C PROGRAMMING LAB

EXPERIMENT LIST

1. Write a C program to find the sum of individual digits of a positive integer.

2. 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.

3. Write a C program to generate all the prime numbers between 1 and n, where n is a value supplied by the user.

4. Write C programs that use both recursive and non-recursive functions
(a) To find the factorial of a given integer.

5. Write a C program to find both the largest and smallest number in a list of integers.

6. Write a C program that uses functions to perform the following:
(a) Addition of Two Matrices
(b) Multiplication of Two Matrices

7. Write a C program to determine if the given string is a palindrome or not

8. Write C programs that uses non recursive function to search for a key value in a given list of integers using Linear search

9. Write C programs that uses non recursive function to search for a key value in a given list of integers using Binary search

10. Write C programs that implements the Insertion sort method to sort a given array of integers in ascending order.

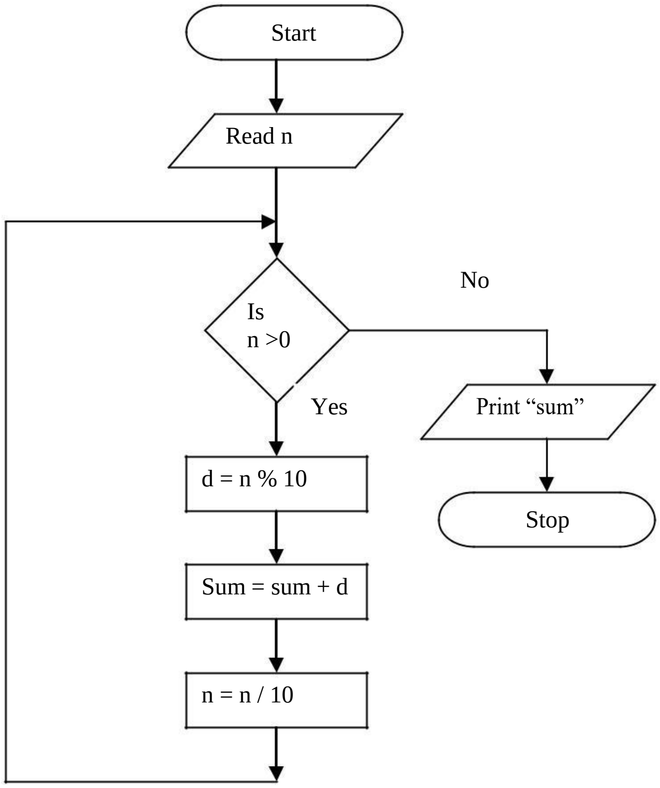
11. Write C programs that implements the Bubble sort method to sort a given array of integers in ascending order.

1. 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:

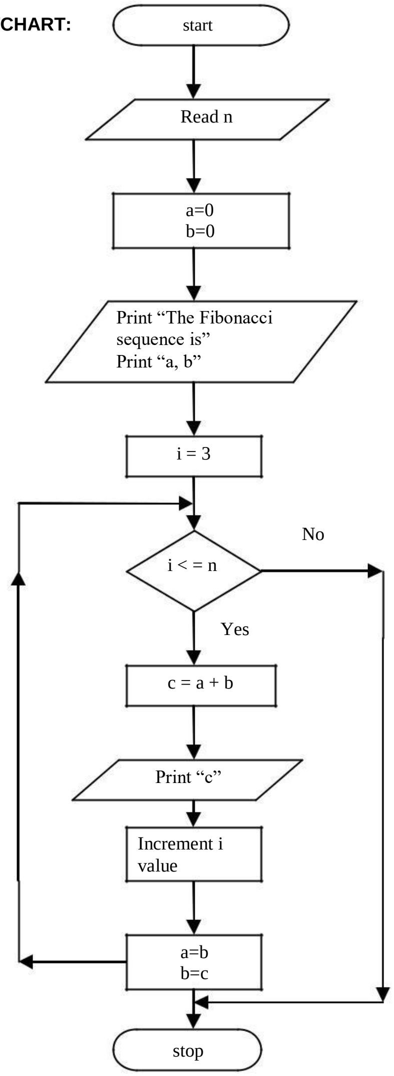
```
Enter any integer: 1234
Sum of individual digits is: 10
```

2. 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:

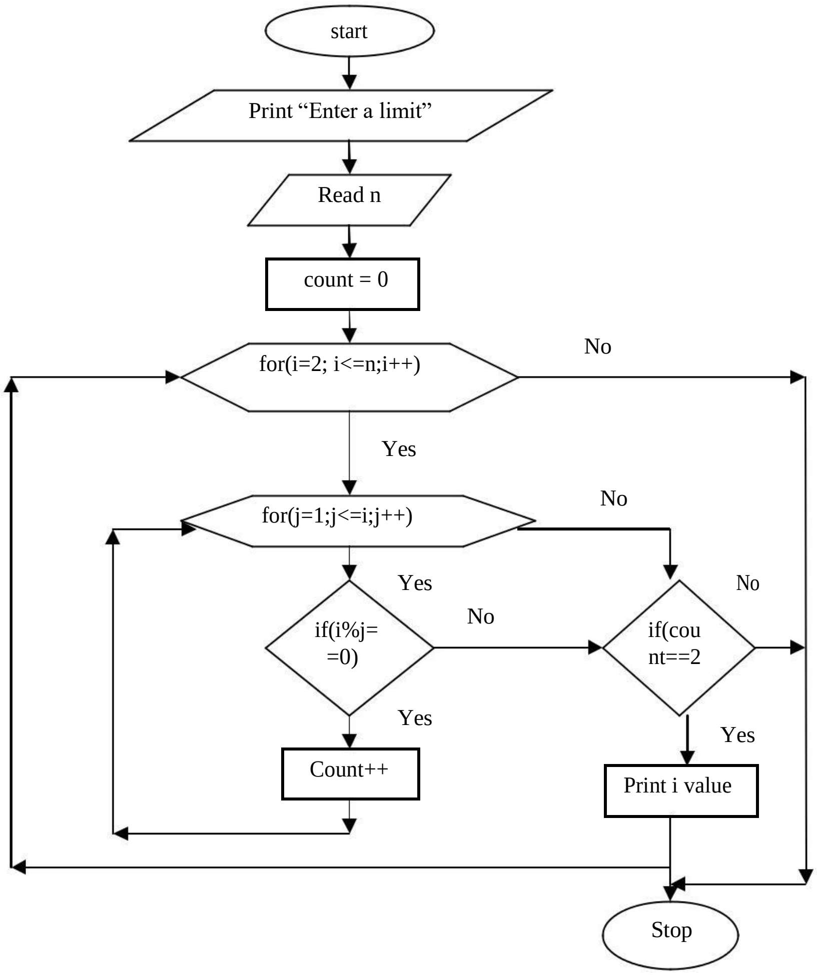
```
Enter no of items: 5
The Fibonacci sequence is
0 1 1 2 3
```

3. 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
```

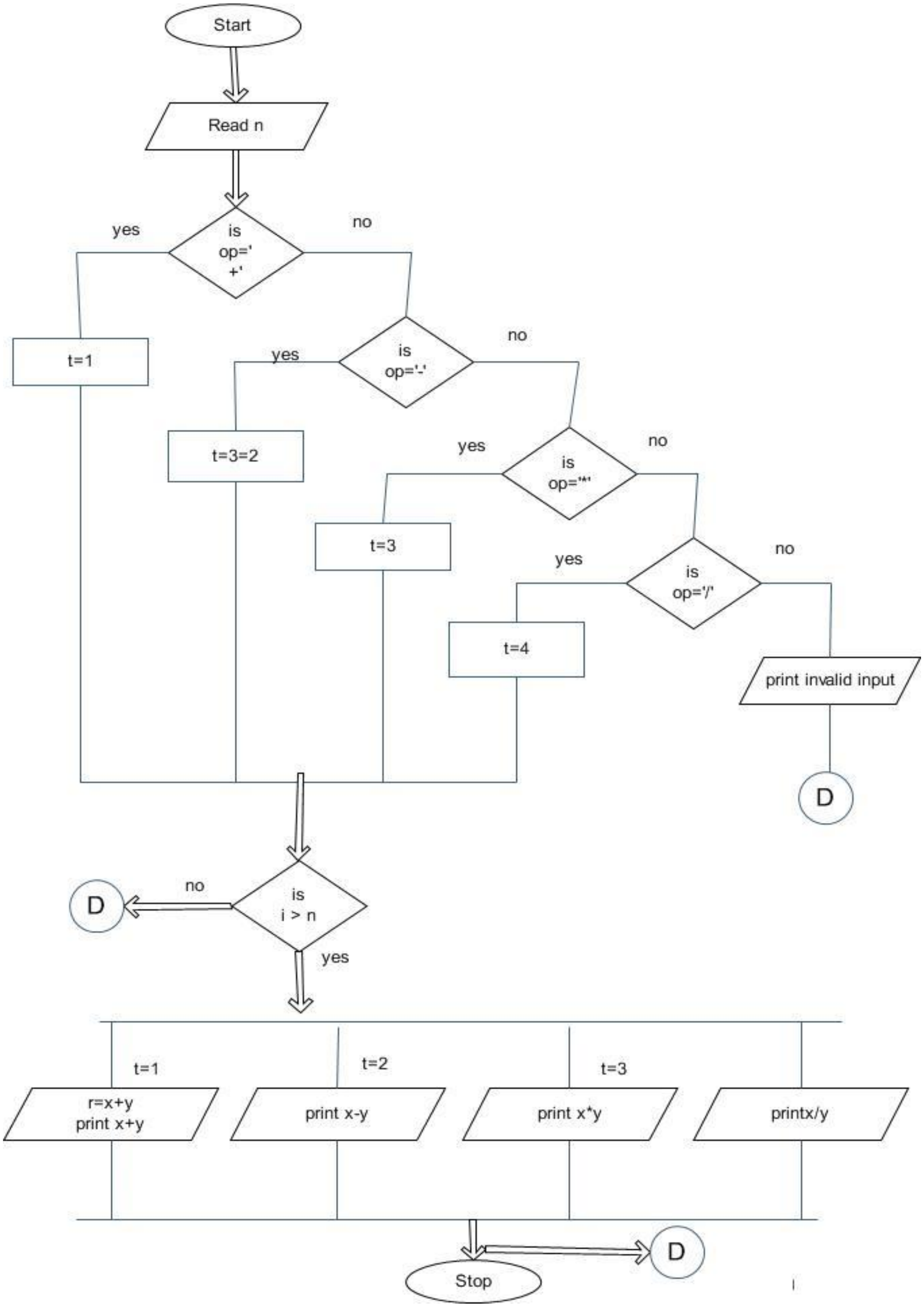
B) AIM: Two integer operands and one operator form user, performs the operation and then prints the result.

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

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:

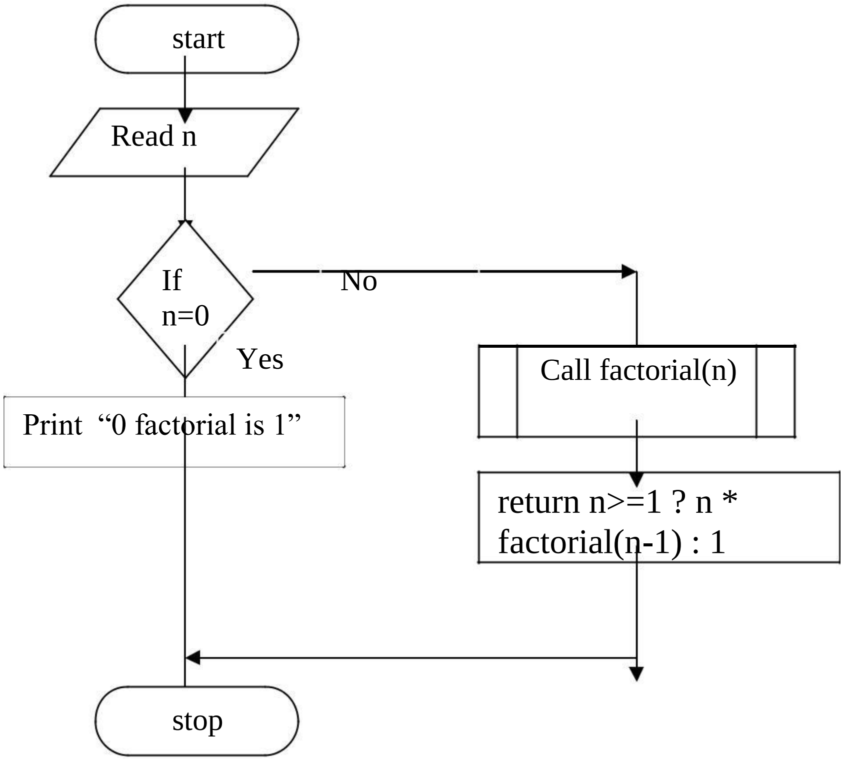
```
1.enter two operands:2 3
   enter an operator:+
   sum of two numbers 2 3 is: 5

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
```

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
```

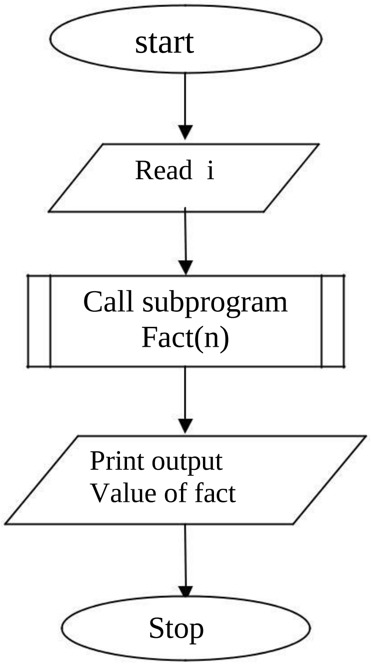
ii)Non-Recursive Algorithm: main program

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

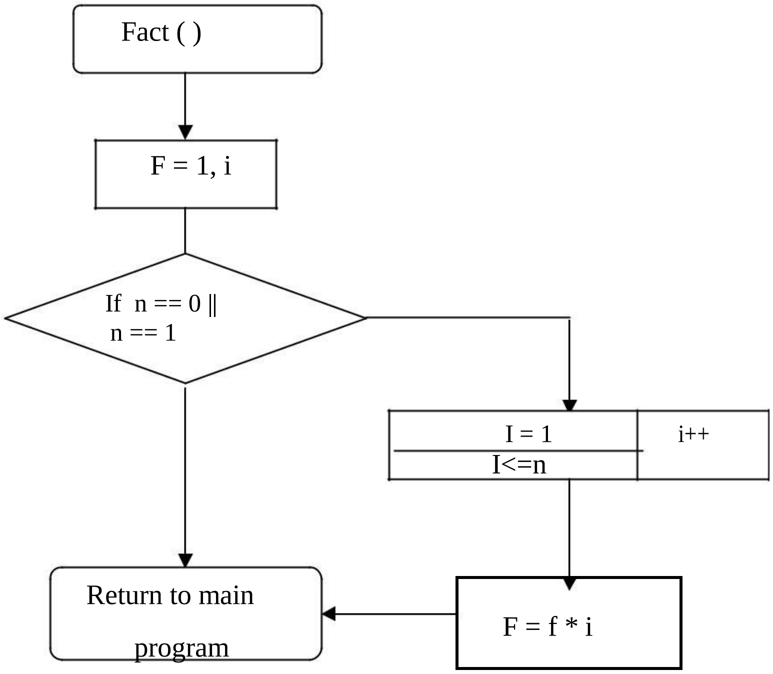
Sub program: fact

```
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:

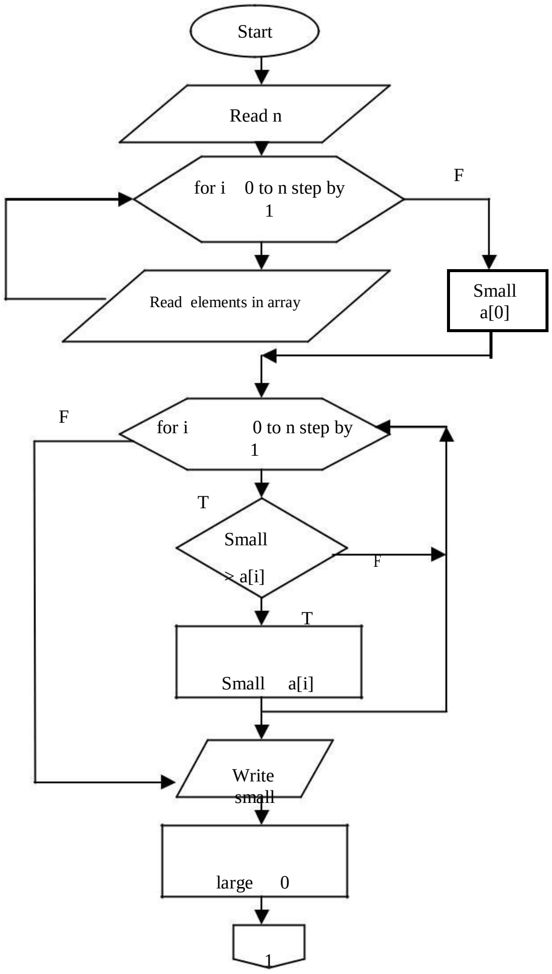
```
1.Enter a number: 7
  Factorial of number: 5040
```

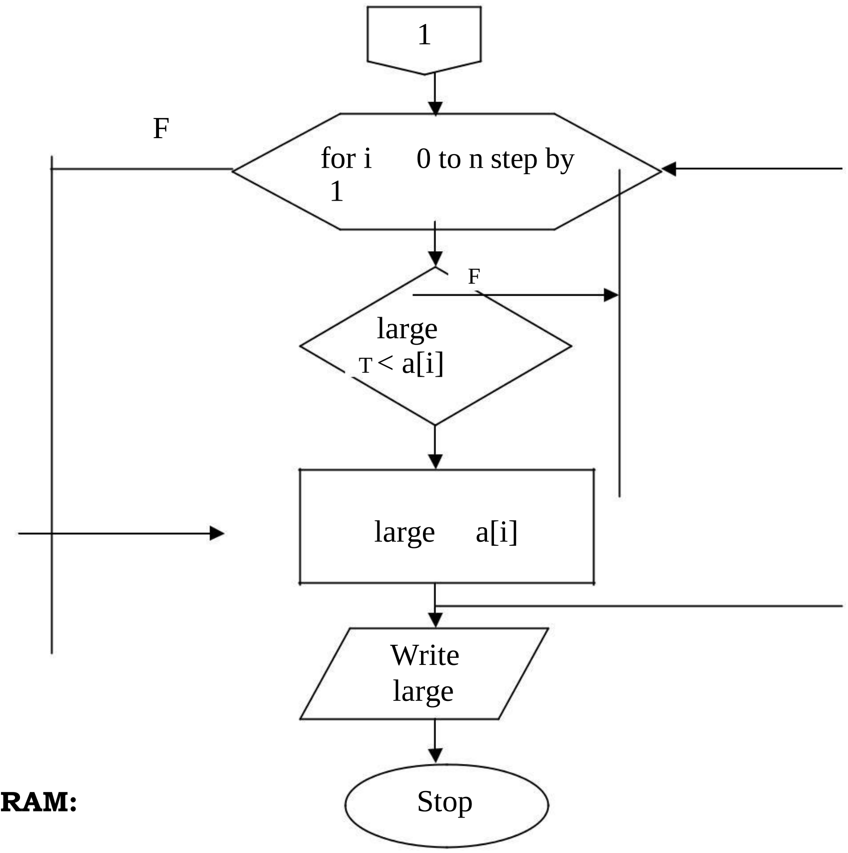
A) AIM: - A C program to find both the largest and smallest number in list of integers

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

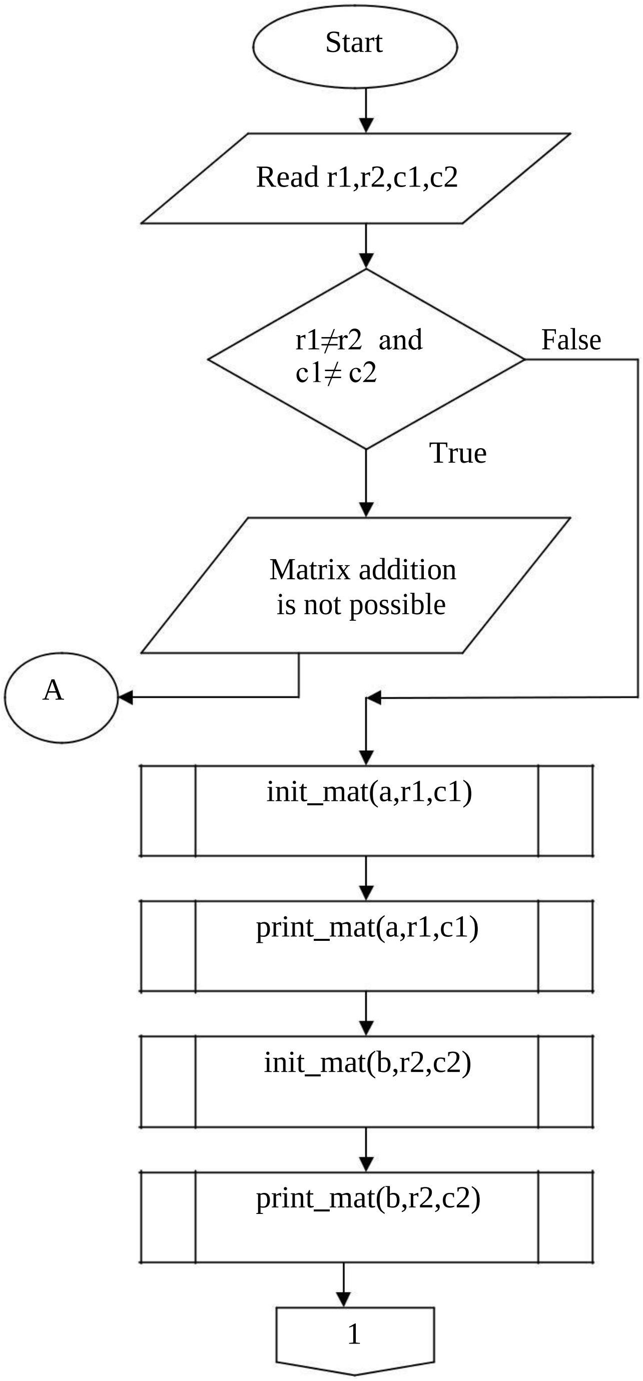
```
1. Start
2. read r1,r2,c1,c2
3. if $r1 \neq r2$ and $c1 \neq c2$
4.     then "matrix addition is not possible"
5. else
6.     do  init_mat(a,r1,c1)
7.         print_mat(a,r1,c1)
8.         init_mat(b,r2,c2)
9.         print_mat(b,r2,2)
10.        add_mat(a,b,c,r1,c1)
11.        print_mat(c,r1,c1)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

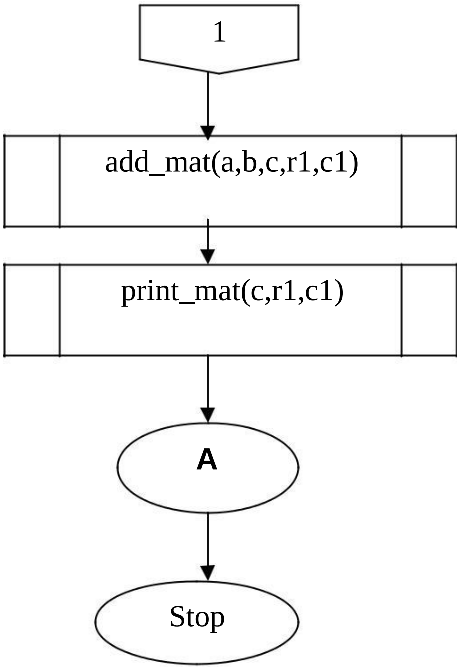
```
init_mat(a4,r4,c4)
1. for i   0 to r4
2.     do for j   0 to c4
3.             read a4[i][j]
```

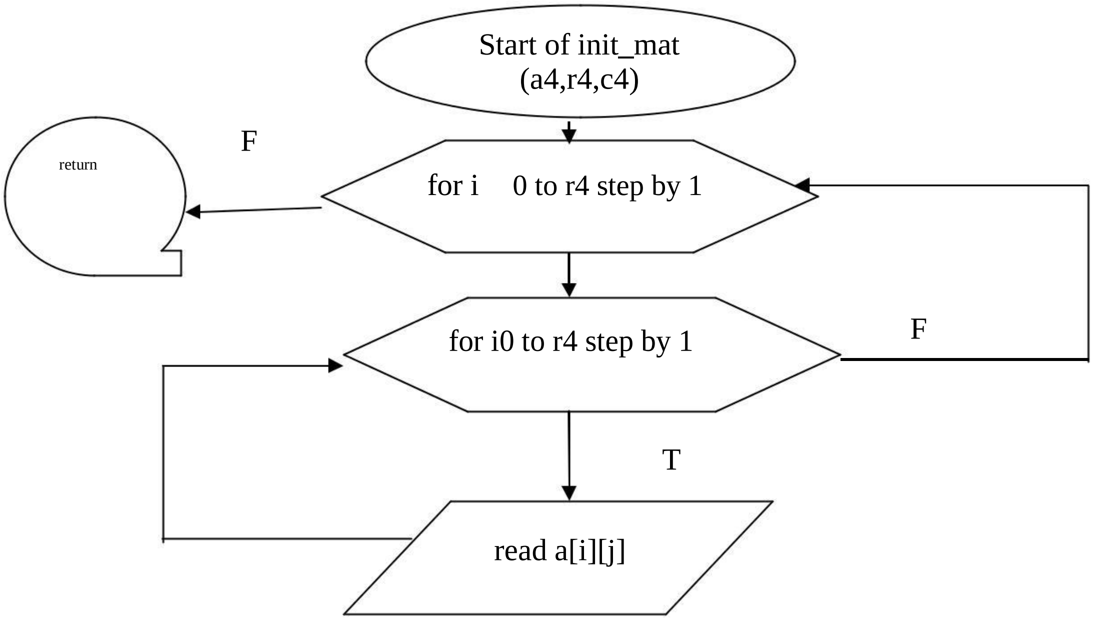
```
print _mat(a4,r4,c4)
1. for i   0 to r4
2.     do for j   0 to c4
3.             print a[i][j]
4.     print next_line
```

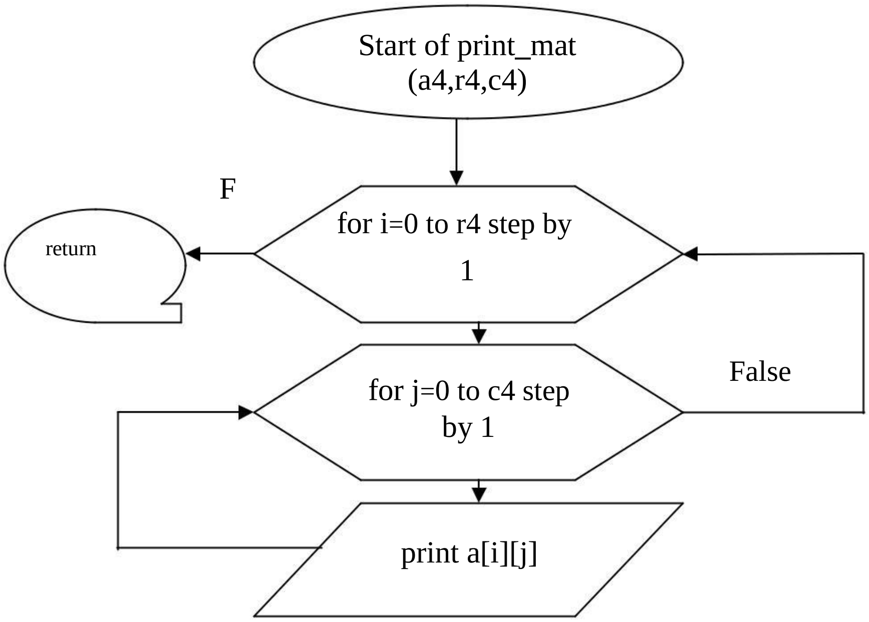
```
add_ mat(a4,b4,c24.r4,c4)
1. for i   0 to r4
2.     do for j   to c4
3.             c[i][j]   a[i][j] + b[i][j]
```

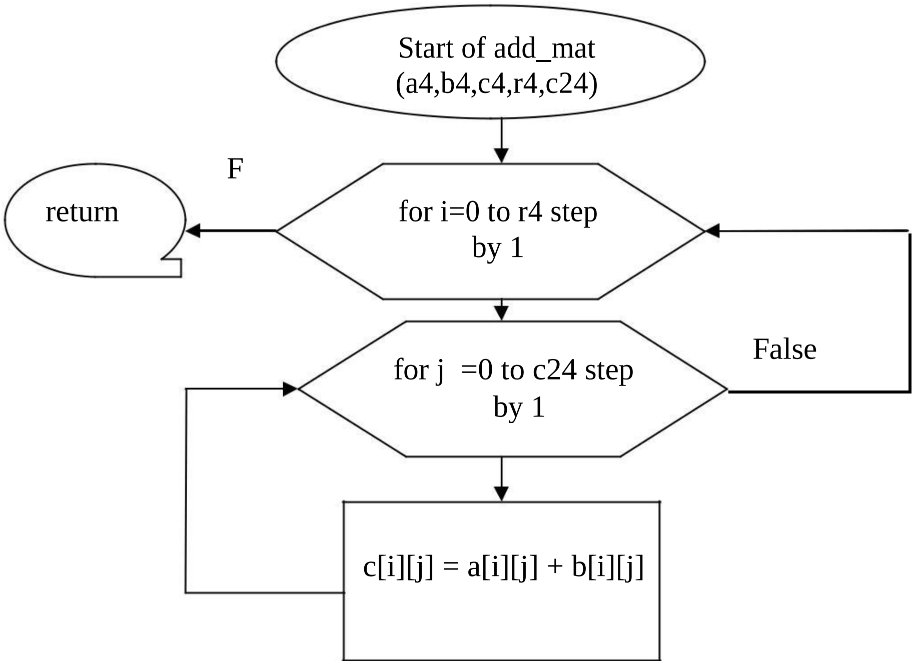
Flowchart:-


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)
F
return
for i 0 to r4 step by 1
F
for i0 to r4 step by 1
T
read a[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

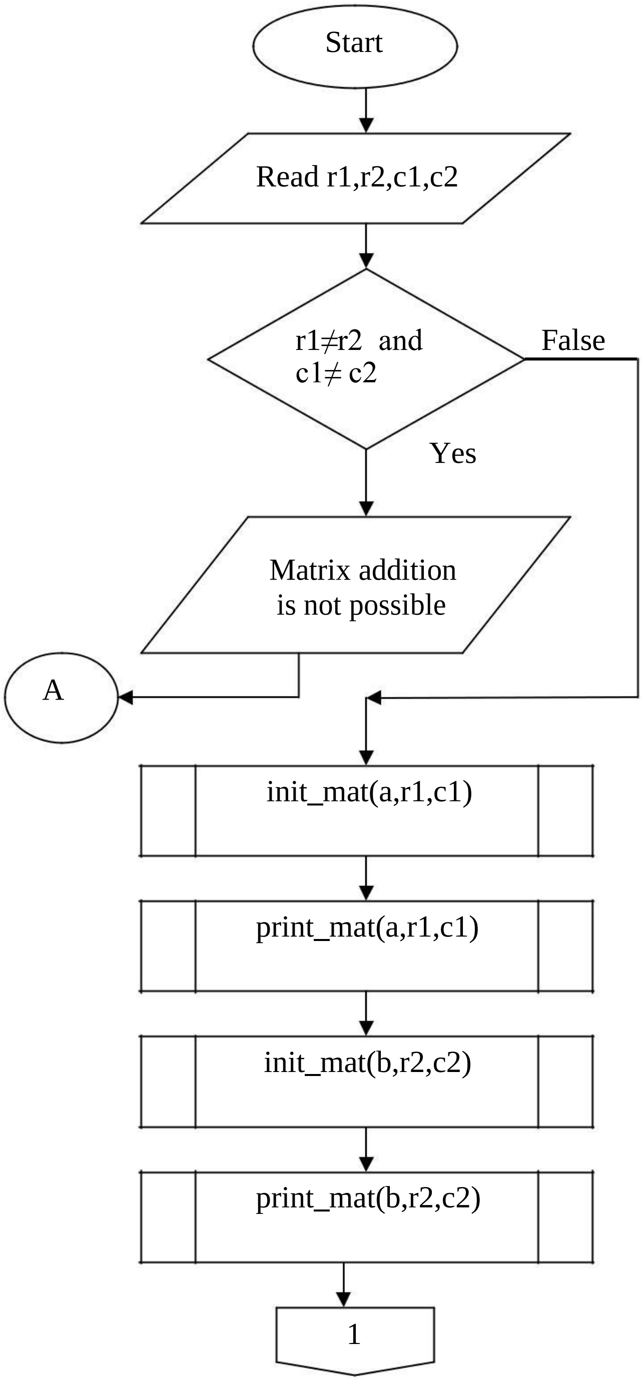
```
1.  Start
2.  read r1,r2,c1,c2
3.  if r1 ≠ c2
4.       then “matrix multiplication is not possible”
5.  else
6.       do  init_mat(a,r1,c1)
7.              print_mat(a,r1,c1)
8.              init_mat(b,r2,c2)
9.              print_mat(b,r2,2)
10.             mul_mat(a,b,c,r1,c1,c2)
11.          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

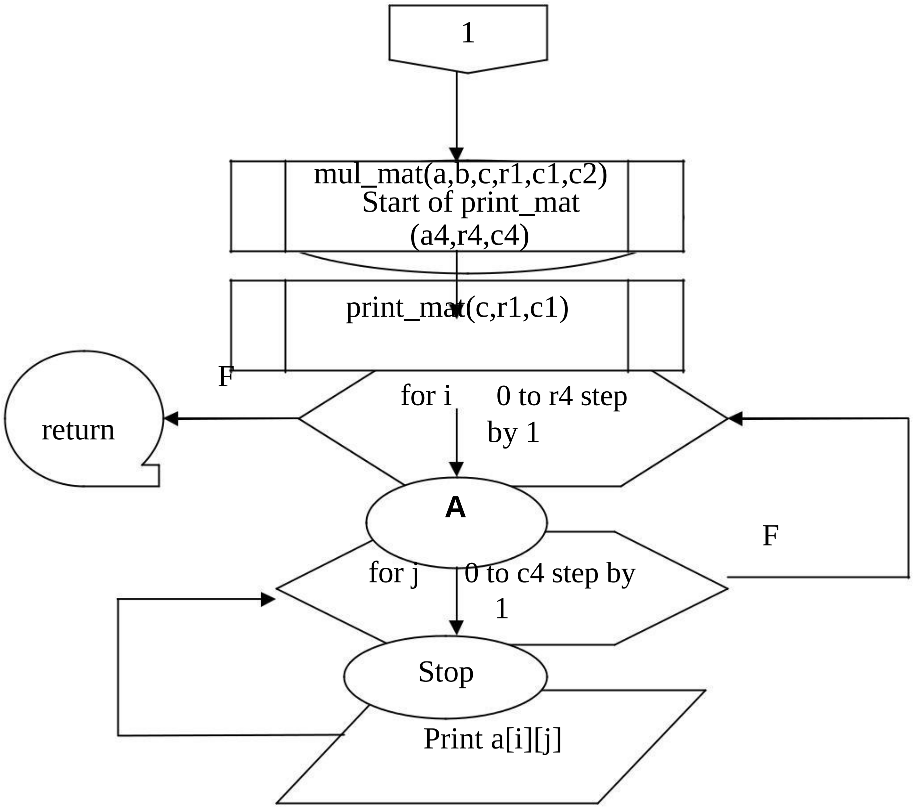
```
init_mat(a4,r4,c4)
        1.  for i   0 to r4
        2.  do for j   0 to c4
        3.                read a4[i][j]
```

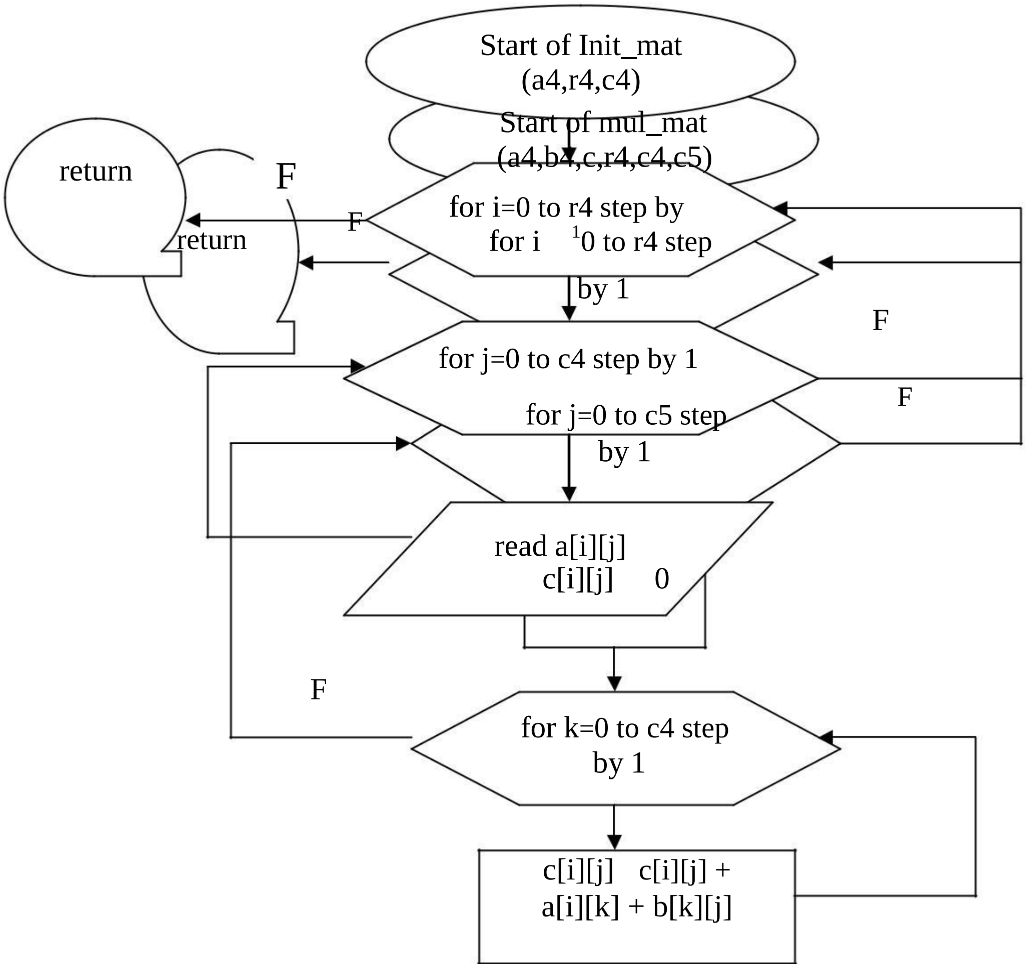
```
print_mat(a4,r4,c4)
        1.  for i   0 to r4
        2.  do for j   0 to c4
        3.            print a[i][j]
        4.      print next_line
```

```
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)
Start of print_mat
(a4,r4,c4)
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
for i 1 0 to r4 step
by 1
F
for j=0 to c4 step by 1
F
for j=0 to c5 step
by 1
read a[i][j]
c[i][j] 0
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 :

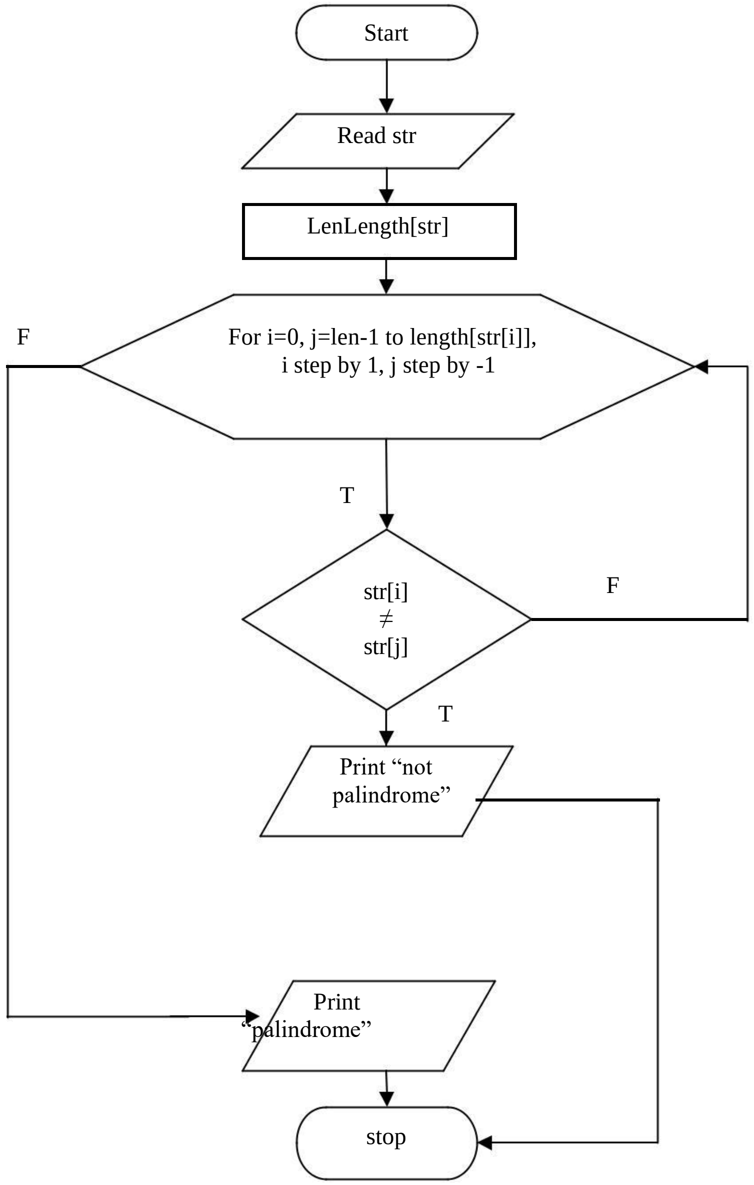
```
Matrix Multiplication is not possible
```

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:
```

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
```

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

Pseucode for Bubble sort :

```
begin BubbleSort(list)
        for all elements of list
            if list[i] > list[i+1]
              swap(list[i], list[i+1])
            end if
        end for
        return list
end BubbleSort
```

PROGRAM:

```
#include<stdio.h>
int main()
{
   int data[100],i,n,step,temp;
   printf("Enter the number of elements to be sorted: ");
   scanf("%d",&n);
   for(i=0;i<n;++i)
   {
      printf("%d. Enter element: ",i+1);
      scanf("%d",&data[i]);
   }

   for(step=0;step<n-1;++step)
   for(i=0;i<n-step-1;++i)
   {
      if(data[i]>data[i+1])
      {
         temp=data[i];
         data[i]=data[i+1];
         data[i+1]=temp;
      }
   }
   printf("In ascending order: ");
   for(i=0;i<n;++i)
       printf("%d  ",data[i]);
   return 0;
}
```

Output:

```
Enter the number of elements to be sorted: 6
1. Enter element: 12
2. Enter element: 3
3. Enter element: 0
4. Enter element: -3
5. Enter element: 1
6. Enter element: -9
In ascending order: -9 -3 0 1 3 13
```

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

Pseudocode for Insertion sort :

```
INSERTION_SORT ($A$)

FOR $j \leftarrow 2$ TO length[$A$]
    DO key $\leftarrow A[j]$
       {Put $A[j]$ into the sorted sequence $A[1 \,.\,.\, j - 1]$}
       $i \leftarrow j - 1$
       WHILE $i > 0$ and $A[i] >$ key
            DO $A[i+1] \leftarrow A[i]$
               $i \leftarrow i - 1$
       $A[i + 1] \leftarrow$ key
```

PROGRAM:

```
#include<stdio.h>
int main()
{
        int data[100],n,temp,i,j;
        printf("Enter number of terms: ");
        scanf("%d",&n);
        printf("Enter elements: ");
        for(i=0;i<n;i++)
        {
                scanf("%d",&data[i]);
        }
        for(i=1;i<n;i++)
        {
                temp = data[i];
                j=i-1;
                while(temp<data[j] && j>=0)
                {
                        data[j+1] = data[j];
                        --j;
                }
                data[j+1]=temp;
        }
        printf("In ascending order: ");
        for(i=0; i<n; i++)
                printf("%d\t",data[i]);
  return 0;
}
```

Output:

```
Enter number of terms: 5
Enter elements: 12
1
2
5
3
In ascending order: 1     2     3     5     12
```

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

Pseucode for Insertion sort :

```
INSERTION_SORT (A)

    FOR j ← 2 TO length[A]
        DO key ← A[j]
            {Put A[j] into the sorted sequence A[1 . . j − 1]}
            i ← j − 1
            WHILE i > 0 and A[i] > key
                DO A[i +1] ← A[i]
                    i ← i − 1
            A[i + 1] ← key
```

PROGRAM:

```
#include<stdio.h>
int main()
{
        int data[100],n,temp,i,j;
        printf("Enter number of terms: ");
        scanf("%d",&n);
        printf("Enter elements: ");
        for(i=0;i<n;i++)
        {
                scanf("%d",&data[i]);
        }
        for(i=1;i<n;i++)
        {
                temp = data[i];
                j=i-1;
                while(temp<data[j] && j>=0)
                {
                        data[j+1] = data[j];
                        --j;
                }
                data[j+1]=temp;
        }
        printf("In ascending order: ");
        for(i=0; i<n; i++)
                printf("%d\t",data[i]);
  return 0;
}
```

Output:

```
Enter number of terms: 5
Enter elements: 12
1
2
5
3
In ascending order: 1     2     3     5     12
```