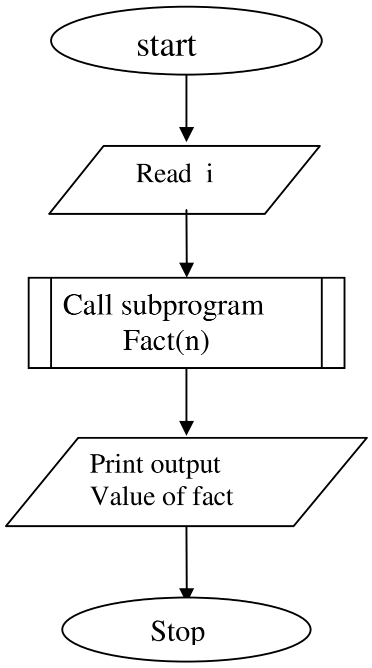
**ii)Non-Recursive Algorithm:** main program

```
Step 1: start
Step 2: read n
Step 3: call the sub program fact(n)
Step 4: print the f value
Step 5: stop
```

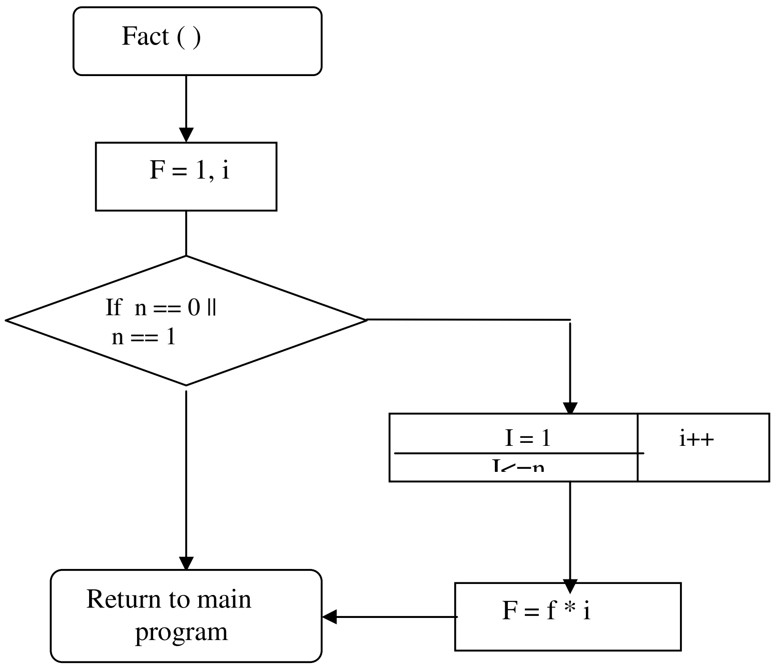
**Sub program: fact**

```
Step 1: initialize the f=1
Step 2: if n==0 or n=1 return 1 to main program. If not goto step 3
Step 3: perform the looping operation as follows
          For i=1 i<=n; i++
Step 4: f=f*i
Step 5: return f value to the main program
```

**Factorial nonrecursive**



**Sub program**



**Program:**

```
#include<stdio.h>
#include<conio.h>
int fact(int n)   //starting of the sub program
 {
         int f=1,i;
         if((n==0)||(n==1))  // check the condition for n value
         return(1);
         else
        for(i=1;i<=n;i++)  // perform the looping operation for calculating the factorial
         f=f*i;
         return(f);
 }
void main()
{
         int n;
         clrscr();
        printf("enter the  number :");
         scanf("%d",&n);
         printf("factoria of number%d",fact(n));
         getch();
}
```

**Result:**

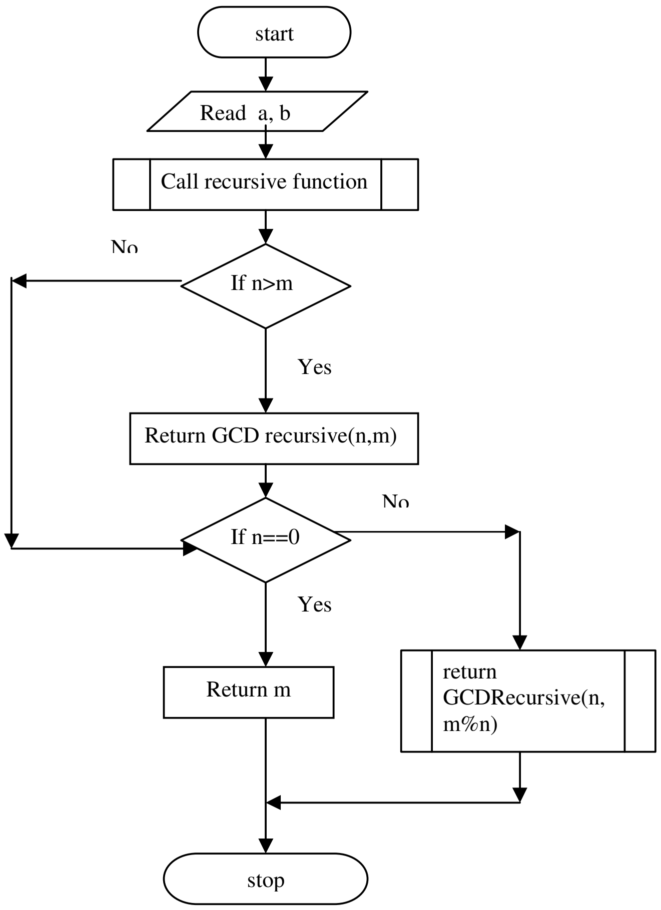
```
1.Enter a number: 7
  Factorial of number: 5040
```

**B) AIM:** Write a C program to find the GCD(greatest common divisor) of two given integers by using recursive and Non-recursive functions.

**i)Recursive Algorithm:**

1. Define the recursive function
2. Read the a,b values
3. Call the recursive function
    a. if n>m
    b. then return the function with parameters m,n
    c. if n==0
    d. then return m
    e. else return the function with parameters n,m%n.

**Flow chart:**



**<u>Program</u>:**

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

unsigned int GCDRecursive(unsigned m, unsigned n);
int main(void)
{
  int a,b;
  clrscr();

  printf("Enter the two numbers whose GCD is to be found: ");
  scanf("%d%d",&a,&b);

  printf("GCD of %d and %d Using Recursive Function is %d\n",a,b,GCDRecursive(a,b));

  getch();
}

/* Recursive Function*/
unsigned int GCDRecursive(unsigned m, unsigned n)
{
 if(n>m)
        return GCDRecursive(n,m);
 if(n==0)
         return m;
 else
     return GCDRecursive(n,m%n);
}
```

**<u>Result:</u>**

```
Enter the two numbers whose GCD is to be found 18 6
GCD of 18 and 6 Using Recursive Function is 6
```

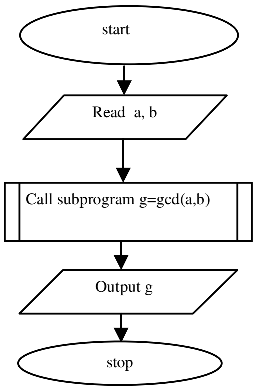
**ii)Non-Recursive Algorithm:**

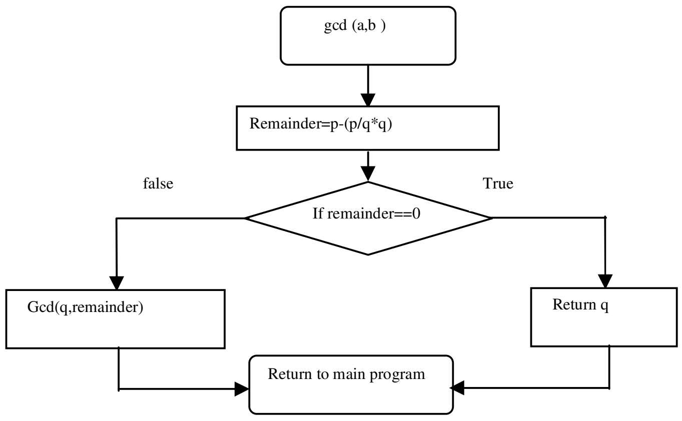
```
Step 1: start
Step 2: read a,b
Step 3: call sub program g=GCD(a,b)
Step 4: print the g value
Step 5: stop
```

Sub program: GCD(a,b)

```
Step 1: initialize the p=1, q, remainder
Step 2: remainder=p-(p/q*q)
Step 3: remainder=0 return q else goto step 4
Step 4: GCD(q,remainder) return to main program
```

**Flowchart:**





**Program:**

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
int gcdnonrecursive(int m,int n)
{
        int remainder;
        remainder=m-(m/n*n);
         if(remainder==0)
        return n;
         else
         gcdnonrecursive(n,remainder);
}

void main()
{
         int a,b,igcd;
          clrscr();
         printf("enter the two numbers whose gcd is to be found:");
        scanf("%d%d",&a,&b);
        printf("GCD of %d",gcdnonrecursive(a,b));
                        getch();
}
```

**Result:**

```
1. enter the two numbers whose gcd is to be found:5,25
   GCD of  a,b is : 5
2. enter the two numbers whose gcd is to be found:36,54
   GCD of  a,b is : 18
```

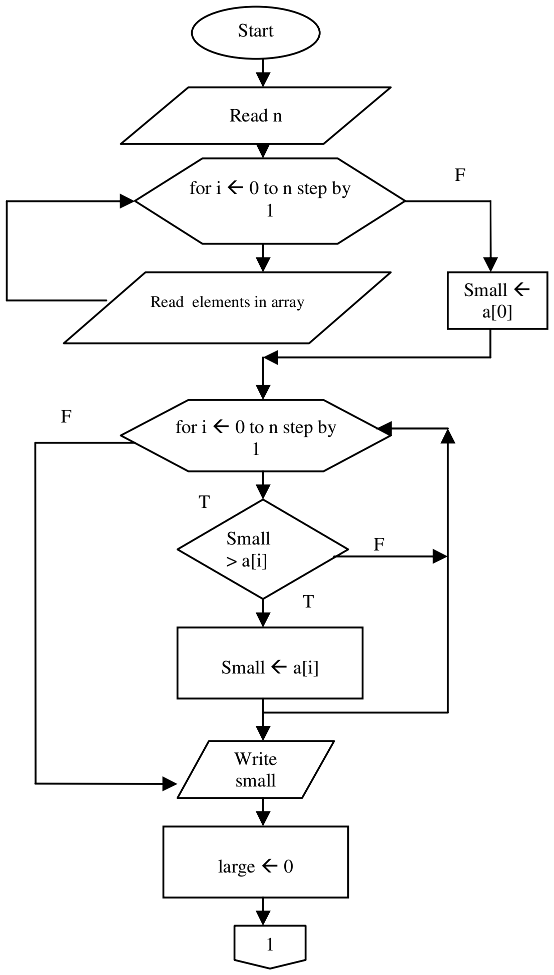
## WEEK-5

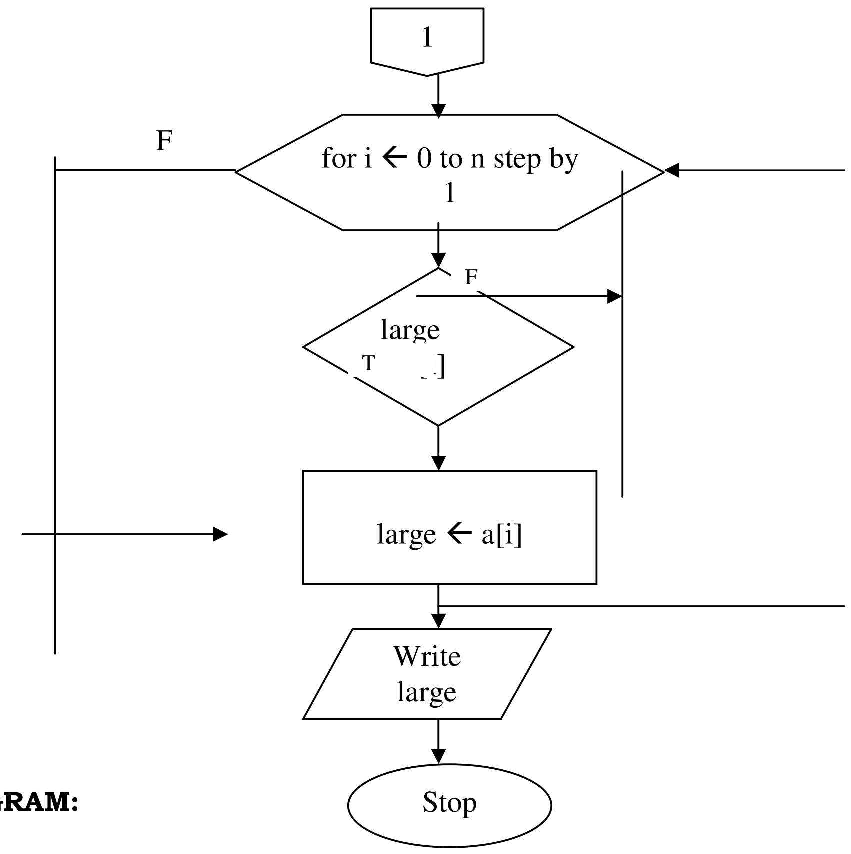
**A) AIM: -** A C program to find both the largest and smallest number in list of integers

**Algorithm:**

```
1.  Start
2.  Read n
3.  for i ← 0 to n
4.          do read a[i]
5.  small  ← a[0]
6.  for i ← 0 to n
7.        do if small > a[i]
8.              then  small ← a[i]
9.  write small
10. large ← 0
11. for i ← 0 to n
12.       do if large <a[i]
13.             then large ← a[i]
14.  write large
15. Stop
```

**Flowchart:**





**PROGRAM:**

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

Void main()
{

        int i,n,small=0,large=0;
        int a[30];

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

        printf("\n Enter values in array elements:");
        for(i=0;i<n;i++)
        {
                scanf("%d",&a[i]);
         }
        small = a[0];
        for(i=0;i<n;i++)
        {
                if(small > a[i])
                   small = a[i];
        }
        printf("\n The smallest element in given array is %d",small);
large=0;
        for(i=0;i<n;i++)
        {
                if(large < a[i])
                  large = a[i];
         }
         printf("\n The largest element in given array is %d",large);

        printf("\n :End of the Main Program:");
        getch();
}
```

**RESULT:**

```
Input :
     Enter size of the array: 9
     Enter values in array elements:
     96   46   86   6   36   76   26  16   56
Output:
     The smallest element in given array is 6
     The largest element in given array is  96
```

**B)Aim:** Write a c- program that uses functions to perform addition and multiplication on two mattrices.

**Algorithm**

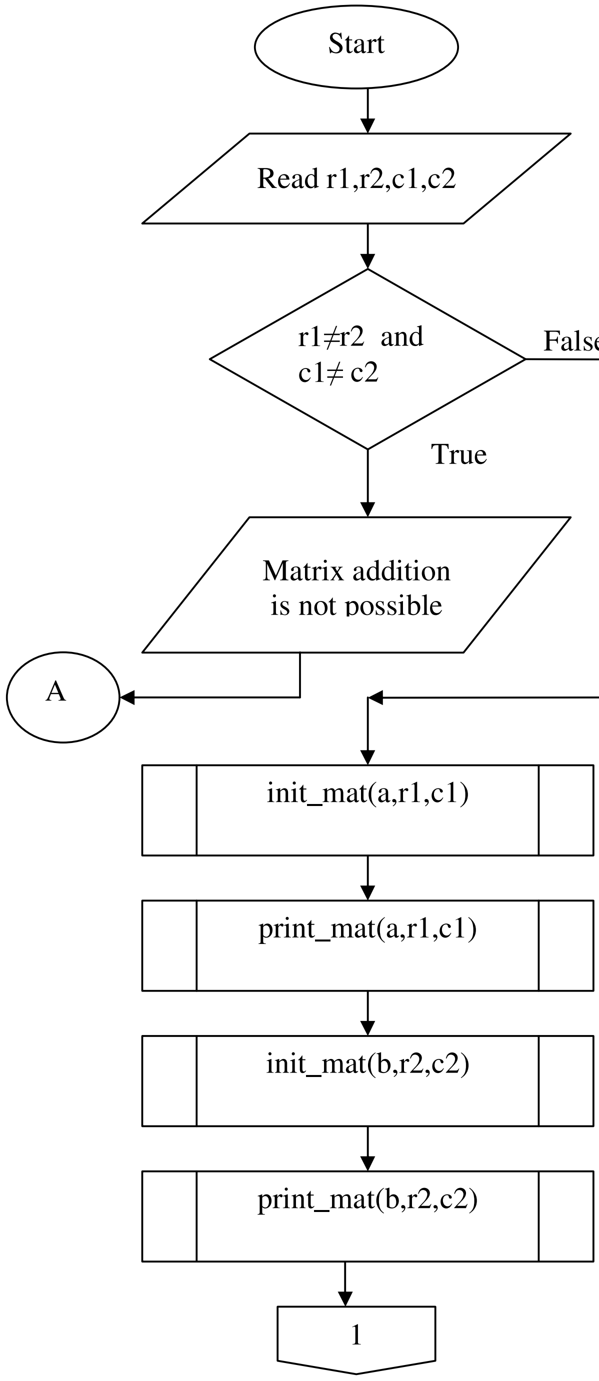
```
1.  Start
2.  read r1,r2,c1,c2
3.  if r1 ≠ r2 and c1 ≠ c2
4.      then "matrix addition is not possible"
5.  else
6.      do  init_mat(a,r1,c1)
7.          print_mat(a,r1,c1)
8.          init_mat(b,r2,c2)
9.          print_mat(b,r2,2)
10.         add_mat(a,b,c,r1,c1)
11.         print_mat(c,r1,c1)
12. Stop
```

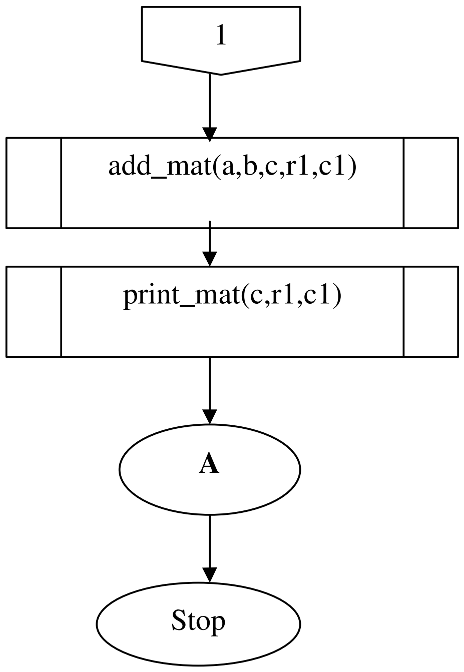
```
init_mat(a4,r4,c4)
  1. for i ← 0 to r4
  2.     do  for j ← 0 to c4
  3.             read a4[i][j]
```

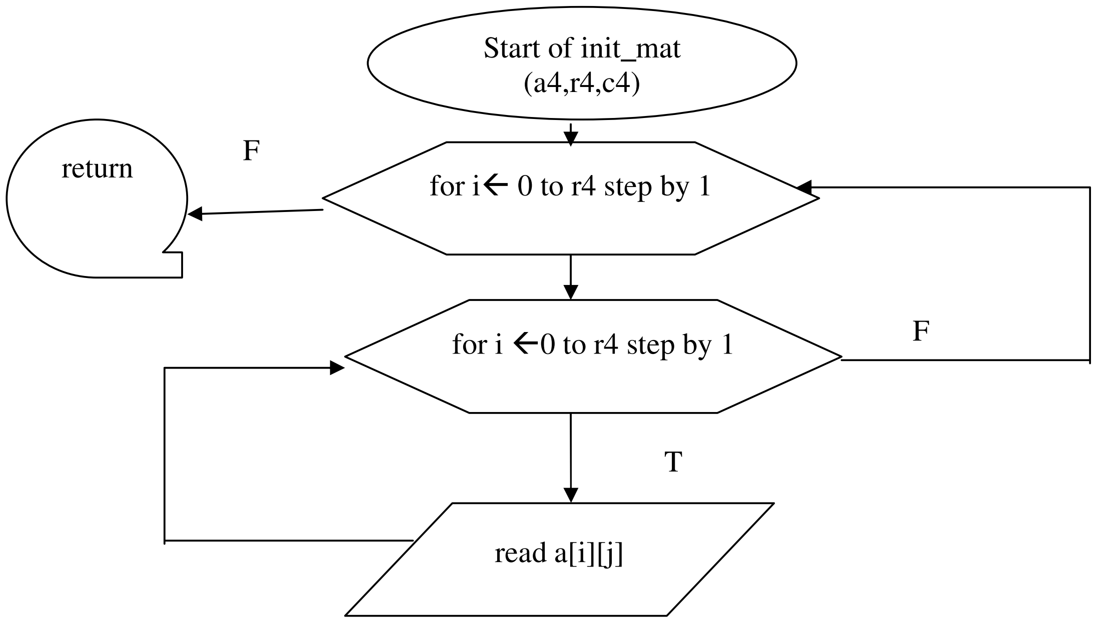
```
print_mat(a4,r4,c4)
  1. for i ← 0 to r4
  2.    do  for j ← 0 to c4
  3.            print a[i][j]
  4.     print next_line
```

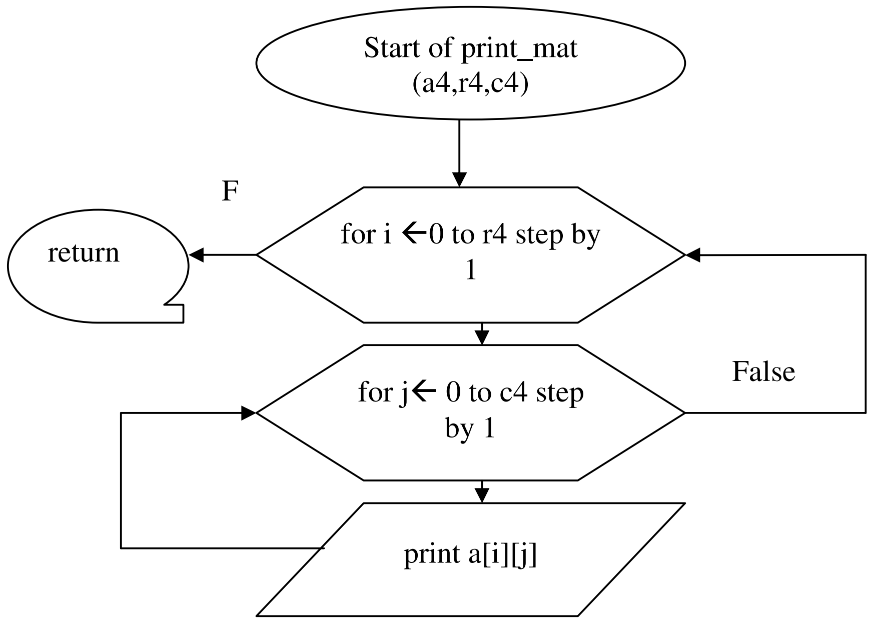
```
add_mat(a4,b4,c24.r4,c4)
  1. for i ← 0 to r4
  2.      do for j ← to c4
  3.            c[i][j] ← a[i][j] + b[i][j]
```

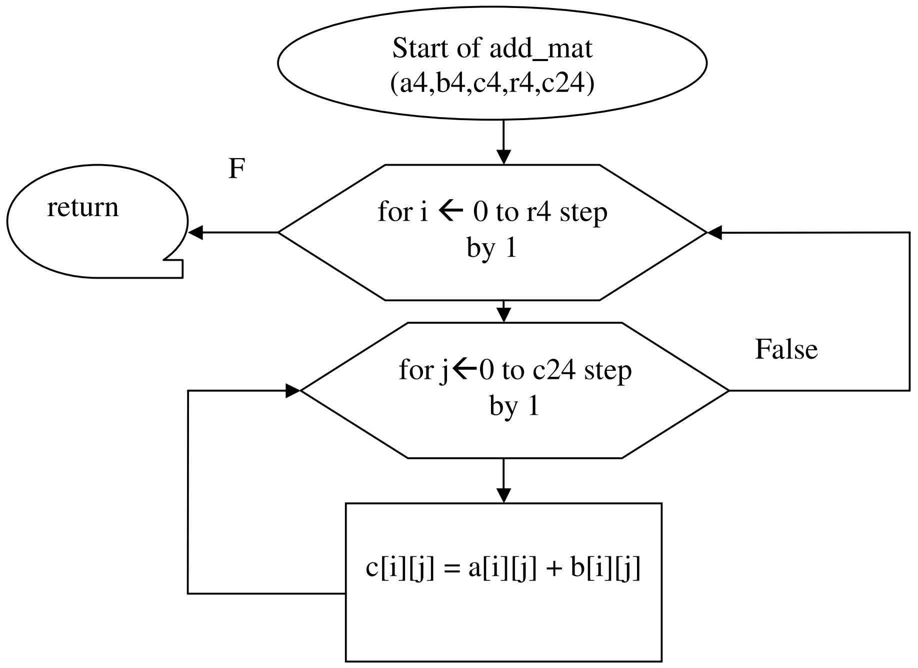
**Flowchart:-**


Start
Read r1,r2,c1,c2
r1≠r2 and c1≠ c2
False
True
Matrix addition is not possible
A
init_mat(a,r1,c1)
print_mat(a,r1,c1)
init_mat(b,r2,c2)
print_mat(b,r2,c2)
1


1
add_mat(a,b,c,r1,c1)
print_mat(c,r1,c1)
A
Stop


Start of init_mat (a4,r4,c4)
return
F
for i← 0 to r4 step by 1
for i ←0 to r4 step by 1
F
T
read a[i][j]




Start of add_mat (a4,b4,c4,r4,c24)
F
return
for i ← 0 to r4 step by 1
for j←0 to c24 step by 1
False
c[i][j] = a[i][j] + b[i][j]

**PROGRAM:**

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

void init_mat (int [][10], int, int);
void print_mat (int [][10], int, int);
void add_mat (int [][10], int [][10], int [][10], int, int);
```

```
main()
{
 int r1,r2,c1,c2;
 int a[10][10],b[10][10],c[10][10];
 clrscr();

 /* Giving order of the Matrix - A */
 printf("\n Enter the order of Matrix – A:");
 scanf("%d%d",&r1,&c1);

 /* Giving order of the Matrix - B */
 printf("\n Enter the order of Matrix – B:");
 scanf("%d%d",&r2,&c2);

 if(r1!=r2 || c1!=c2)
 {
         printf("\n Matrix Addition is not possible ");
         getch();
         exit(0);
  }
 else
  {
   /*  Matrix - A */
   printf("\n Enter the elements of Matrix – A:");
   init_mat(a,r1,c1);
   printf("\n The elements of Matrix - A");
   print_mat(a,r1,c1);

   /* Matrix - B */
   printf("\n Enter the elements of Matrix - B");
   init_mat(b,r2,c2);
   printf("\n The elements of Matrix - B");
   print_mat(b,r2,c2);


   /* Function call to Matrix addition logic */
   add_mat(a,b,c,r1,c1);

   /* Matrix after addition */
   printf("\n The elements of Matrix - C after addition of A & B");
   print_mat(c,r1,c1);
   }
   getch();
 }

/* Function for  two dimensional array initialization */
void init_mat(int mat[][10],int r,int c)
{
     int i,j;
     for(i=0;i<r;i++)
     {
          for(j=0;j<c;j++)
          {
             scanf("%d",&mat[i][j]);
          }
     }
 }

 /* Function for printing element in Matrix form */
void print_mat(int mat[][10],int r, int c)
{
     int i,j;
     printf("\n");
     for(i=0;i<r;i++)
     {
          for(j=0;j<c;j++)
          {
             printf(" %d ",mat[i][j]);
          }
          printf("\n");
     }
 }

/* function for matrix addition logic */
void add_mat(int a[][10],int b[][10],int c[][10],int r1,int c1)
{
     int i,j;

     for(i=0;i<r1;i++)
     {
          for(j=0;j<c1;j++)
          {
             c[i][j] = a[i][j]+b[i][j];
          }     } }
```

**RESULT:**
***Case - 1***
**Input :**

```
      Enter the order of Matrix – A:  2  2
      Enter the order of Matrix – B:  2  2
      Enter the elements of Matrix – A: 1 2 3 4
      The elements of Matrix – A:
       1    2
       3    4
      Enter the elements of Matrix – B: 1 2 3 4
      The elements of Matrix – B:
      1   2
      2   4
```

**Output:**

```
     The elements of Matrix - C after addition of A & B:
      2    4
      4    8
```

***Case – 2***
**Input :**

```
     Enter the order of Matrix – A:  2  3
     Enter the order of Matrix – B:  2  2
```

**Output :**

```
    Matrix Addition is not possible
```

**B) AIM:** Write A C- Program That Uses Functions To Perform Matrice Multiplication On Two Matrices.

**Algorithm**

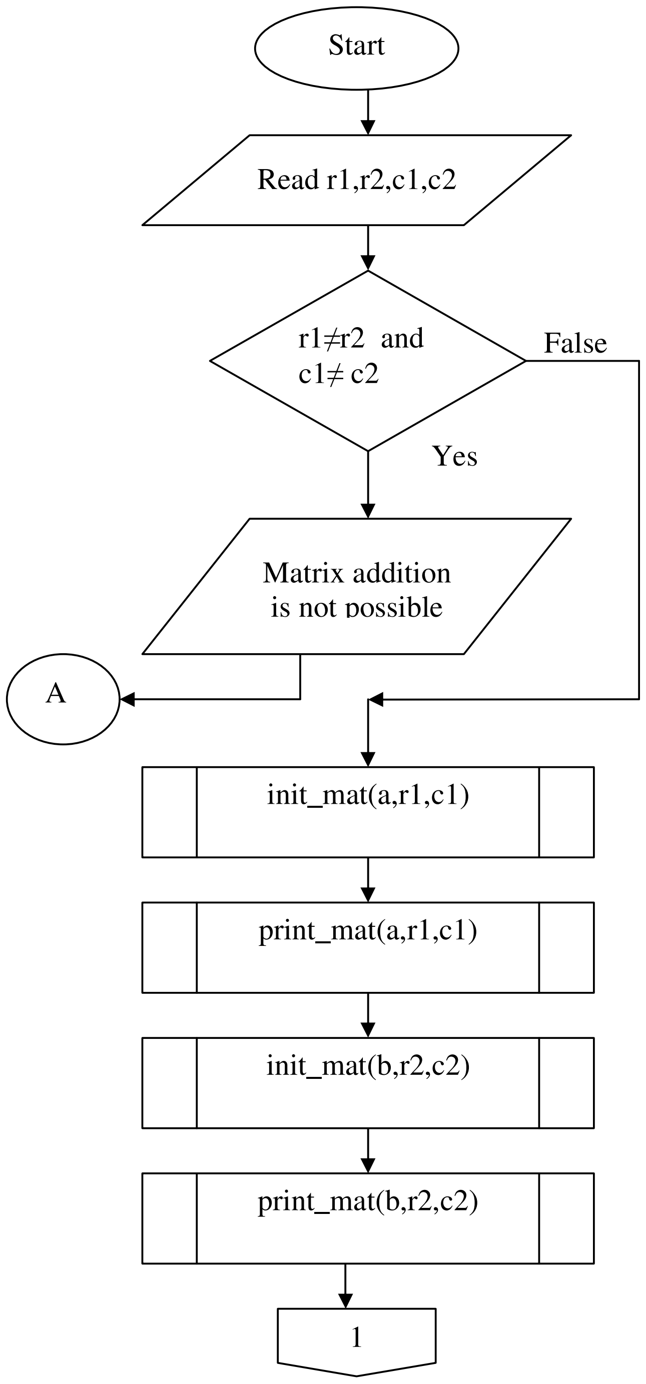
```
1.  Start
2.  read r1,r2,c1,c2
3.  if r1 ≠ c2
4.      then "matrix multiplication is not possible"
5.  else
6.      do  init_mat(a,r1,c1)
7.          print_mat(a,r1,c1)
8.          init_mat(b,r2,c2)
9.          print_mat(b,r2,2)
10.         mul_mat(a,b,c,r1,c1,c2)
11.         print_mat(c,r1,c1)
12. Stop
```

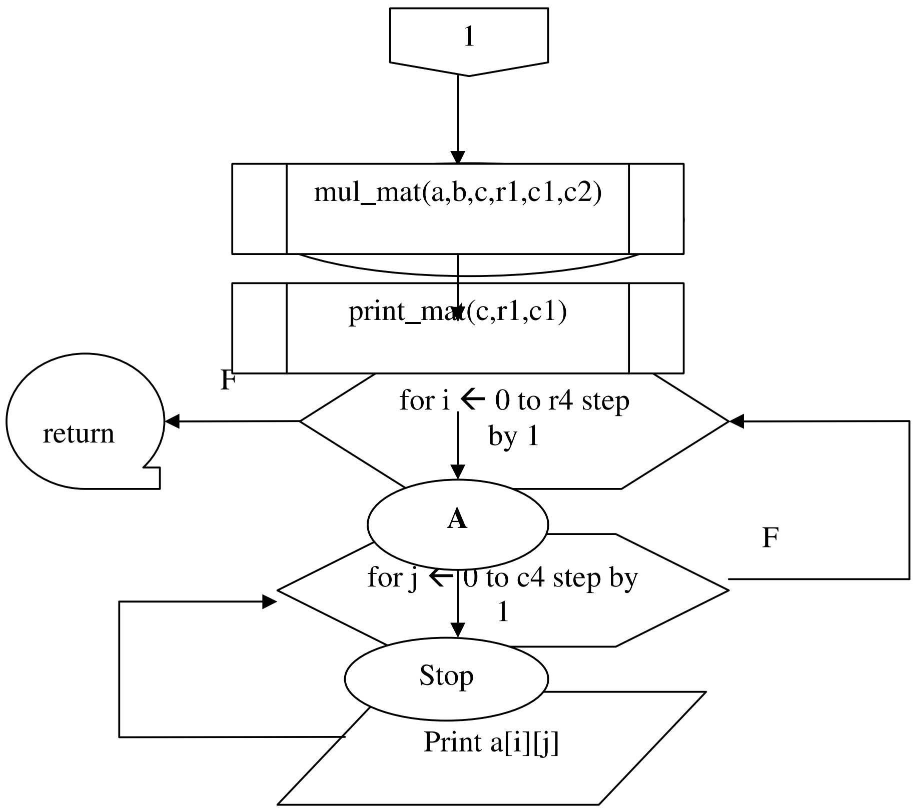
```
init_mat(a4,r4,c4)
    1. for i ← 0 to r4
    2. do  for j ← 0 to c4
    3.             read a4[i][j]
```

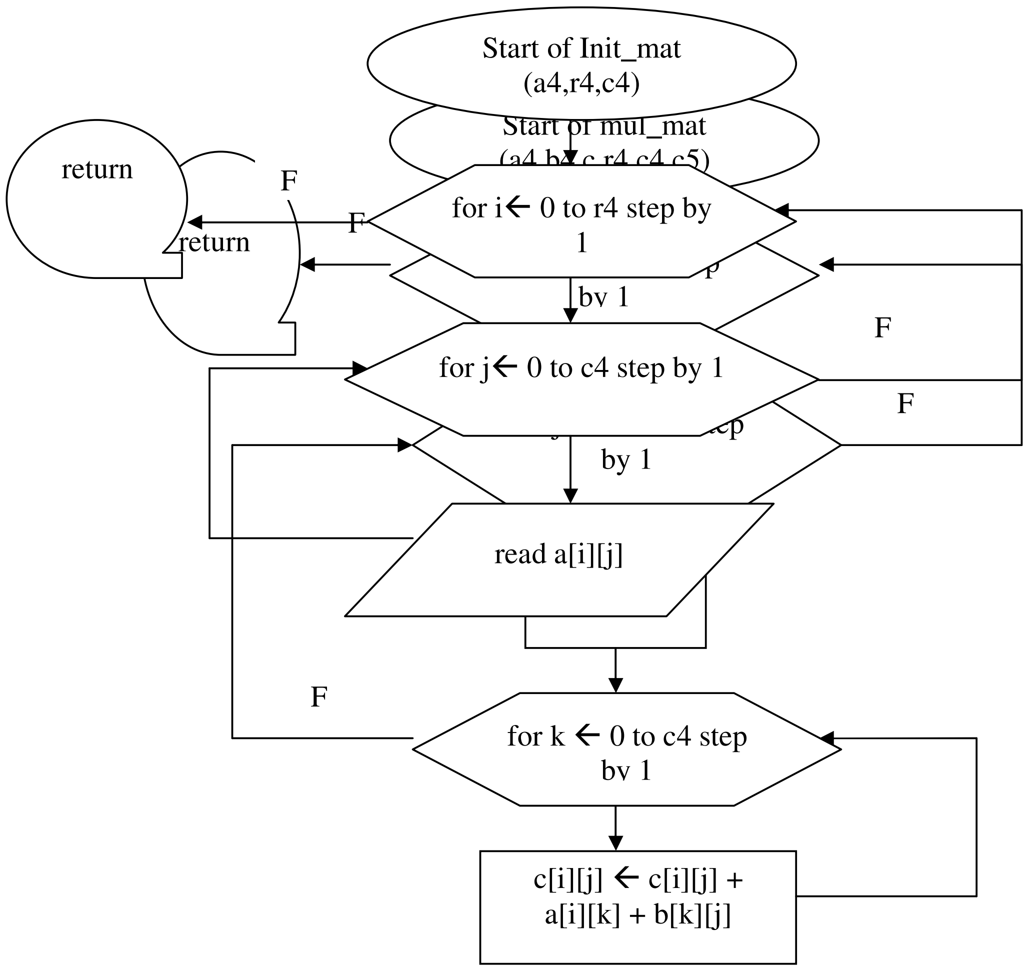
```
print_mat(a4,r4,c4)
    1. for i ← 0 to r4
    2. do  for j ← 0 to c4
    3.          print a[i][j]
    4.     print next_line
```

```
mul_mat(a4,b4,c24.r4,c4,c5)
    1. for i ← 0 to r4
    2.     do for j ← to c5
    3.         do c[i][j] ← 0
    4.             for k ← 0 to c4
    5.                 c[i][j] ← c[i][j] + a[i][k]*b[k][j]
```

**Flow Chart:**


Start
Read r1,r2,c1,c2
r1≠r2 and c1≠ c2
False
Yes
Matrix addition is not possible
A
init_mat(a,r1,c1)
print_mat(a,r1,c1)
init_mat(b,r2,c2)
print_mat(b,r2,c2)
1


1
mul_mat(a,b,c,r1,c1,c2)
print_mat(c,r1,c1)
F
return
for i ← 0 to r4 step by 1
A
F
for j ← 0 to c4 step by 1
Stop
Print a[i][j]


Start of Init_mat (a4,r4,c4)
Start of mul_mat (a4,b4,c,r4,c4,c5)
return
F
F
return
for i← 0 to r4 step by 1
by 1
F
for j← 0 to c4 step by 1
F
by 1
read a[i][j]
F
for k ← 0 to c4 step by 1
c[i][j] ← c[i][j] + a[i][k] + b[k][j]

**PROGRAM :**

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

/* Declaring function prototypes */
void init_mat (int [][10], int, int);
void print_mat (int [][10], int, int);
void mul_mat (int [][10], int [][10], int [][10], int, int, int);

/* Main Function starting */
main()
{
 int r1,r2,c1,c2;
 int a[10][10],b[10][10],c[10][10];
 clrscr();

 /* Giving order of the Matrix - A */
 printf("\n Enter the order of Matrix – A:");
 scanf("%d%d",&r1,&c1);

 /* Giving order of the Matrix - B */
 printf("\n Enter the order of Matrix – B:");
 scanf("%d%d",&r2,&c2);

 if(r1!=c2)
 {
         printf("\n :: Matrix Multiplication is not possible :: ");
         getch();
         exit(0);
  }
 else
  {

   /*  Matrix - A */
   printf("\n Enter the elements of Matrix – A:");
   init_mat(a,r1,c1);
   printf("\n The elements of Matrix – A:");
   print_mat(a,r1,c1);

   /* Matrix - B */
   printf("\n Enter the elements of Matrix – B:");
   init_mat(b,r2,c2);
   printf("\n The elements of Matrix – B:");
   print_mat(b,r2,c2);

   /* Logic for matrix multiplication */
   mul_mat(a,b,c,r1,c1,c2);

   /* Matrix after Multiplication */
   printf("\n The elements of Matrix - C after multiplication of A & B:");
   print_mat(c,r1,c2);
   }
   getch();
}
```

```
/* Function for  two dimensional array initialization */
void init_mat(int mat[][10],int r,int c)
{
    int i,j;
    for(i=0;i<r;i++)
    {
        for(j=0;j<c;j++)
        {
          scanf("%d",&mat[i][j]);
        }
    }
 }

 /* Function for printing elements in Matrix form */
void print_mat(int mat[][10],int r, int c)
{
    int i,j;
    printf("\n");
    for(i=0;i<r;i++)
    {
        for(j=0;j<c;j++)
        {
          printf(" %d ",mat[i][j]);
        }
        printf("\n");
    }
 }

/* Function for matrix multiplication logic */
void mul_mat(int a[][10],int b[][10],int c[][10],int r1,int c1,int c2)
{
    int i,j,k;
    for(i=0;i<r1;i++)
    {
        for(j=0;j<c2;j++)
        {
            /* Initializing Matrix - C with 0's */
            c[i][j] = 0;
              /* logic for Multiplication */
            for(k=0;k<c1;k++)
            {
                 c[i][j] += a[i][k] * b[k][j];
            }
        }
     }
 }
```

**RESULT:**

***Case - 1***

**Input :**

```
     Enter the order of Matrix – A:  2  2
     Enter the order of Matrix – B:  2  2
     Enter the elements of Matrix – A: 1 2 3 4
```

```
 The elements of Matrix – A:
  1    2
  3    4
 Enter the elements of Matrix – B: 1 2 3 4
 The elements of Matrix – B:
 3   2
 4   4
```

**Output:**

```
The elements of Matrix - C after multiplication of A & B:
 7    10
 15    22
```

**Case – 2**

**Input :**

```
Enter the order of Matrix – A:  2  3
Enter the order of Matrix – B:  1  2
```

**Output :**

```
Matrix Multiplication is not possible
```

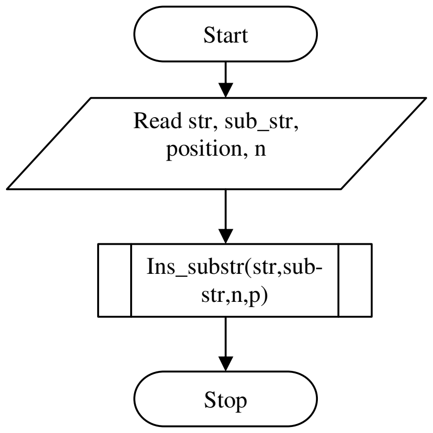
## WEEK-6

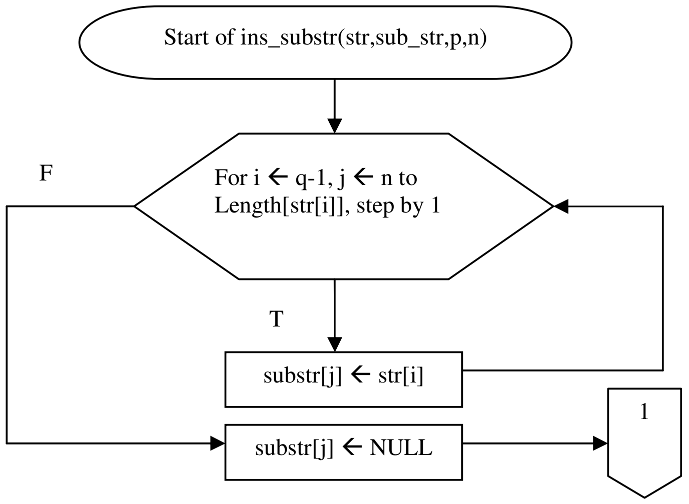
**A). AIM: -** Write A C- Program That Uses Functions To Insert A Sub-String In To A Given Main String From A Given Position.

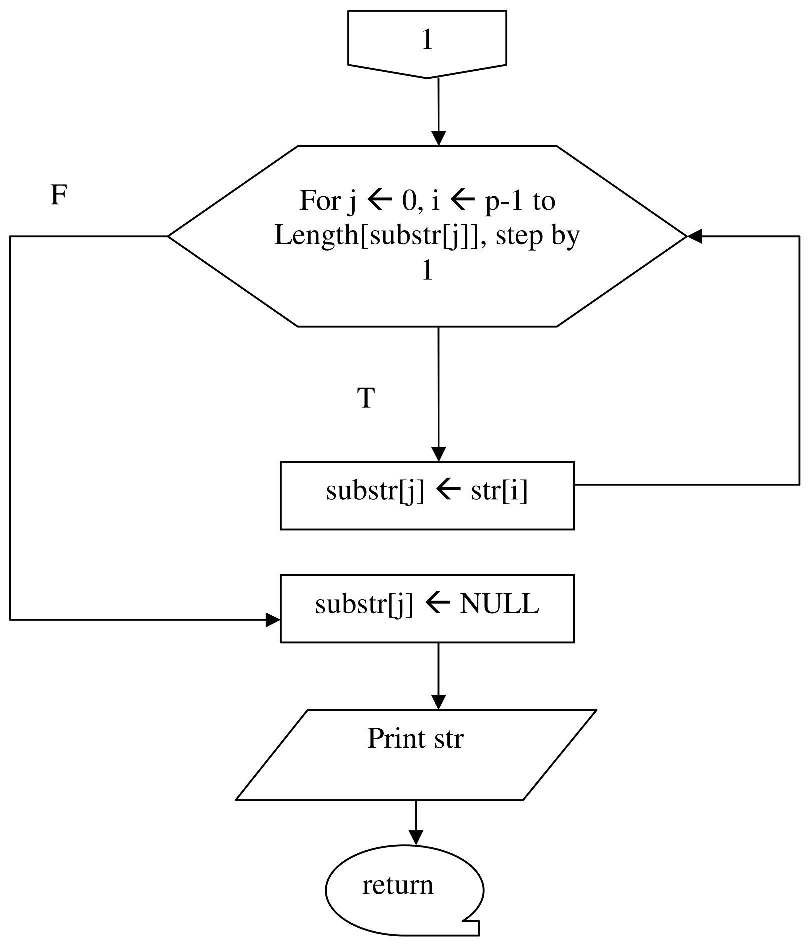
**Algorithm:**

```
1. start
2. read str (string)
3. read n(position), substr (sub string)
4. ins_substr(str,substr,p,n)
5. stop
ins_substr(str,substr,p,n)
1. k ← Length[str]
2. m ← p-1
3. for i ← m and j ← n to k
4.      substr[j] ← str[i]
5.  substr[j] ← NULL
6. for j ← 0 and  i ← m to Length[substr]
7.         str[i]  ← substr[j]
8. str[i] ← NULL
9. print str
```

**Flowchart:**




Start of ins_substr(str,sub_str,p,n)
F
For i ← q-1, j ← n to Length[str[i]], step by 1
T
substr[j] ← str[i]
substr[j] ← NULL
1



**PROGRAM:**

```
/* Declaring C-Libraries */
#include <stdio.h>
#include <conio.h>

/* Declaring function prototypes */
void ins_substr(char [], char [], int, int);

/* main function is starting */
main()
{
int p,n,i,j;
char str[50],substr[50];
clrscr();

/* Initializing character array */
puts("\n Enter the String:");
gets(str);
 fflush(stdin);

/*  Entering the position where you want to insert a substring */
printf("Enter the specific position ");
scanf("%d",&p);
printf("\n Enter the Number of Characters:");
scanf("%d",&n);
fflush(stdin);
puts("\n Enter Sub-String:");
gets(substr);

/* function call to inserting string in main string */
ins_substr(str,substr,p,n);

printf("\n :: End of the main program ::");
getch();
}

/* logic to insert sub string in main string */
void ins_substr(char str[], char substr[], int p, int n)
{
int q,i,j;
q=p-1;
for(i=q,j=n;str[i]!='\0';i++,j++)
  substr[j]=str[i];
substr[j]='\0';
for(j=0,i=q;substr[j]!='\0';j++,i++)
  str[i]=substr[j];
str[i]='\0';
printf("\n The string after inserting substring :");
puts(str);
 }
```

**RESULT:**
**Case - 1**
**Input :**

```
Enter the String: HELO  WORLD
Enter the specific position : 3
Enter the Number of Characters: 1
Enter Sub-String: L
```

**Output :**

```
The string after inserting substring : HELLO WORLD
 :: End of the main program ::
```

**Case - 2**
**Input :**

```
Enter the String: HELLO
Enter the specific position : 5
Enter the Number of Characters: 5
Enter Sub-String: WORLD
```

**Output :**

```
The string after inserting substring : HELLO WORLD
  :: End of the main program ::
```

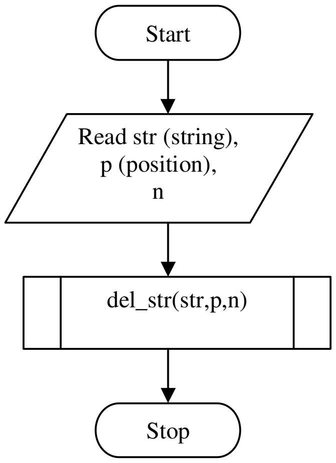
**AIM: -** Write A C- Program That Uses Functions To Delete N – Charactres From A Given Position In A Given String.

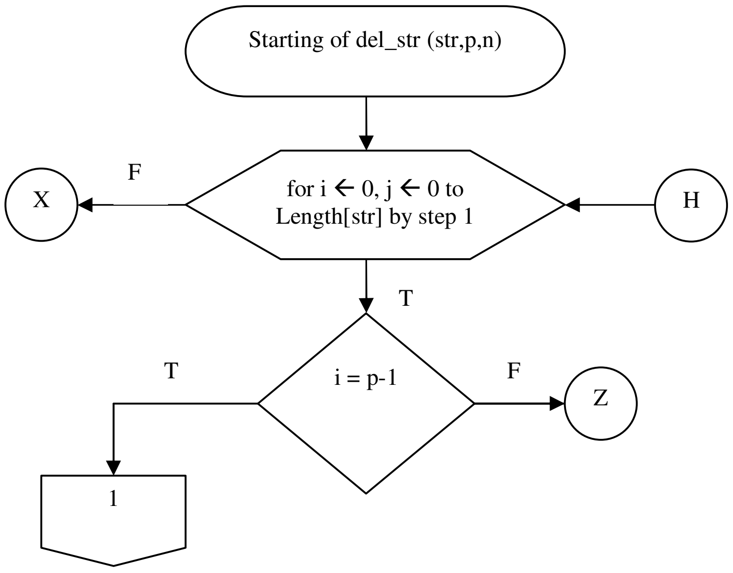
**Algorithm:**

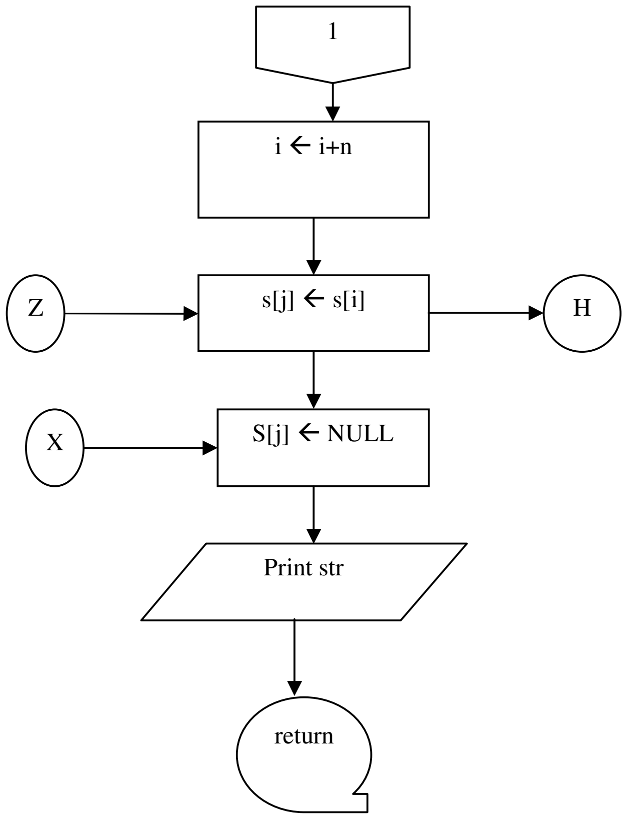
```
1. start
2. read str(main string)
3. read p (position)
4. read n (number of characters to delete)
5. del_str(str,p,n)
6. stop
```

```
del_str(str,p,n)
1. for i ← 0 , j ← 0 to Length[str]
2.        do if i = p-1
3.               i ← i + n
4.        str[j] ← str[i]
5. str[j] ← NULL
6. print str
```

**Flow Chart:**




Starting of del_str (str,p,n)
for i ← 0, j ← 0 to Length[str] by step 1
F
X
H
T
i = p-1
T
F
Z
1



**PROGRAM:**

```
//* Declaring C - Libraries */
#include <stdio.h>
#include <conio.h>

/* declaring prototype of function */
void del_str(char [],int, int);

/* Main function starting */
main()
{
 int n,p;
 char str[30];
 clrscr();
 printf("\n Enter the String::");
 gets(str);
 fflush(stdin);
 printf("\n Enter the position from where the characters are to be deleted:");
 scanf("%d",&p);
 printf("\n Enter Number of characters to be deleted:");
 scanf("%d",&n);

 /* function call to deletion of n-characters */
 del_str(str,p,n);
 printf("::End of the Main program::");
 getch();
}

 /* function call to Logic of delete n-characters from string */
 void del_str(char str[],int p, int n)
```

```
{
   int i,j;
   for(i=0,j=0;str[i]!='\0';i++,j++)
     {
        if(i==(p-1))
         {
           i=i+n;
         }
    str[j]=str[i];
     }
   str[j]='\0';

  /* the string after deletion */
  puts(" The string after deletion of characters::");
  puts(str);
 }
```

**RESULT:**
**Case - 1**
**Input :**

```
Enter the String: ABCD EFGH IJKL
Enter the position from where the characters are to be deleted: 5
Enter Number of characters to be deleted: 4
```

**Output :**

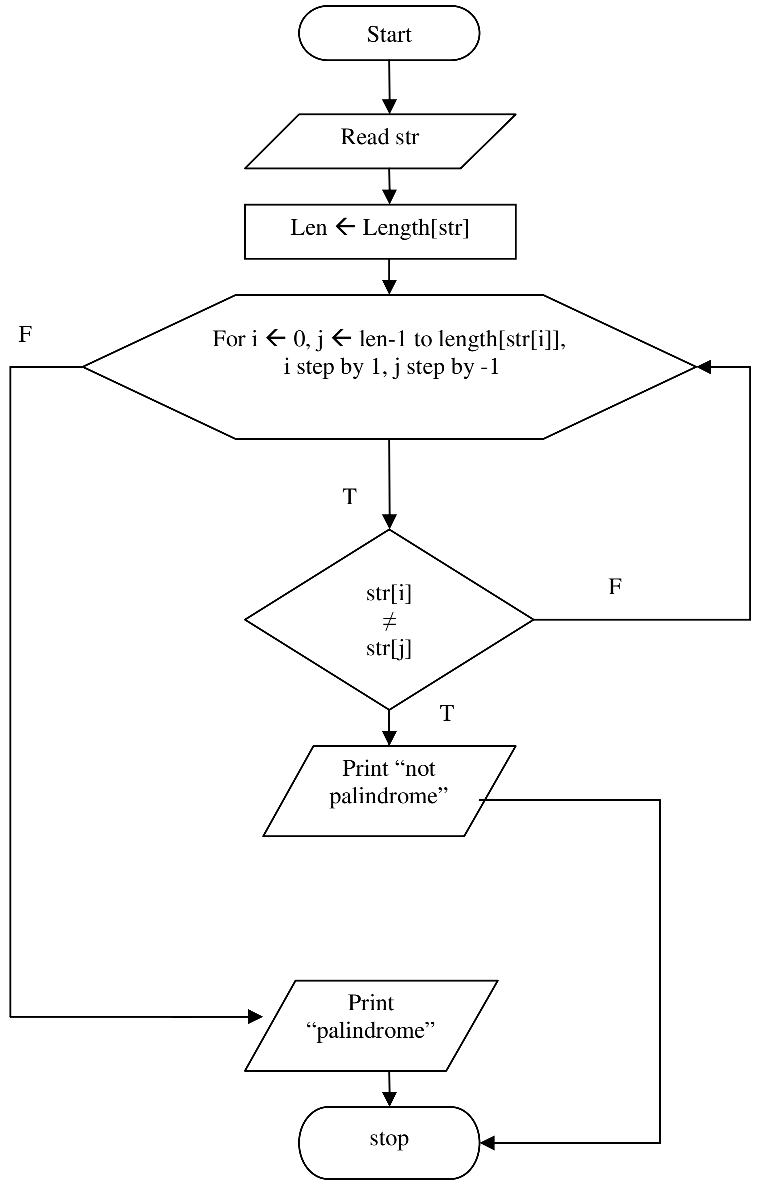
```
The string after deletion of characters:: ABCD IJKL
:: End of the main program ::
```

**B) AIM:** Write A C- Program To Determine If The Given String Is A Palindrome Or Not

**Algorithm:**

```
1. Start
2. read str (string)
3. len ← Length[str]
4. for i ← 0 (increment step), j ← len-1 (decrement step) to Length[str]
5.     do str[i] ≠ str[j]
6.                 print " not palindrome"
7.                 stop
8. print "palindrome"
9. stop
```

**Flowchart:-**



**PROGRAM:**

```
/* Declaring C-library */
#include <stdio.h>
#include <conio.h>

/* Main function definition */
main()
{
 int i,n,j,len=0;
 char str[30];
 clrscr();
 printf("\n Enter String:");
 gets(str);

 /* logic to checking string for palindrome  */
 for(i=0;str[i]!='\0';i++)
 len++;
 printf("\n The length of the string is %d",len);
 for(i=0,j=len-1;str[i]!='\0';i++,j--)
 {
  if(str[i]!=str[j])
  {
   printf("\n :The given string is not a palindrome:");
   getch();
   exit(0);
   }
 }
 printf("\n :the given string is palindrome:");
 getch();
 }
```

**RESULT:**
**Case - 1**
**Input :**

```
     Enter the String: MALAYALAM
     The length of the string is 9
```
**Output :**
```
     :the given string is palindrome:
```

**Case - 2**
**Input :**
```
     Enter the String: ABC
     The length of the string is 3
```

**Output :**
```
  The given string is not a palindrome:
```

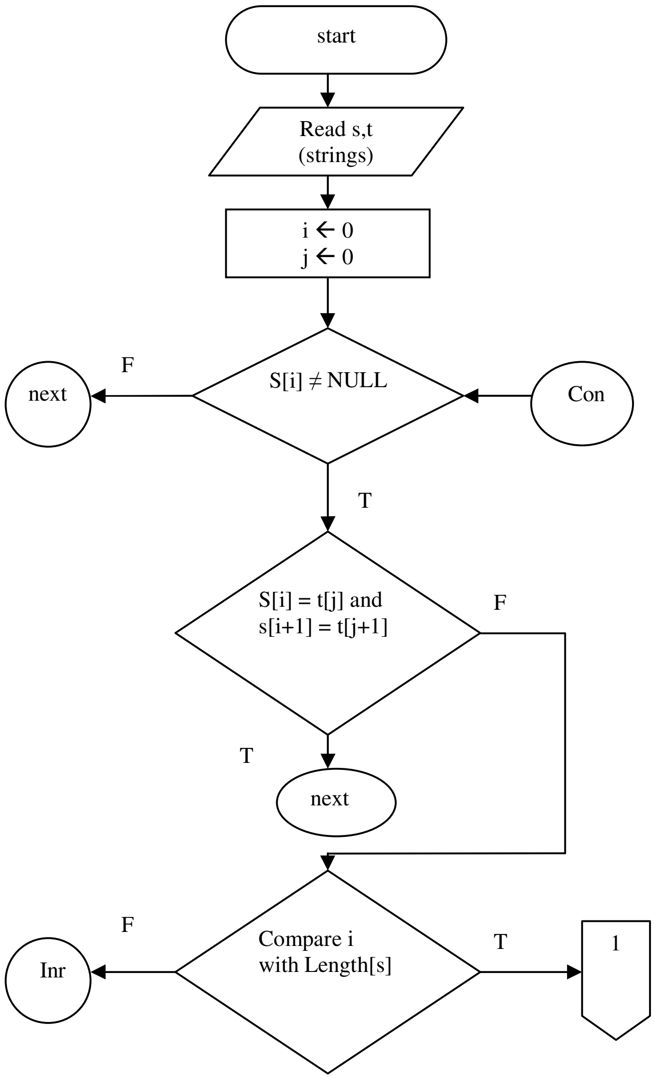
## WEEK-7

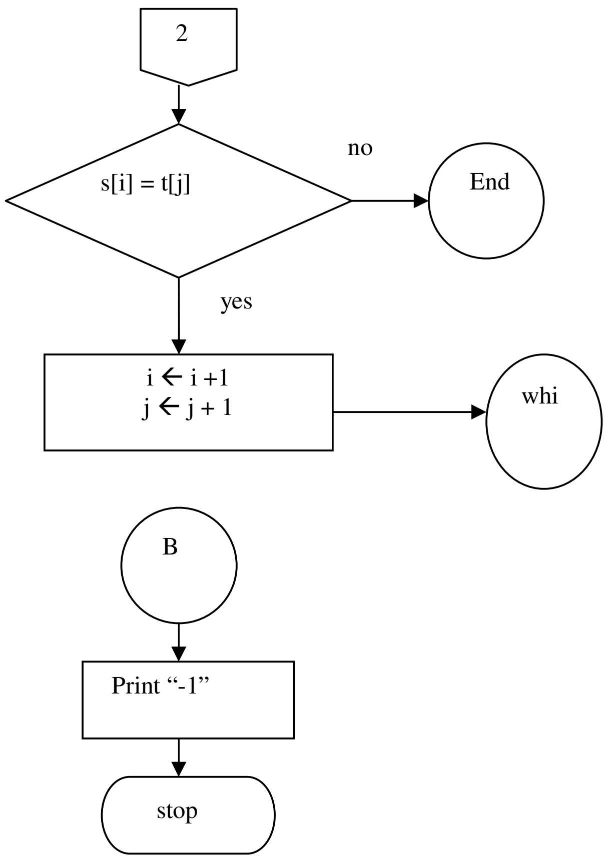
**A) AIM: -** Write A C- Program That Displays The Positions Or Index Of In The String – S Where The String – T Begins, Or -1 If String – S Doesnot Contain String – T.

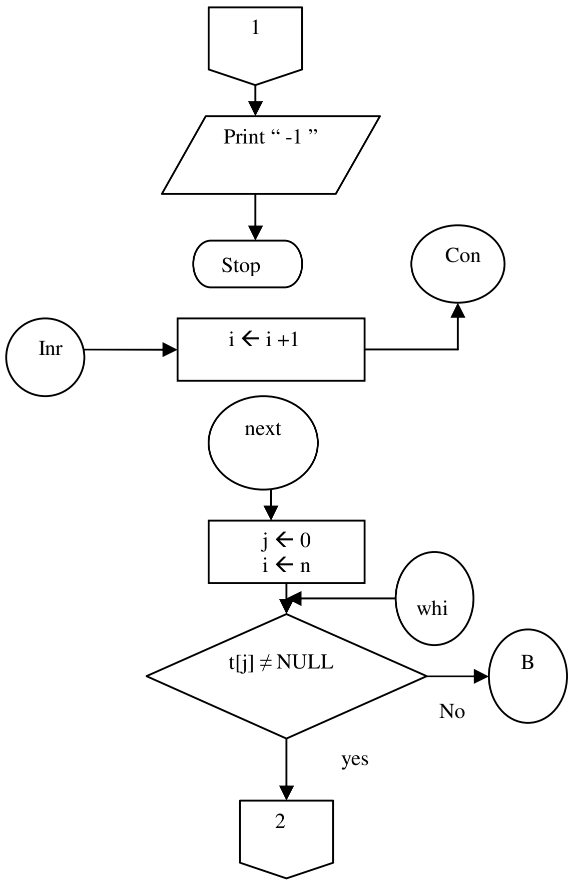
**Algorithm:**

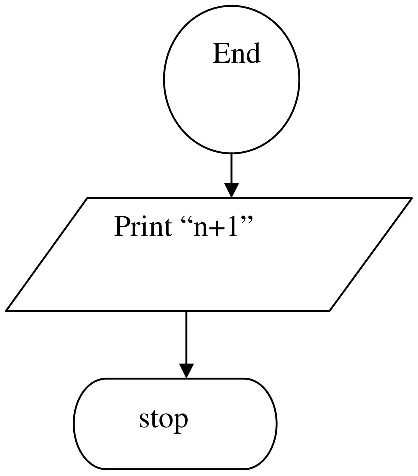
```
1.  Start
2.  read s, t
3.  while Length[str]
4.      do  if s[i] == t[j] and s[i+1] == t[j+1]
5.                  then break
6.          if i = Length[s]
7.                then   print "-1"
8.                            goto end
9.  j ← 0, n ← i
10. while Length[t[j]]
11.      do if s[i] == t[j]
12.              then     i ← i+1
13.                         j ← j+1
14.            else
15.                  print "-1"
16.                   goto end
17. print "n+1"
18. end:
19. stop
```

**Flowchart:**









**PROGRAM:**

```
/* declaring C-Libraries */
#include <stdio.h>
#include <conio.h>
#include <string.h>

/* main function is starting */
```

```
main()
{
 int i,j,n;
 char s[40],t[40];
 clrscr();
 printf("\n Enter the string:");
 gets(s);
 fflush(stdin);
 printf("\n Enter the sub string:");
 gets(t);

 /* logic to search sub string */
 i=0;
 j=0;
 while( s[i] != '\0')
  {
     if((s[i] == t[j]) && (s[i+1] == t[j+1]))
              break;
     if(i==strlen(s))
      {
         printf("-1");
         getch();
         exit(0);
      }
      i++;
  }
  j=0;
  n=i;
  while(t[j] != '\0')
  {
    if(s[i] == t[j])
     {
         i++;
         j++;
     }
     else
      {
         printf("-1");
         getch();
         exit(0);
      }
  }
  printf("\n The string is found at %d",n+1);
  getch();  }
```

**RESULT:**

**Input :**

```
     Enter the String: HELLO WORLD
     Enter substring : WORLD
```

**Output :**

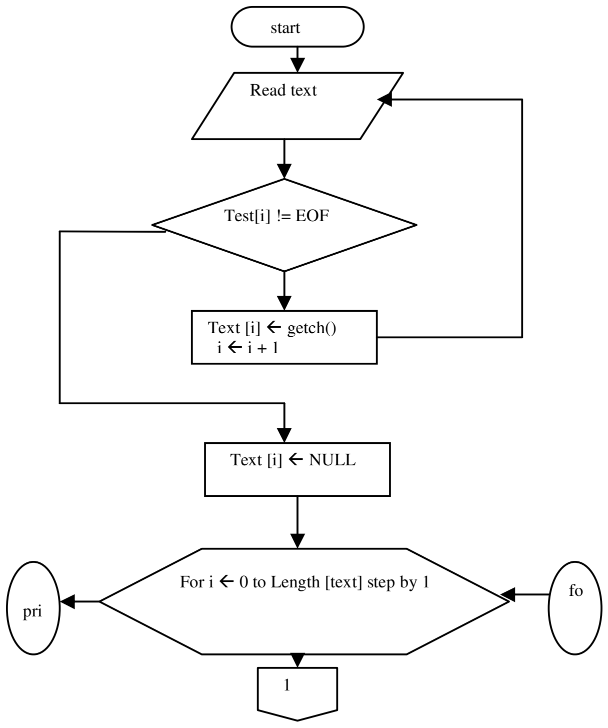
```
    The String is found at 6
```

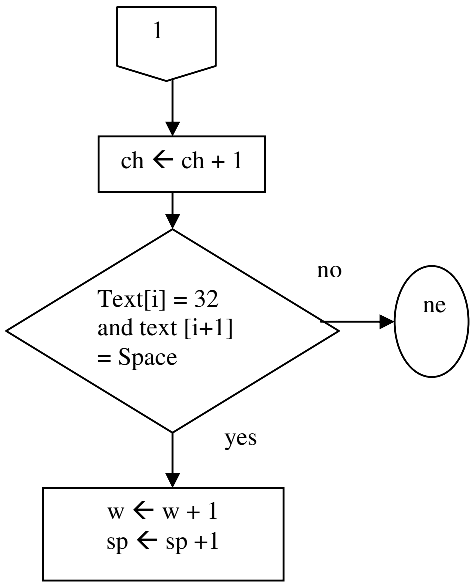
**AIM: -** Write A C- Program To Count The Lines, Words, Characters In A Given Text

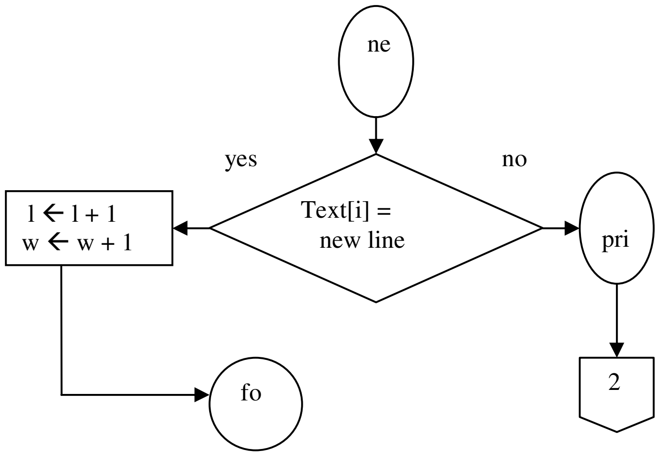
**Algorithm:**

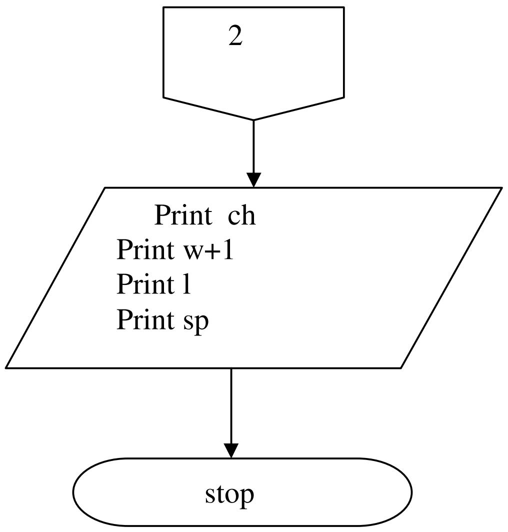
```
1.  start
2.  read text
3.  while text[i] != EOF
4.     do i ← i+1
5.  print  i
6.  for i ← 0 to Length[text]
7.      do ch++
8.            if text[i]  = 32 and text[i+1] ≠ ' '
9.                 then     w++
10.                         sp++
11.          if text[i] = '\n'
12.                then      l++
13.                          w++
14. print ch
15. print w+1
16. print l
17. print sp
```

**Flow Chart:**




1
ch ← ch + 1
Text[i] = 32
and text [i+1]
= Space
no
ne
yes
w ← w + 1
sp ← sp +1


ne
yes
no
Text[i] =
new line
l ← l + 1
w ← w + 1
pri
fo
2


2
Print ch
Print w+1
Print l
Print sp
stop

**PROGRAM:**

```
/* Declaring C - Libraries */
#include <conio.h>
#include <stdio.h>
/* Main function is starting */
main()
{
 char text[200];
 int i,l,ch,w,sp;
 clrscr();

 /* logic to count number of characters */
 i=0;
 printf("\n Enter lines of text and press ^Z");
 while((text[i]=getchar())!=EOF)
 {
  i++;
  }
 printf("\n The number of characters is %d",i);

 /* logic to count the lines, words and spaces in text */
 text[i]='\0';
 l=0;
 ch=w=sp=0;
 for(i=0;text[i]!='\0';i++)
 {
    ch++;
    if(text[i]==32 && text[i+1] != ' ')
      {
       w++;
       sp++;
       }
      if(text[i] == '\n')
       {
          l++;
          w++;
       }
   }
 printf("\n Total size of the text : %d",ch);
 printf("\n Number of Words : %d",w+1);
 printf("\n Number of Lines : %d",l);
 printf("\n Number of Spaces : %d",sp);
 getch();
}
```

**Result:**

```
 Enter lines of text and press ^Z”” ABCD EFGH IJKL MNOP

Total size of the text: 14
 Number of Words:8
 Number of Lines:1
 Number of Spaces:4
```

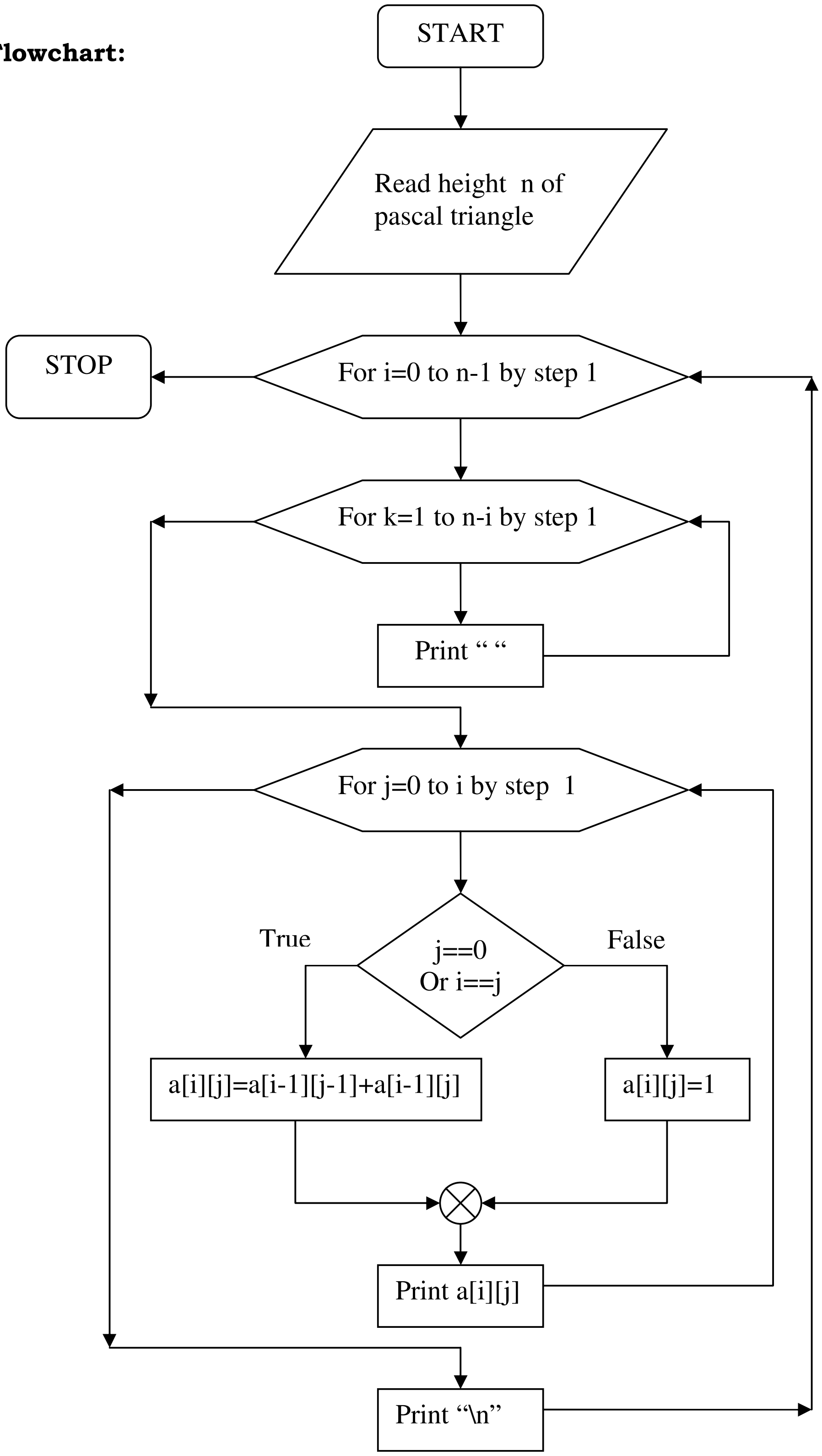
## WEEK 8

**A)Aim:** Write an algorithm, flowchart and program to generate Pascal's triangle.

**Algorithm:**

```
Step 1: Start

Step 2: Read the height of the pascal triangle

Step 3: for i := 0  to n-1 do
        Step 3.1: for k:= 1 to n-i do
                Step 3.1.1:   Print blankspace
                Step 3.1.2:   k := k+1

        Step 3.2: for j:= 0 to i do
                Step 3.2.1:   if ( j==0) or (i==j) then
                        Step 3.2.1.1: a[i,j] := 1
                Step 3.2.2:   else
                        Step 3.2.2.1: a[i,j] = a[i-1,j-1] + a[i-1,j]
                Step 3.2.3:   End if
                Step 3.2.4:   print a[i,j]
                Step 3.2.5:   j := j+1

        Step 3.3: i := i+1

Step 4: Stop
```

**Flowchart:**


START
Read height n of pascal triangle
STOP
For i=0 to n-1 by step 1
For k=1 to n-i by step 1
Print " "
For j=0 to i by step 1
True
j==0
Or i==j
False
a[i][j]=a[i-1][j-1]+a[i-1][j]
a[i][j]=1
Print a[i][j]
Print "\n"

**Program:**

```
#include <stdio.h>
#include <conio.h>
main()
{
        int a[10][10],i,j,k,n;
        clrscr();
        printf("Enter the height of the pascal traingle");
        scanf("%d",&n);
        for(i=0;i<n;i++)
        {
                for(k=1;k<=n-i;k++)
                        printf(" ");
                for(j=0;j<=i;j++)
                {
                        if(j==0  | | i==j)
                                a[i][j]=1;
                        else
                                a[i][j]=a[i-1][j-1]+a[i-1][j];
                        printf("%d ",a[i][j]);

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

**Result:**

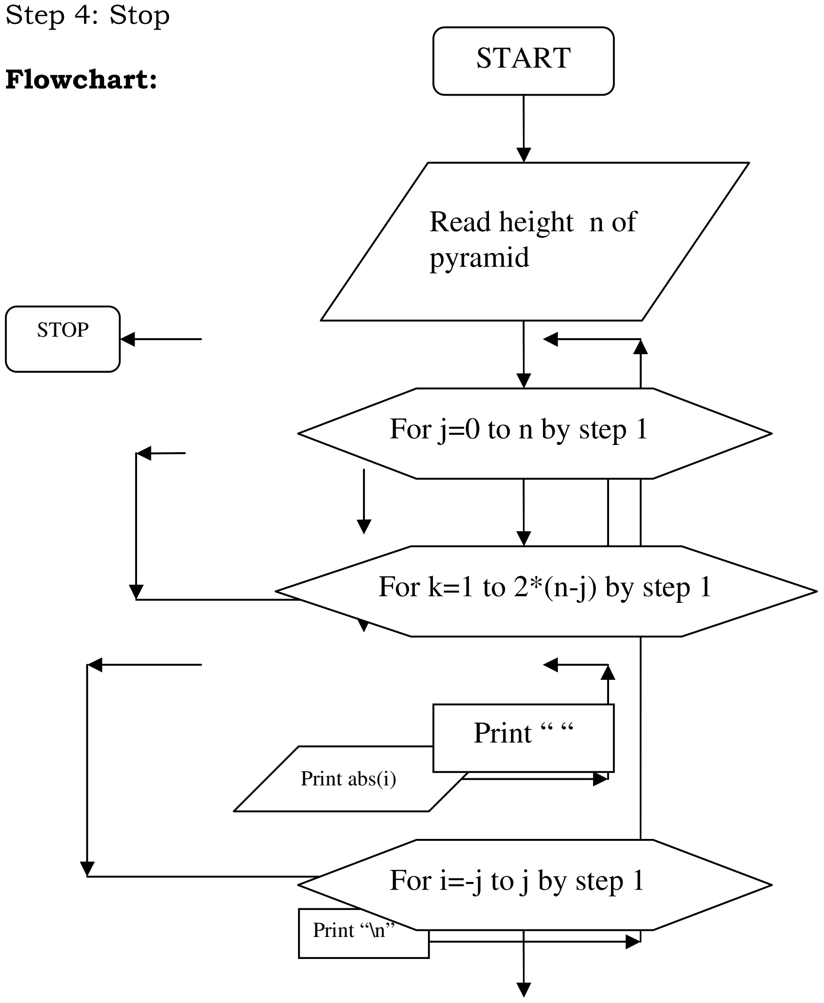
```
Enter height of pascal triangle: 4
                        1
                1       2       1

        1       3       3       1
```

**b) Aim:** Write an algorithm, flowchart and program to construct a pyramid of numbers.

**Algorithm:**

```
Step 1: Start

Step 2: Read height n of the pyramid

Step 3: for j := 0  to n do

    Step 3.1: for k:= 1 to 2*(n-j) do
        Step 3.1.1:   Print blankspace
        Step 3.1.2:   k := k+1

    Step 3.2: for i:= -j to j do
        Step 3.2.1: print abs(i)
        Step 3.2.2: i := i+1

    Step 3.3: j := j+1

Step 4: Stop
```

**Flowchart:**



**Program:**

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

main()
{
        int i,j,k,n;
        clrscr();
        printf("Enter the height of the pyramid");
        scanf("%d",&n);
        for(j=0;j<=n;j++)
        {
                for(k=1;k<=2*(n-j);k++)
                        printf(" ");
                for(i=-j;i<=j;i++)
                        printf("%d ",abs(i));
                printf("\n");
        }
}
```

**Result:**

```
Enter the height of the pyramid: 2
        1
2               2
```

## WEEK 9

**A) Aim:** Write an algorithm, flowchart and a C program to read in two numbers, x and n, and then compute the sum of this geometric progression:

$1+x+x^2+x^3+x^4+\ldots\ldots\ldots+x^n$

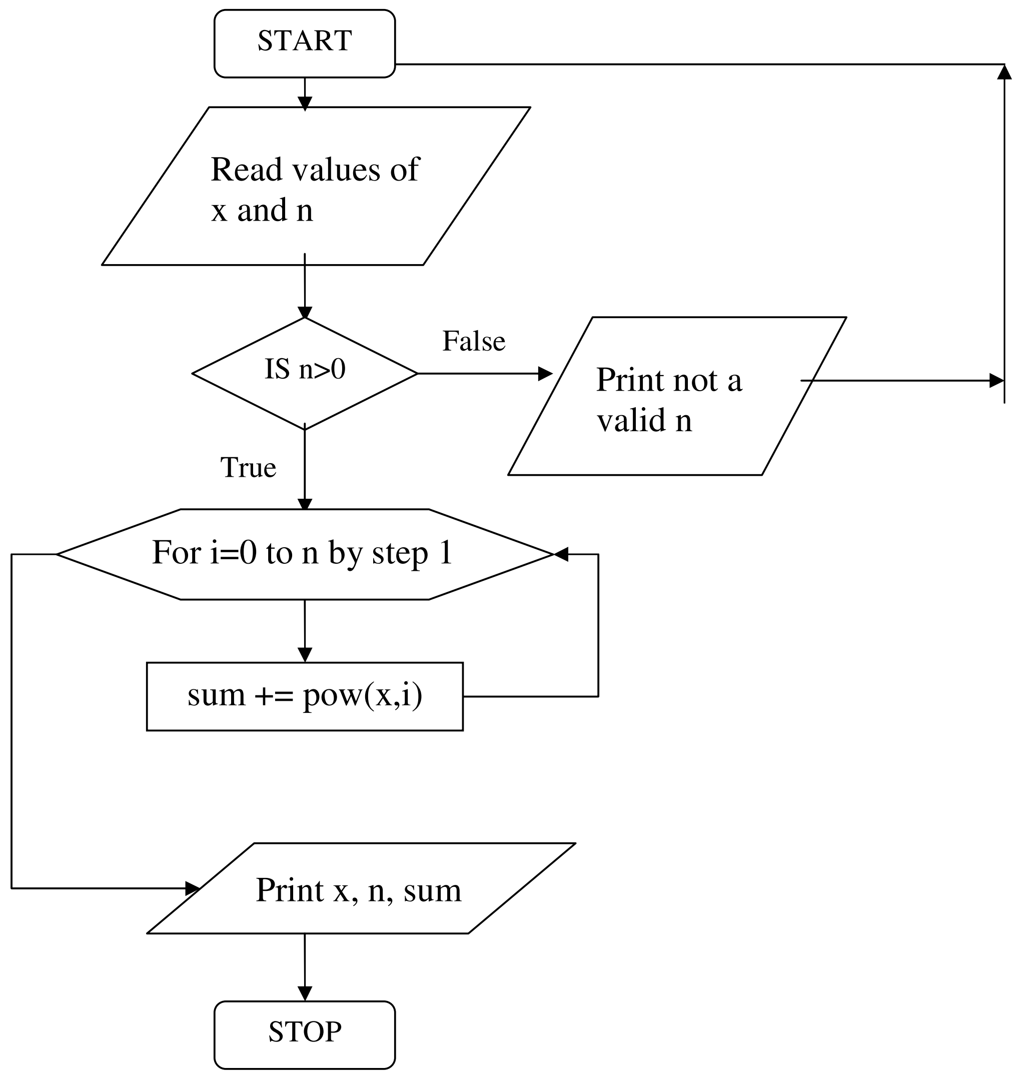
Print x, n, the sum

Perform error checking. For example, the formula does not make sense for negative exponents- if n is less than 0. Have your program print an error message if n<0, then go back and read in the next pair of numbers of without computing the sum. Are any values of x also illegal ? If so, test for them too.

**Algorithm :**

```
Step 1: Start

Step 2: rept:

Step 3: Read values for x and n

Step 4: if n > 0 then
        Step 4.1: for i := 0  to n do
                Step 4.1.1: sum := sum +pow(x,i)
                Step 4.1.2: i := i+1
        Step 4.2 : print x, n and sum

Step 5: else
        Step 5.1: print not a valid n value
        Step 5.2: goto rept

Step 6: End if

Step 7: Stop
```

**Flowchart:**



**Program:**

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
main()
{
        int x,n,sum=0,i;
        start:
        clrscr();
        printf("enter the values for x and n");
        scanf("%d%d",&x,&n);
        if(n>0)
        {
                for(i=0;i<=n;i++)
                {
                        sum = sum+pow(x,i);
                }
                printf(“ x is - %d, n is -%d \n”,x,n);
                printf("The sum of the geometric progression is:%d",sum);
        }
        else
        {
                printf("not a valid n:%d value",n);
                getch();
                goto start;
        }
}
```

**Result:**

```
Enter the values for x and n  1 1
The sum of the geometric progression: 2
```

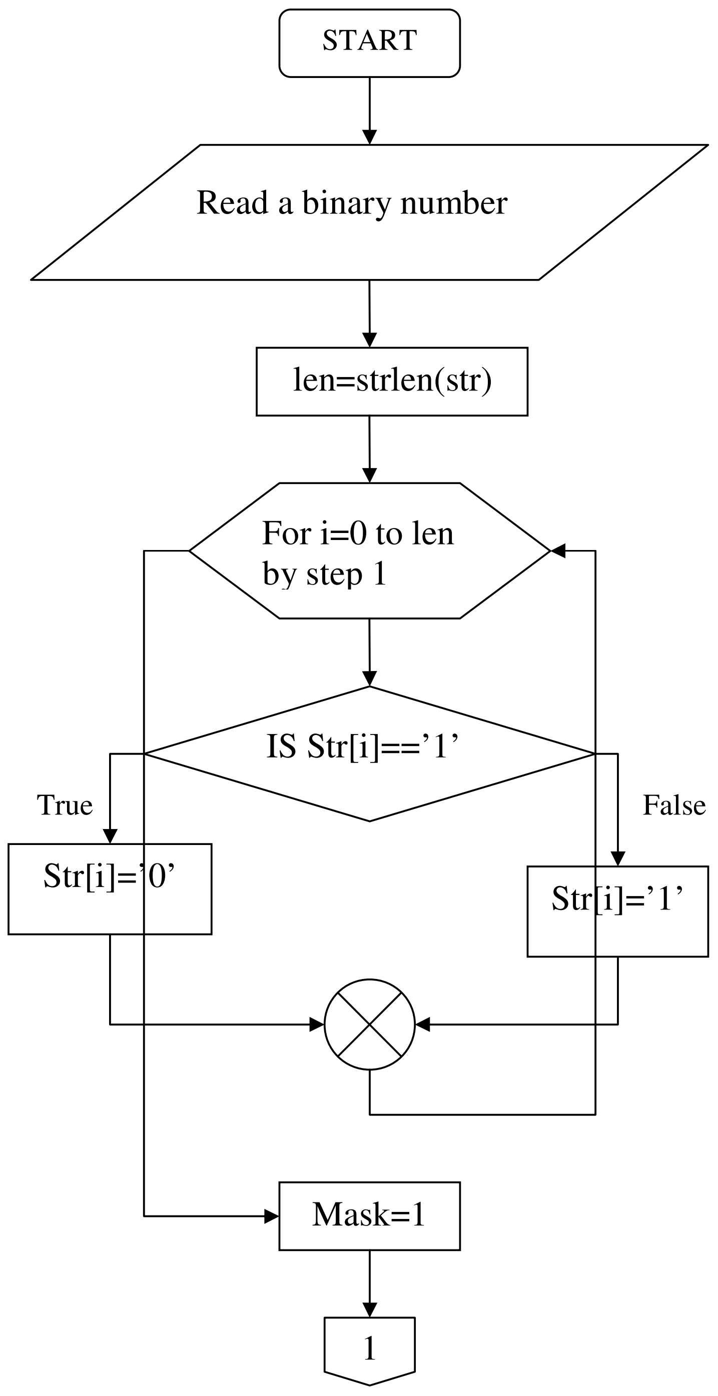
## WEEK 10

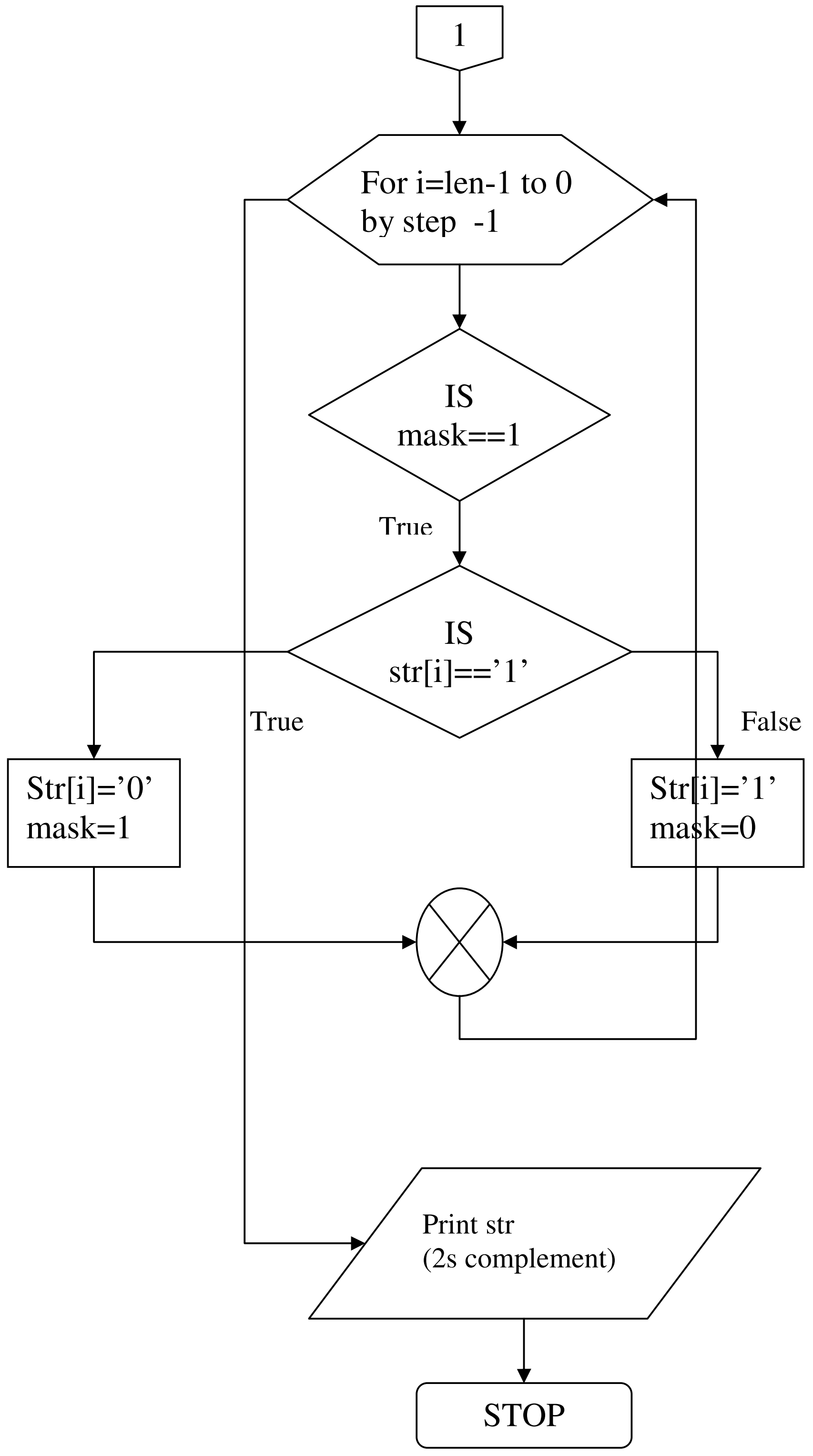
**A) Aim:** Write an algorithm, flowchart and a C program for finding the 2’s complement of a binary number

**Algorithm:**

```
Step 1: Start

Step 2: Read the binary number

Step 3: copying the binary number to strdp

Step 4: len := strlen(str)

Step 5: for i := 0 to  len-1 do
    Step 5.1: if str[i] == ‘1’ then
        Step 5.1.1:  str[i] == ‘0’
    Step 5.2: else
        Step 5.2.1: str[i] == ‘1’
    Step 5.3: i := i+1

Step 6: mask := 1

Step 7: for i := len-1 to 0 do
    Step 7.1: if mask == 1 then
        Step 7.1.1: if str[i] == ‘1’ then
            Step 7.1.1.1: str[i] := ‘0’
            Step 7.1.1.2: mask := 1
        Step 7.1.2: else
            Step 7.1.2.1: str[i] := ‘1’
            Step 7.1.2.2: mask := 0
        Step 7.1.3: End if
    Step 7.2: End if

Step 8: print the 2’s complement

Step 9: Stop
```

**Flowchart:**


START
Read a binary number
len=strlen(str)
For i=0 to len by step 1
IS Str[i]=='1'
True
False
Str[i]='0'
Str[i]='1'
Mask=1
1



**Program:**

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

```
#include <conio.h>
#include <string.h>
main()
{
        char str[32],strdp[32];
        int mask,i;
        clrscr();
        printf("Enter a binary number:");
        scanf("%s",str);
        strcpy(strdp,str); /* creating duplicate copy */
        for(i=0;i<strlen(str);i++)      /* computing 1's complement */
        {
                if(str[i]=='1')
                        str[i]='0';
                else
                        str[i]='1';
        }
        printf("1\'s complement of %s is %s\n",strdp,str);
        mask=1;
        for(i=strlen(str)-1;i>=0;i--)   /* computing 2's complement */
        {
                if(mask==1)
                {
                        if(str[i]=='1')
                        {
                                str[i]='0';
                                mask=1;
                        }
                        else
                        {
                                str[i]='1';
                                mask=0;
                        }
                }
        }
        printf("2\'s complement of %s is %s",strdp,str);
}
```

**Result:**

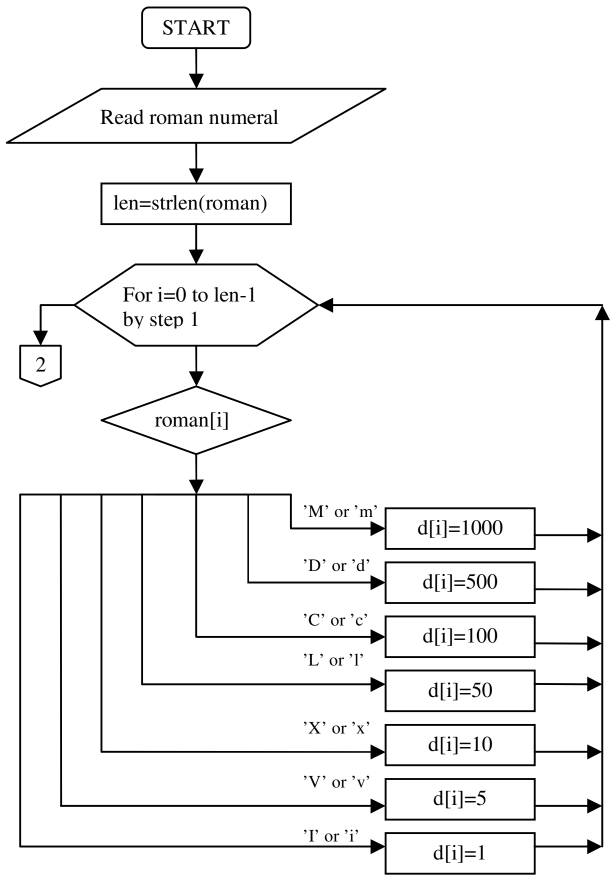
```
Enter a binary number: 1111
1's complement is: 0000
2's complement is: 0001
```

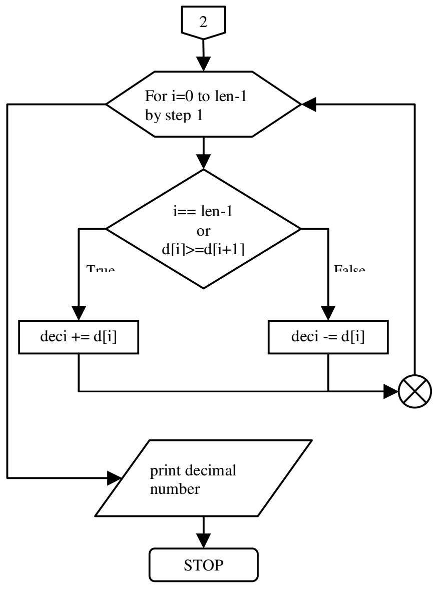
**B) Aim:** Write an algorithm, flowchart and a C program to convert a Roman numeral to its decimal equivalent.

**Algorithm:**

```
Step 1: Start
Step 2: Read the roman numeral
Step 3: len := strlen(roman)
Step 4: for i := 0 to len-1 do
        Step 4.1: switch(roman[i])
                Step 4.1.1: case 'm':
                Step 4.1.2: case 'M':
                                Step 4.1.2.1: d[i]:=1000
                Step 4.1.3: case 'd':
                Step 4.1.4: case 'D':
                                Step 4.1.4.1: d[i]:=500
                Step 4.1.5: case 'c':
                Step 4.1.6: case 'C':
                                Step 4.1.6.1: d[i]:=100
                Step 4.1.7: case 'l':
                Step 4.1.8: case 'L':
                                Step 4.1.8.1: d[i]:=50
                Step 4.1.9: case 'x':
                Step 4.1.10: case 'X':
                                Step 4.1.10.1: d[i]:=10
                Step 4.1.11: case 'v':
                Step 4.1.12: case 'V':
                                Step 4.1.12.1: d[i]:=5
                Step 4.1.13: case 'i':
                Step 4.1.14: case 'I':
                                Step 4.1.14.1: d[i]:=1

Step 5: for i :=0 to len-1 do
        Step 5.1: if (i==len-1) or (d[i]>=d[i+1]) then
                Step 5.1.1: deci += d[i]
        Step 5.2: else
                Step 5.2.1: deci -= d[i]
Step 6: print the decimal equivalent of roman numeral
Step 7: Stop
```

**Flowchart:**





**Program:**

```
#include <stdio.h>
#include <conio.h>
main()
{
char roman[30];
int deci=0;
int len,i,d[30];
clrscr();
printf("The following table shows the Roman equivalent to decimal\n");
printf("Decimal:.........Roman\n");
printf("%5d...........%3c\n",1,'I');
printf("%5d...........%3c\n",5,'V');
printf("%5d...........%3c\n",10,'X');
printf("%5d...........%3c\n",50,'L');
printf("%5d...........%3c\n",100,'C');
printf("%5d...........%3c\n",500,'D');
printf("%5d...........%3c\n",1000,'M');
printf("Enter a Roman numeral:");
scanf("%s",roman);
len=strlen(roman);
for(i=0;i<len;i++)
{
        switch(roman[i])
        {

                case 'm':
                case 'M':  d[i]=1000;          break;
                case 'd':
                case 'D':  d[i]= 500;   break;
                case 'c':
                case 'C':  d[i]= 100;   break;
                case 'l':
                case 'L':  d[i]= 50;    break;
                case 'x':
                case 'X':  d[i]= 10;    break;;
                case 'v':
                case 'V':  d[i]= 5;     break;
                case 'i':
                case 'I':  d[i]= 1;
        }
}
for(i=0;i<len;i++)
{
        if(i==len-1 || d[i]>=d[i+1])
                deci += d[i];
        else
                deci -= d[i];
}
printf("The Decimal equivalent of Roman numeral %s is  d", roman, deci);
}
```

**Result:**

```
Enter a Roman numeral: L
The Decimal equivalent of Roman numeral L is  :50
```

## WEEK 11

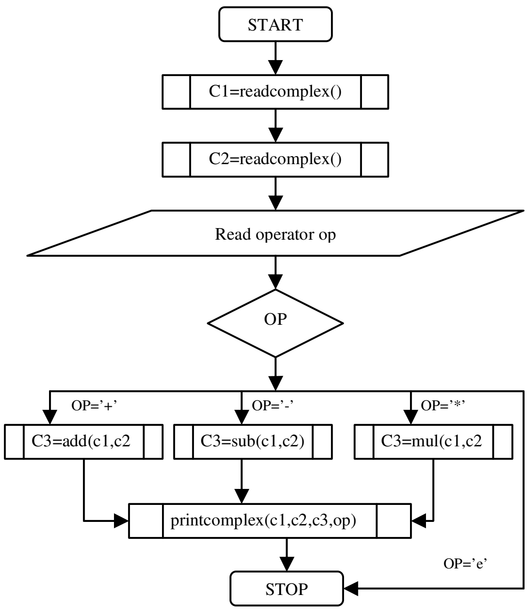
**Aim:** Write an algorithm, flowchart and a C program that uses functions to perform the following operations:

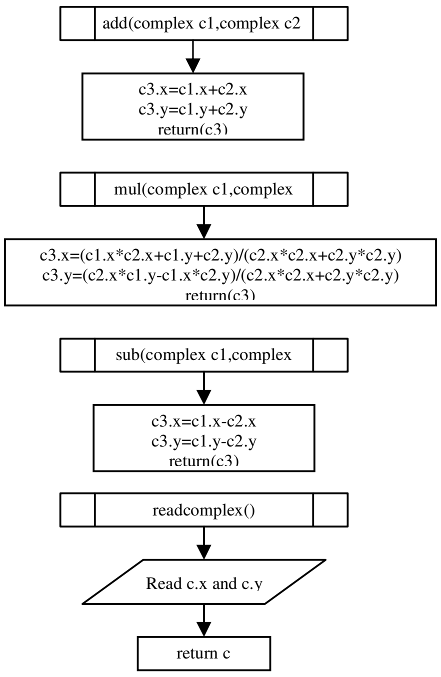
i) Reading a complex number
ii) Writing a complex number
iii) Addition of two complex numbers
iv) Multiplication of two complex numbers

**Algorithm:**

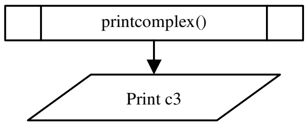
```
Step 1:  Start
Step 2: Read the first complex number by calling readcomplex()
Step 3: Read the second complex number by calling readcomplex()
Step 4: read the operator op
Step 5: switch(op)
        Step 5.1: case '+': c3 := add(c1,c2)
        Step 5.2: case '-' : c3 := sub(c1,c2)
        Step 5.3: case '*': c3 := mul(c1,c2)
        Step 5.4: case 'e': program end
Step 6: print c3 by calling printcomplex(c1,c2,c3,op)
Step 7: Stop

add (c1,c2)
step 1: c3.x := c1.x + c2.x
step 2: c3.y := c1.y + c2.y
step 3: return c3

sub(c1,c2)
step 1: c3.x := c1.x - c2.x
step 2: c3.y := c1.y - c2.y
step 3: return c3

mul(c1,c2)
step 1: c3.x :=(c1.x*c2.x+c1.y+c2.y)/(c2.x*c2.x+c2.y*c2.y)
step 2: c3.y :=(c2.x*c1.y-c1.x*c2.y)/(c2.x*c2.x+c2.y*c2.y)
step 3: return c3
```

**Flowchart:**





**Program:**



```
#include <stdio.h>
#include <stdlib.h>
struct compl
{
        int x;
        int y;
};
typedef struct compl complex;
main()
```

```
{
        complex c1,c2,c3;
        complex  add(complex c1,complex c2);
        complex  sub(complex c1,complex c2);
        complex  mul(complex c1,complex c2);
        complex readcomplex();
        complex printcomplex(complex c1,complex c2,complex c3,char op);
        char op;
        clrscr();
        printf("Reading the first complex number\n");
        c1=readcomplex();
        printf("Reading the second complex number\n");
        c2=readcomplex();
        printf("Enter + for addition \n"
              "Enter * for multiplication\n"
              "Enter - for subtraction\n"
              "Enter e for exit:");
        fflush(stdin);
        op=getche();
        switch(op)
        {
                case '+': c3=add(c1,c2);
                          break;
                case '-': c3=sub(c1,c2);
                          break;
                case '*': c3=mul(c1,c2);
                          break;
                case 'e': exit(0);
        }
        printcomplex(c1,c2,c3,op);
        getch();
}


complex  add(complex c1,complex c2)
{
        complex c3;
        c3.x=c1.x+c2.x;
        c3.y=c1.y+c2.y;
        return(c3);
}
complex sub(complex c1,complex c2)
{
        complex c3;
        c3.x=c1.x-c2.x;
        c3.y=c1.y-c2.y;
        return(c3);
}
complex mul(complex c1,complex c2)
{
        complex c3;
        c3.x=(c1.x*c2.x+c1.y+c2.y)/(c2.x*c2.x+c2.y*c2.y);
        c3.y=(c2.x*c1.y-c1.x*c2.y)/(c2.x*c2.x+c2.y*c2.y);
        return(c3);
}
complex readcomplex()
```

```
{
        complex c;
        printf("Enter the values of x and y of a complex number");
        scanf("%d%d",&c.x,&c.y);
        return(c);
}
complex printcomplex(complex c1,complex c2,complex c3,char op)
{
        printf("\n(%d+i%d)%c(%d+i%d)=%d+i(%d)",c1.x,c1.y,op,c2.x,c2.y,c3.x,c3.y);
}
```

**Result:**

```
Reading the first complex number: 2 + 2i
Reading the second  complex number: 2 - 2i
Enter + for addition
              "Enter * for multiplication"
              "Enter - for subtraction"
              "Enter e for exit : *
Result is: 0
```

## WEEK-12

**A)AIM:** Write a C program which copies one file to another.

**PROGRAM:**

```
#include <stdio.h>
#include <conio.h>
#include <process.h>
void main(int argc, char *argv[])
{
 FILE *fs,*ft;
 char ch;
 clrscr();
 if(argc!=3)
 {
  puts("Invalid number of arguments.");
  exit(0);
  }
 fs = fopen(argv[1],"r");
 if(fs==NULL)
 {
 puts("Source file cannot be opened.");
 exit(0);
 }
 ft = fopen(argv[2],"w");
 if (ft==NULL)
 {
 puts("Target file cannot be opened.");
 fclose(fs);
 exit(0);
 }
 while(1)
 {
  ch=fgetc(fs);
  if (ch==EOF)
  break;
  else
  fputc(ch,ft);
  }
  fclose(fs);
  fclose(ft);
  getch();
}
```

**Result:**
File created is passed as parameter:
File is copied

**B) AIM:** Write a C program to reverse the first n characters in a file.
(Note: The file name and n are specified on the command line.)

**ALGORITHM:**

```
STEP1: Declare the variables and file pointer

STEP2: Check the number of arguments in command line
           if arg is not equal to 3
           then print "invalid arguments"
           exit

STEP3: Open an existed file in read mode

STEP4: if file not found
          then print "file can not be open"

STEP5: Assign 2nd argument in command line to a variable

STEP6: Read the contents from existed file and reverse first n characters in the
string from file.
```

**PROGRAM:**

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <process.h>
void main(int argc, char *argv[])
{
  char a[15];
  char s[20];
  char n;
  int k;
  int j=0;
  int i;
  int len;
  FILE *fp;
 if(argc!=3)
  {
  puts("Improper number of arguments.");
  exit(0);
  }
  fp = fopen(argv[1],"r");
  if(fp == NULL)
  {
  puts("File cannot be opened.");
  exit(0);
  }
k=atoi(argv[2]);
  n = fread(a,1,k,fp);
  a[n]='\0';
  len=strlen(a);
  for(i=len-1;i>=0;i--)
  {
   s[j]=a[i];
   printf("%c",s[j]);
   j=j+1;
}
s[j+1]='\0';
getch();
}
```

**Result:**

Abc.txt: He is a good boy

Output: yob doog a si eH

## WEEK 14

**Aim:** Write a C program to Implement the following searching method.
i) linear search ii) Binary search

**Program**
**i) Linear search:**

```
/*SEQUENTIAL SEARCH*/
#include<stdio.h>
main()
{
 int a[10],i,n,key,co=0;
 clrscr();
 printf("how many you want");
 scanf("%d",&n);
 printf("enter array elements:");
 for(i=0;i<n;i++)
 {
  scanf("%d",&a[i]);
 }
 printf("enter the searching elements");
 scanf("%d",&key);
search(a,n);
}

Void search(int a[10], int n)
{
    int i;
 for(i=0;i<n;i++)
 {
  if(a[i]==key)
   co++;
 }
 if(co>0)
  printf("Element is found");
 else
  printf("Not found");
 getch();
}
```

**Output:**

```
how many you want5
enter array elements:3 1 7 12 45
enter the searching elements12
Element is found
```

**ii) Binary search** :

```
/*BINARY SEARCH USING RECURSSION */
#include<stdio.h>
main()
{
 int a[10],i,j,t,n,key,low,high,co;
 clrscr();
 printf("how many you want");
 scanf("%d",&n);
 printf("enter array elements:");
 for(i=0;i<n;i++)
 {
  scanf("%d",&a[i]);
 }
 for(i=0;i<n;i++)
 {
  for(j=0;j<n-i-1;j++)
  {
   if(a[j]>a[j+1])
   {
    t=a[j];
    a[j]=a[j+1];
    a[j+1]=t;
   } }}


 low=0;
 high=n-1;
 printf("enter the searching elements");
 scanf("%d",&key);
 co=Rbinarysearch(a,low,high,key);
 if(co==-1)
  printf("Not found");
 else
  printf("Element is found");
 getch();
}

Rbinarysearch(int a[10],int low,int high,int key)
{
 int mid;
 if(low>high)
  return(-1);
  mid=(low+high)/2;
 if(key==a[mid])
  return(mid);
 if(key<a[mid])
   return(Rbinarysearch(a,low,mid-1,key));
 else
   return(Rbinarysearch(a,mid+1,high,key));
}
```

**Output:**

```
how many you want5
enter array elements:32 1 45 67 98
enter the searching elements98
Element is found
```

## WEEK 15

**AIM:** Write C programs that implement the following sorting methods to sort a given list of integers in ascending order by using Bubble sort.

**Algorithm for Bubble sort :**

```
1. start
2. take list(array), num
3. readlist(list,num)
4. printlist(list,num)
5. bub_sort(list,num)
6. printlist(list,num)
7. stop

readlist (list, num)
1. for j ← 0 to num
2. read  list[j].

printlist(list,num)
1. for j ← 0 to num
2. write list[j].

bub_sort(list,num)
 1. for i ← 0 to num
 2.     for j ← 0 to (num – i)
 3.            if( list[j] > list[j+1])
 4.                   swapList( address of list[j], address of list[j+1])

swapList( address of list[j], address of list[j+1])
 1. temp ← value at list[j]
 2. value at list[j]  ← value at list[j+1]
 3. value at list[j+1] ← temp
```

**PROGRAM:**

```
#include <stdio.h>
#define MAX 10

void swapList(int *m,int *n)
{
   int temp;
   temp = *m;
   *m = *n;
   *n = temp;
}

/* Function for Bubble Sort */
void bub_sort(int list[], int n)
{
   int i,j;
   for(i=0;i<(n-1);i++)
      for(j=0;j<(n-(i+1));j++)
            if(list[j] > list[j+1])
                  swapList(&list[j],&list[j+1]);
}

void readlist(int list[],int n)
{
   int j;
   printf("\nEnter the elements: \n");
   for(j=0;j<n;j++)
       scanf("%d",&list[j]);
}

/* Showing the contents of the list */
void printlist(int list[],int n)
{
   int j;
   for(j=0;j<n;j++)
      printf("%d\t",list[j]);
}

void main()
{
   int list[MAX], num;
   clrscr();
   printf("\n\n\n***** Enter the number of elements [Maximum 10] *****\n");
   scanf("%d",&num);
   readlist(list,num);
   printf("\n\nElements in the list before sorting are:\n");
   printlist(list,num);
   bub_sort(list,num);
   printf("\n\nElements in the list after sorting are:\n");
   printlist(list,num);
   getch();
}
```

**Output:**

```
Enter the number of elements [Maximum 10]: 5
Enter the elements:4 1 8 2 3
nElements in the list before sorting are: 4 1 8 2 3
Elements in the list after sorting are:1 2 3 4 8
```

## WEEK-16

**AIM:** Write a C program that uses functions to perform the following operations on singly linked list.

i) creation ii) Traversal

**Program:**

```
#include <stdio.h>
#include <stdlib.h>
#define  NULL 0

struct linked_list
{
        int number;
        struct linked_list *next;
};
typedef struct linked_list node;  /* node type defined */

main()
{
        node *head;
        int opt;
        void create(node *p);
        node *insert(node *head);
        node *delete(node *head);
        int count(node *p);
        void print(node *p);
        clrscr();
        head = (node *)malloc(sizeof(node));
        create(head);
        printf("\n");
        print(head);
        printf("\n");
        printf("\nNumber of items = %d \n", count(head));

        while(1)
        {
                printf("Enter 1. insertion\n"
                       " 2. deletion \n"
                       " 3. traversal \n"
                       " 4. exit:");
                scanf("%d",&opt);
                switch(opt)
                {
                        case 1: head=insert(head);
                                break;
                        case 2: head= delete(head);
                                break;
                        case 3: print(head);
                                break;
                        case 4: exit(0);

                }
        }
}
void create(node *list)
{
        printf("Input a number\n");
        printf("(type -999 at end): ");
        scanf("%d", &list -> number); /* create current node */

        if(list->number == -999)
        {
                list->next = NULL;
        }
        else
        {
                list->next = (node *)malloc(sizeof(node));
                create(list->next);
        }
        return;
}



void print(node *list)
{
                if(list->next != NULL)
                {
                  printf("%d-->",list ->number);  /* print current item */

                   if(list->next->next == NULL)
                           printf("%d", list->next->number);

                  print(list->next);              /* move to next item */
                }
                return;
        }

int count(node *list)
{
                if(list->next == NULL)
                      return (0);
                else
                      return(1+ count(list->next));
}
node *delete(node *head)
{
        node *find(node *p, int a);
        int  key;                       /* item to be deleted */
        node *n1;                       /* pointer to node preceding key node */
        node *p;                        /* temporary pointer */
        printf("\n What is the item (number) to be deleted?");
        scanf("%d", &key);
        if(head->number == key) /* first node to be deleted) */
        {
                p = head->next;         /* pointer to 2nd node  in list */
                free(head);             /* release space of key node */
                head = p;               /* make head to point to 1st node */
        }
        else
        {
                n1 = find(head, key);
                if(n1 == NULL)
                   printf("\n key not found \n");
                else                              /* delete key node */
                {
                        p = n1->next->next;          /*  pointer to the node
                                                     following the keynode */

                        free(n1->next);                      /* free key node */
                        n1->next = p;                        /* establish link */
                }
        }
        return(head);
}
node *insert(node *head)
{
        node *find(node *p, int a);
        node *new;
        node *n1;
        int key;
        int x;

        printf("Value of new item?");
        scanf("%d", &x);
        printf("Value of key item ? (type -999 if last) ");
        scanf("%d", &key);

        if(head->number == key)
        {
           new = (node *)malloc(sizeof(node));
           new->number = x;
           new->next = head;
           head = new;
        }
        else
        {
                n1 = find(head, key);

                if(n1 == NULL)
                   printf("\n key is not found \n");
                else
                {
             new = (node *)malloc(sizeof(node));
             new->number = x;
             new->next = n1->next;
             n1->next = new;
                }
        }
        return(head);
}
node *find(node *list, int key)
{
                if(list->next->number == key)       /* key found */
```

```
                        return(list);
        else

                if(list->next->next == NULL)          /* end */
                        return(NULL);
        else
                find(list->next, key);          }
```

**Result:**

```
1. Create
2. Insert
3. Delete
4. Display
5. Exit
Enter Choice : 1

Enter the List of no.s and stop with 100
25 36 45 69 100
Count = 4
1. Create
2. Insert
3. Delete
4. Display
5. Exit
Enter Choice : 2
Enter the no you want to insert : 3
Enter the position you want to insert : 3
Count = 5
The inserted element is : 3
1. Create
2. Insert
3. Delete
4. Display
5. Exit
Enter Choice : 3
Enter position do you want to delete : 3
Count = 3
Deleted element is : 3
1. Create
2. Insert
3. Delete
4. Display
5. Exit
Enter Choice : 4
Null ->25->36->45->69->NULL
1. Create
2. Insert
3. Delete
4. Display
5. Exit
Enter Choice : 5
```

## WEEK-17

**AIM**: Write a C program that implements stack (its operations) using Arrays

**Program**:

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

int st_arr[20];
int t=-1;

void push_ele(int ele);
int pop_ele();
void display_ele();

void main()
{
  char choice,num1=0,num2=0;
  while(1)
  {
    clrscr();
    printf("======================================");
    printf("\n\t\t MENU ");
    printf("\n======================================");
    printf("\n[1] Using Push Function");
    printf("\n[2] Using Pop Function");
    printf("\n[3] Elements present in Stack");
    printf("\n[4] Exit\n");
    printf("\n\tEnter your choice: ");
    fflush(stdin);
    scanf("%c",&choice);

    switch(choice-'0')
    {
      case 1:
      {
          printf("\n\tElement to be pushed: ");
          scanf("%d",&num1);
          push_ele(num1);
          break;
      }

      case 2:
      {
           num2=pop_ele(1);
           printf("\n\tElement to be popped: %d\n\t",num2);
           getch();
           break;
      }

      case 3:
      {
          display_ele();
          getch();
          break;
      }

      case 4:
          exit(1);
          break;

      default:
          printf("\nYour choice is invalid.\n");
          break;
    }
  }
}

/*Implementing the push() function. */
void push_ele(int ele)
{
  if(t==99)
  {
    printf("STACK is Full.\n");
    getch();
    exit(1);
  }
  st_arr[++t]=ele;
}

/*Implementing the pop() function. */
int pop_ele()
{
  int ele1;
  if(t==-1)
  {
    printf("\n\tSTACK is Empty.\n");
    getch();
    exit(1);
  }
  return(st_arr[t--]);
}

/*Implementing display() function. */
void display_ele()
{
  int k;
  printf("\n\tElements present in the stack are:\n\t");
  for(k=0;k<=t;k++)
  printf("%d\t",st_arr[k]);
}
```

**Result:**

```
Enter size of stack: 4
Enter option push,pop & display 1
Enter element to push: 3
Enter option push,pop & display 1
Enter element to push: 5
Enter option push,pop & display 2
Element 5 deleted
Enter option push,pop & display 3
Elements in stack are: 3
```

**WEEK-18**

**AIM:** Write a C program that implements Queue (its operations) using Arrays.

**Program:**

```
#include<stdio.h>
#include<alloc.h>
#include<conio.h>
#define size 10
#define true 1
#define false 0

struct q_arr
{
  int f,r;
  int num;
  int a[size];
};

void init(struct q_arr* queue);
int e_que(struct q_arr* queue);
int f_que(struct q_arr* queue);
int add_ele(struct q_arr* queue,int);
int rem_ele(struct q_arr* queue);
void display_ele(struct q_arr* queue);

/*main function*/
void main()
{
  int ele,k;
  int ch;

  struct q_arr *queue = (struct q_arr*)malloc(sizeof(struct q_arr));
  init(queue);

  while(1)
  {
    clrscr();
    printf("\n\n****IMPLEMENTATION OF QUEUE USING ARRAYS****\n");
    printf("============================================");
    printf("\n\t\tMENU\n");
    printf("============================================");
    printf("\n\t[1] To insert an element");
    printf("\n\t[2] To remove an element");
    printf("\n\t[3] To display all the elements");
    printf("\n\t[4] Exit");
    printf("\n\n\t Enter your choice: ");
    scanf("%d",&ch);

    switch(ch)
    {
      case 1:
      {
          printf("\nElement to be inserted:");
```

```
        scanf("%d",&ele);
        add_ele(queue,ele);
        break;
    }

    case 2:
    {
        if(!e_que(queue))
        {
          k=rem_ele(queue);
          printf("\n%d element is removed\n",k);
          getch();
        }
        else
        {
          printf("\tQueue is Empty. No element can be removed.");
          getch();
        }
        break;
    }

    case 3:
    {
        display_ele(queue);
        getch();
        break;
    }

    case 4:
        exit(0);

    default:
        printf("\tInvalid Choice.");
        getch();
        break;
   }
  }
}
/*end main*/

void init(struct q_arr* queue)
{
  queue->f = 0;
  queue->r = -1;
  queue->num = 0;
}

/* Function to check is the queue is empty*/
int e_que(struct q_arr* queue)
{
  if(queue->num==0)
  return true;
  return false;
}

/* Function to check if the queue is full*/
```

```
int f_que(struct q_arr* queue)
{
  if(queue->num == size)
  return true;
  return false;
}

/* Function to add an element to the queue*/
int add_ele(struct q_arr* queue,int j)
{
  if(f_que(queue))
  return false;

  if(queue->r == size - 1)
  queue->r = -1;
  queue->a[++queue->r] = j;
  queue->num++;
  return true;
}

/* Function to remove an element of the queue*/
int rem_ele(struct q_arr* queue)
{
  int j;
  if(e_que(queue))
  return -9999;
  j = queue->a[queue->f++];
  if(queue->f == size)
  queue->f = 0;
  queue->num--;
  return j;
}

/* Function to display the queue*/
void display_ele(struct q_arr* queue)
{
  int j;
  if(e_que(queue))
  {
    printf("Queue is Empty. No records to display.");
    return;
  }
  printf("\nElements present in the Queue are: ");
  for(j=queue->f;j<=queue->r;j++)
    printf("%d\t",queue->a[j]);
    printf("\n");
}
```

**Result:**

```
Enter queue size: 3
Enter add,delete & display 1
Enter element to add: 3
Enter add,delete & display 1
Enter element to add: 5
Enter add,delete & display 3
Elements in queue are: 3 5
```

**AIM:** Write a C program that implements Queue (its operations) using Pointers.

**Program:**

```
#define true 1
#define false 0

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

struct q_point
{
  int ele;
  struct q_point* n;
};

struct q_point *f_ptr = NULL;

int e_que(void);
void add_ele(int);
int rem_ele(void);
void show_ele();

/*main function*/
void main()
{
  int ele,choice,j;
  while(1)
  {
    clrscr();
    printf("\n\n****IMPLEMENTATION OF QUEUE USING POINTERS****\n");
    printf("==============================================");
    printf("\n\t\t  MENU\n");
    printf("==============================================");
    printf("\n\t[1] To insert an element");
    printf("\n\t[2] To remove an element");
    printf("\n\t[3] To display all the elements");
    printf("\n\t[4] Exit");
    printf("\n\n\tEnter your choice:");
    scanf("%d", &choice);

    switch(choice)
    {
      case 1:
      {
          printf("\n\tElement to be inserted:");
          scanf("%d",&ele);
          add_ele(ele);
          getch();
          break;
      }

      case 2:
      {
          if(!e_que())
          {
            j=rem_ele();
            printf("\n\t%d is removed from the queue",j);
            getch();
          }
          else
          {
            printf("\n\tQueue is Empty.");
            getch();
          }
          break;
      }

      case 3:
          show_ele();
          getch();
          break;

      case 4:
          exit(1);
          break;

      default:
          printf("\n\tInvalid choice.");
          getch();
          break;
    }

  }
}

/* Function to check if the queue is empty*/
int e_que(void)
{
  if(f_ptr==NULL)
  return true;
  return false;
}

/* Function to add an element to the queue*/
void add_ele(int ele)
{
  struct q_point *queue = (struct q_point*)malloc(sizeof(struct q_point));
  queue->ele = ele;
  queue->n = NULL;
  if(f_ptr==NULL)
    f_ptr = queue;
  else
  {
    struct q_point* ptr;
    ptr = f_ptr;
    for(ptr=f_ptr ;ptr->n!=NULL; ptr=ptr->n);
      ptr->n = queue;
  }
}
```

```
/* Function to remove an element from the queue*/
int rem_ele()
{
  struct q_point* queue=NULL;
  if(e_que()==false)
  {
    int j = f_ptr->ele;
    queue=f_ptr;
    f_ptr = f_ptr->n;
    free (queue);
    return j;
  }
  else
  {
    printf("\n\tQueue is empty.");
    return -9999;
  }
}

/* Function to display the queue*/
void show_ele()
{
  struct q_point *ptr=NULL;
  ptr=f_ptr;
  if(e_que())
  {
    printf("\n\tQUEUE is Empty.");
    return;
  }
  else
  {
    printf("\n\tElements present in Queue are:\n\t");
    while(ptr!=NULL)
    {
      printf("%d\t",ptr->ele);
      ptr=ptr->n;
    }
  }
}
```

**Result:**

```
Enter queue size: 3
Enter add,delete & display 1
Enter element to add: 3
Enter add,delete & display 1
Enter element to add: 5
Enter add,delete & display 3
Elements in queue are: 3 5
```

## **WEEK 19**

**Aim:** Write C programs to implement the linear regression algorithms.

**Program:**

```
#include<math.h>
#include<stdio.h>
#include<conio.h>
main()
{
 int n,i;
 float x,y,m,c,d;
 float sumx=0,sumxsq=0,sumy=0,sumxy=0;
 clrscr();
 printf("enter the number of values for n:");
 scanf("%d",&n);
 for(i=0;i<n;i++)
 {
  printf("enter values of x and y");
  scanf("%f%f",&x,&y);
  sumx=sumx+x;
  sumxsq=sumxsq+(x*x);
  sumy=sumy+y;
  sumxy=sumxy+(x*y);
 }
 d=n*sumxsq-sumx*sumx;
 m=(n*sumxy-sumx*sumy)/d;
 c=(sumy*sumxsq-sumx*sumxy)/d;
 printf("M=%f\tC=%f\n",m,c);
 getch();
}
```

## WEEK 20

**Aim:** Write C programs to implement the polynomial regression algorithms.

**Program:**

```
#include<math.h>
#include<stdio.h>
#include<conio.h>
main()
{
 int i,j,k,m,n;
 float x[20],y[20],u,a[10],c[20][20],power,r;
 clrscr();
 printf("enter m,n:");
 scanf("%d%d",&m,&n);
 for(i=1;i<=n;i++)
 {
  printf("enter values of x and y");
  scanf("%f%f",&x[i],&y[i]);
 }
 for(j=1;j<=m+1;j++)
  for(k=1;k<=m+1;k++)
  {
   c[j][k]=0;
   for(i=1;i<=n;i++)
   {
    power=pow(x[i],j+k-2);
    c[j][k]=c[j][k]+power;
   }
  }
  for(j=1;j<=m+1;j++)
  {
   c[j][m+2]=0;
   for(i=1;i<=n;i++)
   {
    r=pow(x[i],j-1);
    c[j][m+2]=c[j][m+2]+y[i]*r;
   }
  }

  for(i=1;i<=m+1;i++)
  {
   for(j=1;j<=m+2;j++)
   {
    printf("%.2f\t",c[i][j]);
   }
   printf("\n");
  }
  for(k=1;k<=m+1;k++)
  for(i=1;i<=m+1;i++)
  {
   if(i!=k)
   {
    u=c[i][k]/c[k][k];
    for(j=k;j<=m+2;j++)
    {
     c[i][j]=c[i][j]-u*c[k][j];
    }
   }
  }
  for(i=1;i<=m+1;i++)
  {
   a[i]=c[i][m+2]/c[i][i];
      printf("a[%d]=%f\n",i,a[i]);
  }
 getch();
}
```

## WEEK 21

**Aim:** Write C programs to implement the Lagrange interpolation

**Program:**

```
main()
{
 int i,j,k,n;
 float term,sum,x1,x[20],f[20];
 textcolor(LIGHTCYAN);
 clrscr();
 printf("enter n value");
 scanf("%d",&n);
 for(i=0;i<n;i++)
 {
  printf("enter the values for x,f[x]");
  scanf("%f%f",&x[i],&f[i]);
 }
 printf("enter the value");
 scanf("%f",&x1);
 sum=0;
 for(k=1;k<=n;k++)
 {
  term=1;
  for(j=1;j<=n;j++)
  {
   if(j!=k)
   term=term*((x1-x[j])/(x[k]-x[j]));
  }
  sum=sum+term*f[k];
 }
 printf("%f=%f",x1,sum);
 getch();
}
```

## WEEK 22

**Aim:** Write C programs to implement Newton Gregory Forward Interpolation

**Program:**

```
#include<math.h>
#include<stdlib.h>
main()
{
 int i,j,k,n;
 float h,u,f,x[100],fx[100],term,xx;
 clrscr();
 printf("enter the no.of values");
 scanf("%d",&n);
 for(i=0;i<n;i++)
 {
  printf("enter the values x,fx");
  scanf("%f%f",&x[i],&fx[i]);
 }
 printf("enter the values xx");
 scanf("%f",&xx);
 h=x[2]-x[1];
 u=(xx-x[0])/h;
 for(i=1;i<=n;i++)
 {
  for(j=n-1;j>=i;j--)
  {
   fx[j]=fx[j]-fx[j-1];
  }
 }
 k=1;f=0;term=1;
 for(i=0;i<n;i++)
 {
  f=f+term*fx[i];
  term=term*(u-k+1)/k;
  k++;
 }
 printf("the value is:%f",f);
 getch();
}
```

**WEEK 23**

**Aim:** Write C programs to implement Trapezoidal method.

**Program:**

```
#include<math.h>
#include<stdio.h>
main()
{
 int i,n;
 float sum,s1,s2,h,x0,xn,fn,f0;
 clrscr();
 printf("enter the values x0,xn,n:");
 scanf("%f%f%d",&x0,&xn,&n);
 s1=s2=0;
 h=(xn-x0)/n;
 f0=x0*x0;
 fn=xn*xn;
 s1=f0+fn;
 for(i=1;i<=n-1;i++)
 {
  x0=x0+h;
  f0=x0*x0;
  s2=s2+f0;
  printf("x[%d]=%f\tf[%d]=%f\n",i,x0,i,f0);
 }
 sum=(h*(s1+2*s2))/2;
 printf("\tThe intergal value is:%f",sum);
 getch();
}
```

**WEEK 24**

**Aim:** Write C programs to implement Simpsons method.

**Program:**

```
/*                SIMPSONS 1/3rd RULE          */
#include<math.h>
#include<stdio.h>
main()
{
 int i,n;
 float sum,s0,s1,s2,h,x0,xn,fn,f0;
 clrscr();
 printf("enter the values x0,xn,n:");
 scanf("%f%f%d",&x0,&xn,&n);
 s0=s1=s2=0;
 h=(xn-x0)/n;
 f0=x0*x0;
 fn=xn*xn;
 s0=f0+fn;
 for(i=1;i<=n-1;i++)
 {
  x0=x0+h;
  f0=x0*x0;
  if(i%2!=0)
    s1=s1+f0;
  else
    s2=s2+f0;
 printf("x[%d]=%f\t f[%d]=%f\n",i,x0,i,f0);
 }
 sum=(h*(s0+4*s1+2*s2))/3;
 printf("\tThe intergal value is:%f",sum);
 getch();
}
```

```
/*      SIMPSONS 3/8th RULE           */
#include<math.h>
#include<stdio.h>
main()
{
 int i,n;
 float sum,s0,s1,s2,h,x0,xn,fn,f0;
 clrscr();
 printf("enter the values x0,xn,n:");
 scanf("%f%f%d",&x0,&xn,&n);
 s0=s1=s2=0;
 h=(xn-x0)/n;
 f0=x0*x0;
 fn=xn*xn;
 s0=f0+fn;
 for(i=1;i<=n-1;i++)
 {
  x0=x0+h;
  f0=x0*x0;
  if(i%3==0)
```

```
    s1=s1+2*f0;
  else
    s2=s2+3*f0;
 printf("x[%d]=%f\t f[%d]=%f\n",i,x0,i,f0);
 }
 sum=(3*h*(s0+s1+s2))/8;
 printf("\tThe intergal value is:%f",sum);
 getch();
}
```