# PART - A

## Program: 01

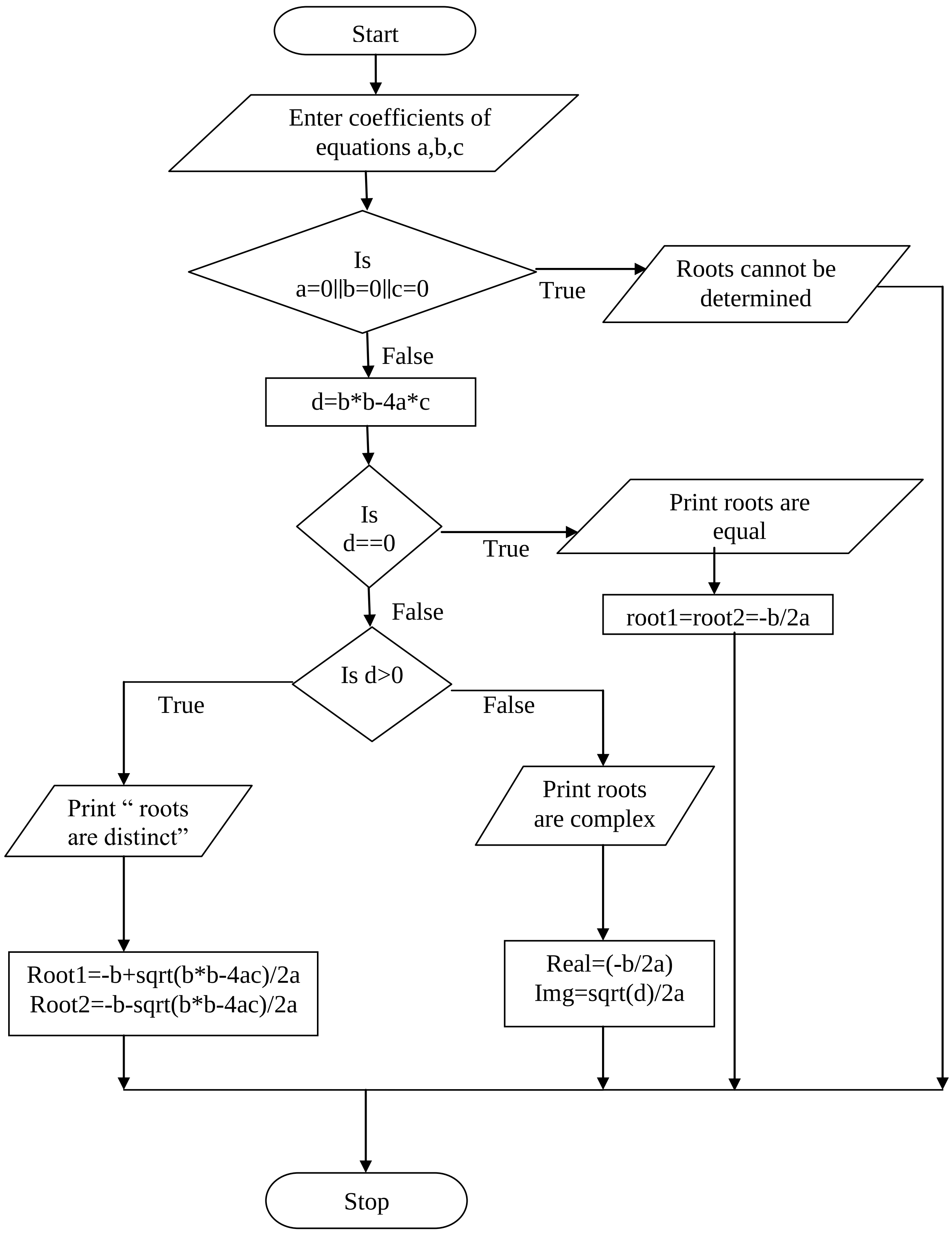
**Design, develop and execute a program in C to find and output all the roots of a given quadratic equation, for non-zero coefficients.**

**Algorithm:**

```
Step 1:[Input the coefficients of quadratic equations]
           read a,b,c
Step 2:[Find two equal roots]
           if(b*b-4ac=0)
               root1=root2=-b/2a
               print "two roots are equal"
               stop
        endif
Step 3:[Find two distinct roots]
           if(b*b-4ac>0) then
               root1=-b+sqrt(b*b-4ac)/2a
               root2=-b-sqrt(b*b-4ac)/2a
               print two roots
               stop
        endif
Step 4:[Find two complex roots]
           if(b*b-4ac>0) then
               root1=-b+isqrt(b*b-4ac)/2a
               root2=-b+isqrt(b*b-4ac)/2a
               print two roots
               stop
        endif

Step 5:[Find any zero coefficients ]
            if(a==0||b==0||c==0)
                print "error:It is impossible to find the roots"
                stop
            endif
Step 6:[Finished]
           Stop
```

**Flowchart:**



Start
Enter coefficients of equations a,b,c
Is a=0||b=0||c=0
True
Roots cannot be determined
False
d=b*b-4a*c
Is d==0
True
Print roots are equal
root1=root2=-b/2a
False
Is d>0
True
False
Print " roots are distinct"
Print roots are complex
Root1=-b+sqrt(b*b-4ac)/2a
Root2=-b-sqrt(b*b-4ac)/2a
Real=(-b/2a)
Img=sqrt(d)/2a
Stop

**Program:**

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

int main()
{
   float a,b,c,r1,r2,imgp,realp;
   int d;
   clrscr();
   printf("enter the coefficients of quadratic eqution\n");
   scanf("%f%f%f",&a,&b,&c);
   if(a==0||b==0||c==0) /*condition to check for if one or more coefficients are equal to zero*/

 {
    printf("Please enter non zero coefficients\n");
    retutn 0;
 }

/*find the discremenent*/
 d=b*b-4*a*c;

/*Find the equal roots*/
 if(d==0)
 {
    printf("roots are equal\n");
    r1=r2=-b/(2*a);
    printf("root1 is %f\n",r1);
    printf("root2 is %f",r2);
 }

/*Find the distinct root*/
 else
 if(d>0)
 {
     printf("roots are distinct\n");
     r1=(-b+sqrt(d))/(2*a);
     r2=(-b-sqrt(d))/(2*a);
     printf("root1 is %f\n",r1);
     printf("root2 is %f",r2);
 }
 /*Find the complex roots*/
 else
 {
     printf("roots are complex\n");
     realp=-b/(2*a);
     imgp=sqrt(abs(d))/(2*a);
     printf("root1 is %f+i%f\n",realp,imgp);
     printf("root2 is %f-i%f",realp,imgp);
 }
 return 0;
}
```

**Output:**

**Case 1:**

```
Enter the coefficients of quadratic equation
1 0 2
Please enter non zero coefficients.
```

**Case 2:**

```
Enter the coefficients of quadratic equation
1 2 1
Roots are equal
Root 1 is -1.000000
Root2 is  -1.000000
```

**Case 3:**

```
Enter the coefficients of quadratic equation
1 4 1
Roots are distinct
Root 1 is -0.267949
Root2 is  -3.732051
```

**Case 4:**

```
Enter the coefficients of quadratic equation
2 3 1
Roots are complex
Root 1 is -0.250000+i1.198958
Root2 is  -0.250000-i1.198958
```

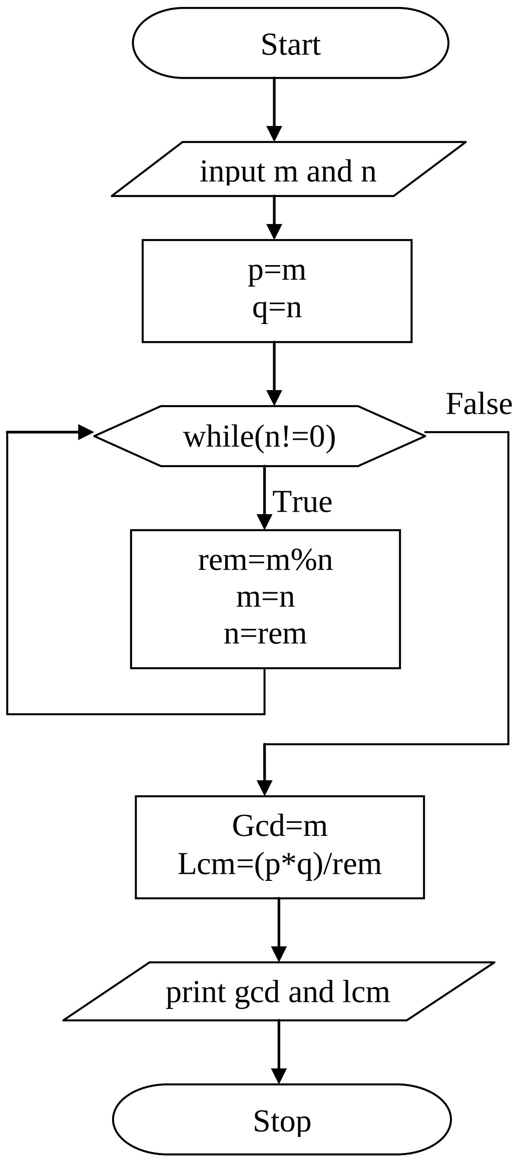
## Program: 02

**Design, develop and execute a program in C to implement Euclid's algorithm to find the GCD and LCM of two integers and to output the results along with the given integers.**

**Algorithm:**

```
step 1:[Input two numbers]
    read m,n
Step 2:[Save the two numbers]
    p=m
    q=n
Step 3:[Compute GcD of two numbers]
    while(n!=0)
        r=m%n
        m=n
        n=r
    end while
Step 4:[Compute GCD and LCM]
    gcd=m
    lcm=p*q/gcd
Step 5:[Display GCD and LCM]
    print GCD and LCM
Step 6:[Finished]
    Stop
```

**Flowchart:**


Start
input m and n
p=m
q=n
while(n!=0)
False
True
rem=m%n
m=n
n=rem
Gcd=m
Lcm=(p*q)/rem
print gcd and lcm
Stop

**Program:**

```
#include<stdio.h>

int main()
{
  int m,n,gcd,lcm,r,p,q;
  printf("enter m and n\n");
  scanf("%d%d",&m,&n);
  p=m;
  q=n;

  while(n!=0)   /*the loop continues untill 'n' becomes equal to  zero*/

    {
       r=m%n; /* r stores the remainder when m is divided by n*/
       m=n;
       n=r;
    }
 gcd=m;
 lcm=(p*q)/gcd; /*formula to find lcm using gcd and input  values*/
 printf("\n the gcd(%d,%d)is %d\n",p,q,gcd);
 printf("\n the lcm(%d,%d) is %d\n",p,q,lcm);
 return 0;
}
```

**Output:**

```
Enter m and n
4        5
The gcd(4,5) is 1
The lcm(4,5) is 20
```

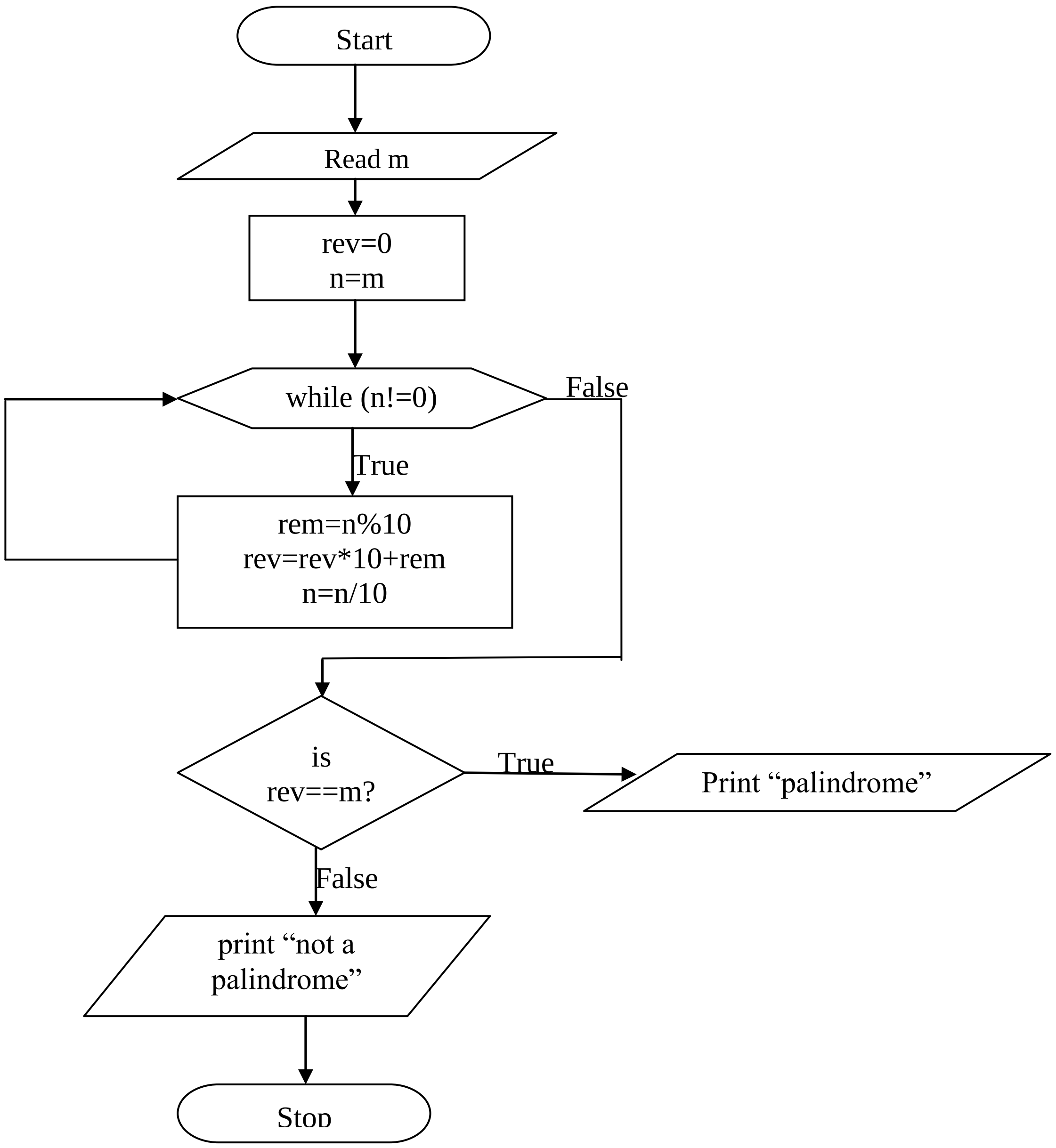
**Program: 03**

**Design, develop and execute a program in C to reverse a given four digit integer number and check whether it is a palindrome or not. Output the given number with suitable message.**

## Algorithm:

```
step 1:[Input the number]
        read n
step 2:[Initialize reverse number rev to 0]
        m=n
        rev=0
step 3:[logic to reverse the number]
        while m!=0
                digit=m%10
                rev=rev*10
                n=n/10
        end while
step 4:[Check for palindrome]
        if(m==rev)
        print "palindrome"
        else
        print "not a palindrome"
step 5:Stop
```

**Flowchart:**


Start
Read m
rev=0
n=m
while (n!=0)
False
True
rem=n%10
rev=rev*10+rem
n=n/10
is
rev==m?
True
Print “palindrome”
False
print “not a
palindrome”
Stop

**Program:**

```
#include<stdio.h>

int main()
{
  int n,rev=0,dig,m;
  printf("enter the number\n");
  scanf("%d",&m);
  n=m;

  while(n!=0) /*The loop continues untill 'n' becomes equal to zero*/
  {
    dig=n%10; /*dig stores remainder when n is divided by 10*/
    rev=rev*10+dig;
    n=n/10;  /*n stores the quotient when n is divided by 10*/
  }

  printf("the reverse of a given number is %d\n",rev);

  if(rev==m) /*condition to check the number is palindrome*/
    printf("the %d number is a palindrome\n",m);
  else
    printf("the %d number is not a palindrome\n",m);
  return 0;
}
```

**Output:**

**Case 1:**

```
Enter the number
121
The reverse number is 121
The 121 number is palindrome
```

**Case 2:**

```
Enter the number
456
The reverse number is 654
The 456 number is not a palindrome
```

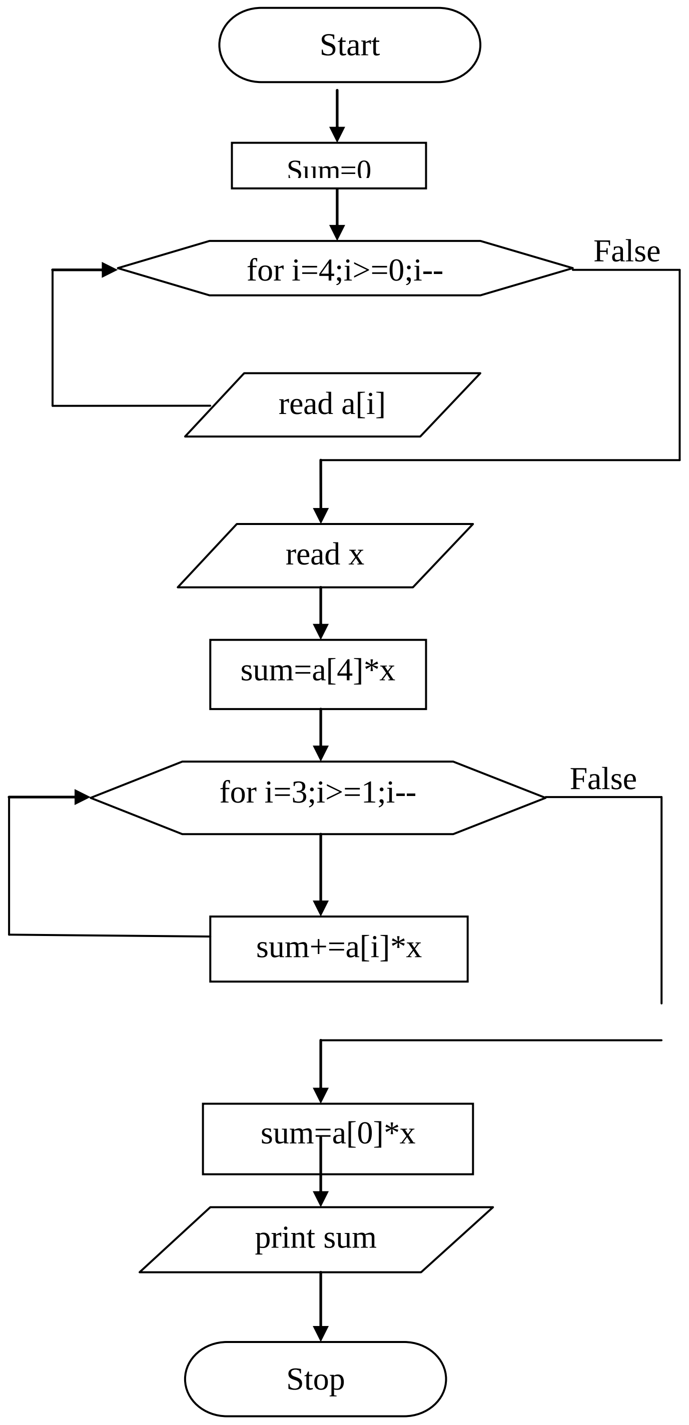
## Program: 04

**Design, develop and execute a program in C to evaluate the given polynomial f(x) = a4x4 + a3x3 + a2x2 + a1x + a0 for given value of x and the coefficients using Horners method.**

**Algorithm:**

```
Step 1:[Read  coefficients from keyboard]
        for i=0 to 4
                read a[i]
        end for
Step 3:[Input value of x]
        read x
Step 4:[Initialize]
        sum=a[4]*x
Step 5:[Compute the result]
        for i=3 down to 1
          sum=(sum+a[i])*x
        end for
Step 6:[Finally add the first term]
        sum=sum+a[0]
Step 7:[output the result]
        print sum
Step 8:[Finished]
        Stop
```

**Flowchart:**


Start
Sum=0
for i=4;i>=0;i--
False
read a[i]
read x
sum=a[4]*x
for i=3;i>=1;i--
False
sum+=a[i]*x
sum=a[0]*x
print sum
Stop

**Program:**

```
 #include<stdio.h>
int main()
 {
   int sum=0,i,x,a[5];
   printf("enter the values for coefficients\n");

   for(i=4;i>=0;i--)
   {
     printf("a[%d]=",i);
     scanf("%d",&a[i]);
   }

  printf("enter the value of x:");
  scanf("%d",&x);

/*Evaluate the polynomial*/
  sum=x*a[4];

  for(i=3;i>=1;i--)
  sum=x*(a[i]+sum);

  sum=sum+a[0];

  printf("result of the given polynomial is f(x)=%d",sum);
  return 0;
}
```

**Output:**

```
Enter the values for coefficients
a[4]=5
a[3]=4
a[2]=3
a[1]=2
a[0]=1
Enter the value of x:1
Result of given polynomial is f(x)=15
```

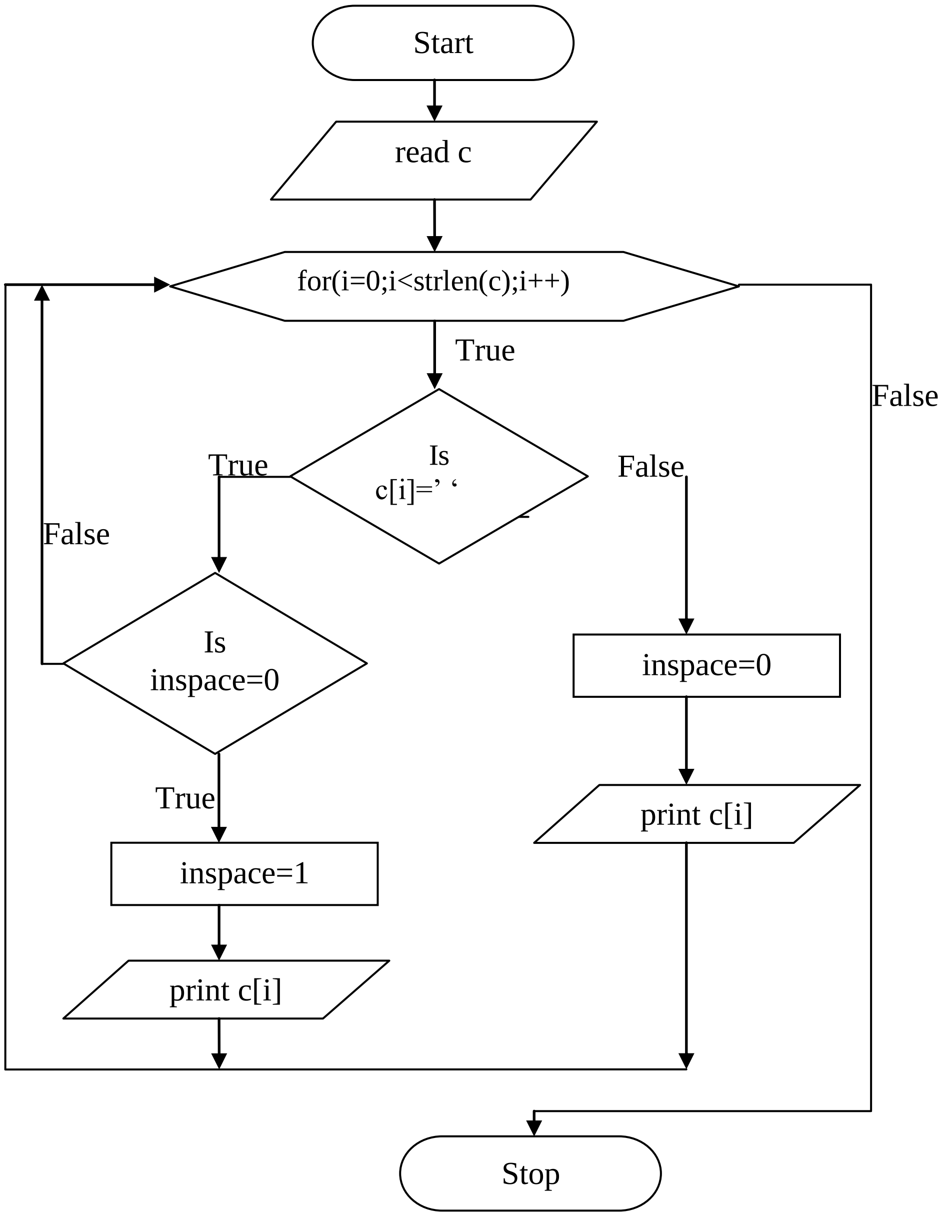
## Program: 05

**Design, develop and execute a program in C to copy its input to its output, replacing each string of one or more blanks by a single blank.**

**Algorithm**

```
Step 1:[Input the string with spaces]
            read c
Step 2:[Initialize]
            inspace=0
Step 3:[Replace string of one or more spaces with one space]
            for i=0 to strlen(c)
                if(str[i]=' ')
                    if(inspace=0)
                           inspace=1
                           print c[i]
                 else
                   inspace=0
                  print c[i]
                end if
              end for
Step 3:[Finished]
            Stop
```

**Flowchart:**


Start
read c
for(i=0;i<strlen(c);i++)
True
False
Is c[i]=' '
True
False
False
Is inspace=0
inspace=0
True
print c[i]
inspace=1
print c[i]
Stop

**Program:**

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

int main()
{
char c[50];
int i,inspace=0;
printf("enter string with space:\n");
gets(c); /*read the string with two or more spaces*/
printf("string with only one space:\n");

/*Replace two or more spaces with one space*/
for(i=0;i<strlen(c);i++)
  {
    if(c[i]==' ')
    {
      if(inspace==0)
      {
            inspace=1;
            printf("%c",c[i]);
      }
    }
    else
    {
      inspace=0;
      printf("%c",c[i]);
    }
  }
  return 0;
}
```

**Output:**

```
Enter string with space:
c  o   m   p uter
string with only one space
c o m p uter
```

## Program: 06

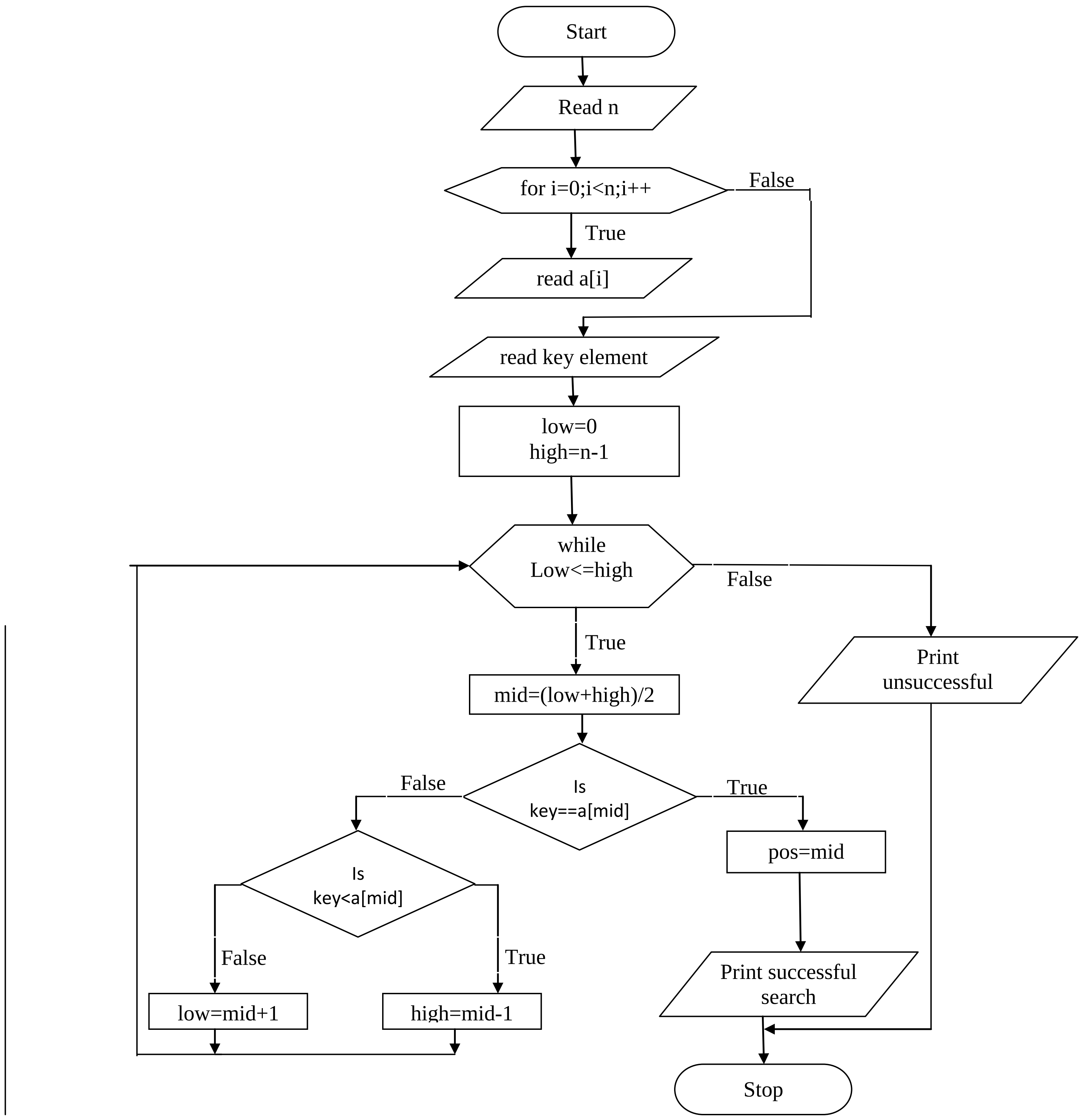
**Design, develop and execute a program in C to input N integer numbers in ascending order into a single dimension array, and then to perform a binary search for a given key integer number and report success or failure in the form of a suitable message.**

**Algorithm:**

```
Step 1:[Input the number]
        read n
Step 2:[Read n elements from keyboard]
        for i=0 to n-1
                read a[i]
        end for
Step 3:[Input the item to be searched]
        read key
Step 4:[Initialisation]
        low=0
        high=n-1
Step 5:[Search the key using binary search]
        while(low<=high)
          mid=(low+high)/2
                if(key==a[mid])
                  print "success"
                  exit(0)
                end if

               else
                if(key<a[mid])
                    high=mid-1
                else
                  low=mid+1
                end if
        end while
          print "unsuccessful"
Step 6:[Finished]
        Stop
```

**Flowchart:**


Start
Read n
for i=0;i<n;i++
False
True
read a[i]
read key element
low=0
high=n-1
while
Low<=high
False
True
Print
unsuccessful
mid=(low+high)/2
Is
key==a[mid]
False
True
pos=mid
Is
key<a[mid]
False
True
Print successful
search
low=mid+1
high=mid-1
Stop

**Program:**

```
 #include<stdio.h>

 int main()
 {
   int a[10],i,n,key,mid,low,high,pos;
   printf("enter the value of n\n");
   scanf("%d",&n);
   printf("enter the elements of the array in ascending order\n");

  for(i=0;i<n;i++)
  {
    scanf("%d",&a[i]);
  }

/*logic of binary search starts*/
  printf("enter the key element\n");
  scanf("%d",&key);
  low=0; /*low limit=1*/
  high=n-1; /*high limit=n-1 initially*/

  while(low<=high) /*loop continues untill low becomes higher  than high*/
  {
    mid=(low+high)/2;
    if(key==a[mid])
      {
       pos=mid;
       printf("the search is successful at pos=%d",pos);
       return 0;
      }
    else
      if(key<a[mid])
            {
            high=mid-1;
            }
      else
            if(key>a[mid])
            {
            low=mid+1;
            }
  }/*End of while*/
 printf("unsuccessful search");
 return 0;
}/*End of main*/
```

**Output:**

**Case 1:**

```
Enter the value of n
5
Enter the elements of array in ascending order
1 2 3 4 5
Enter the key element
5
The search is successfulat pos=4
```

**Case 2:**

```
Enter the value of n
5
Enter the elements of array in ascending order
1 2 3 4 5
Enter the key element
6
The search is unsuccessful
```

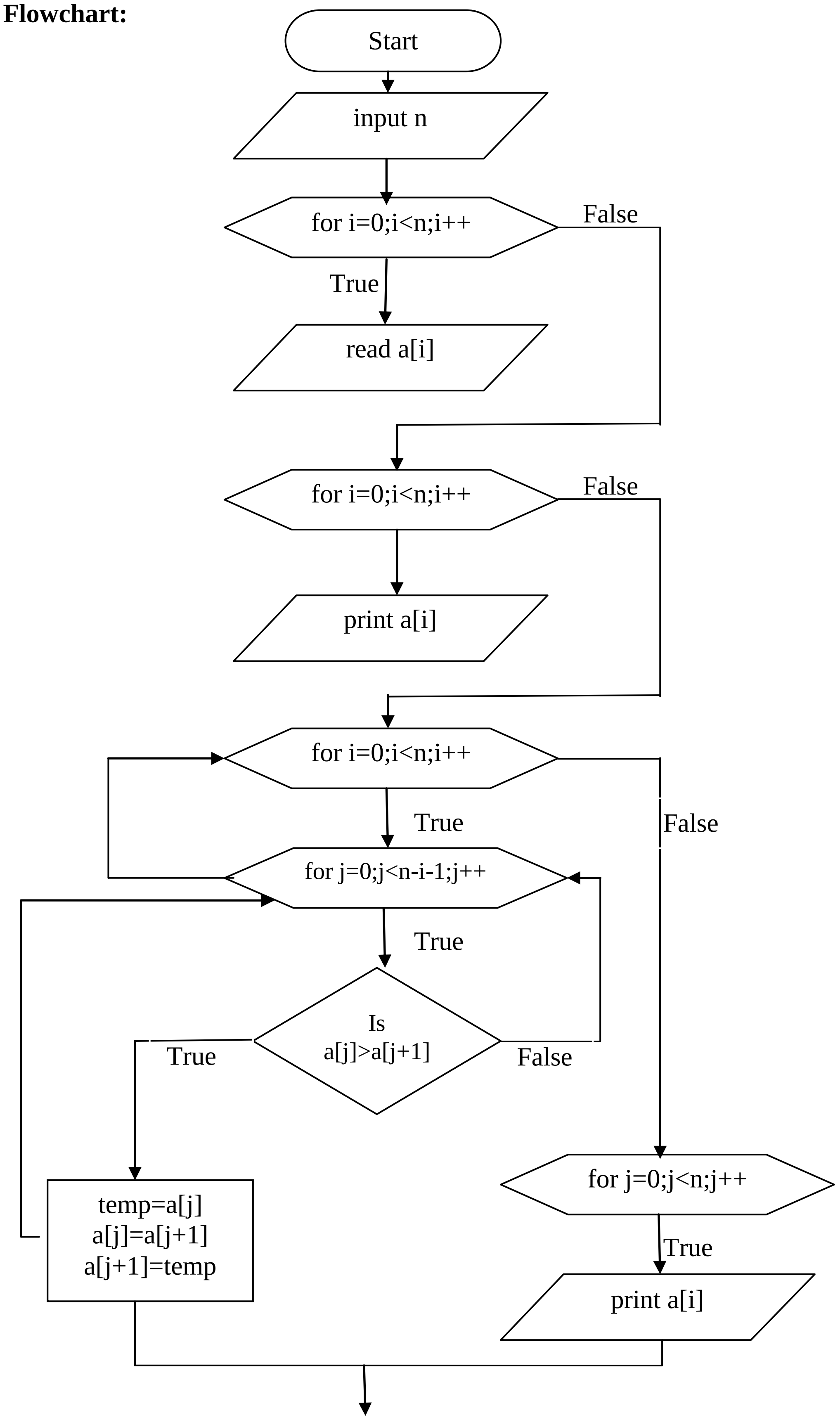
**Program: 07**

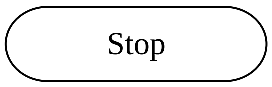
**Design, develop and execute a program in C to input N integer numbers into a single dimension array, sort them in to ascending order using bubble sort technique, and then to print both the given array and the sorted array with suitable headings.**

**Algorithm:**

```
Step 1:[Input the number]
        read n
Step 2:[Read n elements from keyboard]
        for i=0 to n-1
                read a[i]
Step 3:[Sort the elements]
        for i=0 to n-1 do
          for j=0 to j<n-i-1 do
                if(a[j]>a[j+1])
                  temp=a[j];
                  a[j]=a[j+1];
                  a[j+1]=temp;
                 end if
          end for
        end for
Step 4:[Print the sorted elements]
        for i=0 to n-1 do
           print a[i]
        end for
Step 5:[Finished]
        Stop
```

**Flowchart:**


Start
input n
for i=0;i<n;i++
False
True
read a[i]
for i=0;i<n;i++
False
print a[i]
for i=0;i<n;i++
True
False
for j=0;j<n-i-1;j++
True
Is a[j]>a[j+1]
True
False
temp=a[j]
a[j]=a[j+1]
a[j+1]=temp
for j=0;j<n;j++
True
print a[i]



**Program:**

```
 #include<stdio.h>

 int main()
 {
   int i,j,n,a[20],temp;
    printf("enter the number of elements\n");
   scanf("%d",&n);

   printf("enter the array elements\n");

/*read n elements from keyboard*/
   for(i=0;i<n;i++)
   scanf("%d",&a[i]);
   printf("The given array elements are\n");

 /*print the entered elements from keyboard*/
   for(i=0;i<n;i++)
   printf("%d\n",a[i]);

/*sorting of entered elements*/
   for(i=0;i<n;i++)
     {
   for(j=0;j<n-i-1;j++)
      {
      if(a[j]>a[j+1])
            {
            temp=a[j];
            a[j]=a[j+1];
            a[j+1]=temp;
            }
      }
     }
   printf("\nThe array elements after sorting are\n");

/*print sorted elements*/
   for(i=0;i<n;i++)
   printf("%d\n",a[i]);
   return 0;
}
```

**Output:**

```
Enter the values of n
5
Enter the array elements
5 4 3 2 1
The given array elements are
5 4 3 2 1
The array elements after sorting are
1 2 3 4 5
```

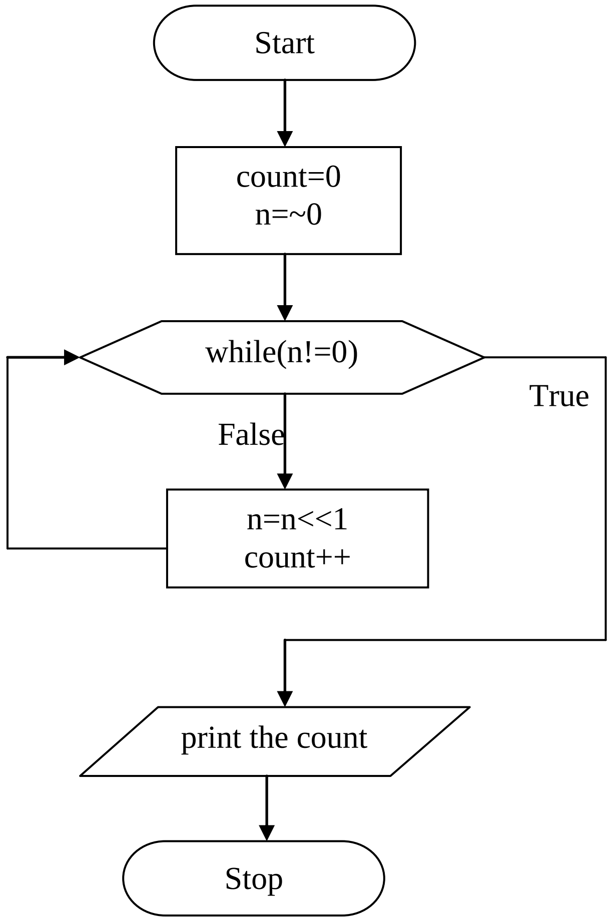
## Program: 08

**Design, develop and execute a program in C to compute and print the word length on the host machine.**

**Algorithm:**

```
Step 1:[Initialize]
        length=0
        n=~0
Step 2:[Find word length of host machine]
        while(n!=0)
          n=n<<1
          length++
        end while
step 3:[display the word length]
        print length
Step 4:[Finished]
        Stop
```

**Flowchart:**



**Program:**

```
#include<stdio.h>

int main()
{
unsigned int n;
int count=0;
n=~0;
clrscr();

/*logic to find the word length*/
while(n!=0)
{
n=n<<1; /*shift n by 1 bit to left*/
count++;
}
printf("word length of the host machine=%d bits\n",count);
return 0;
}
```

**Output:**

```
Word length of the host machine =32 bits
```

## PART - B

### Program: 09

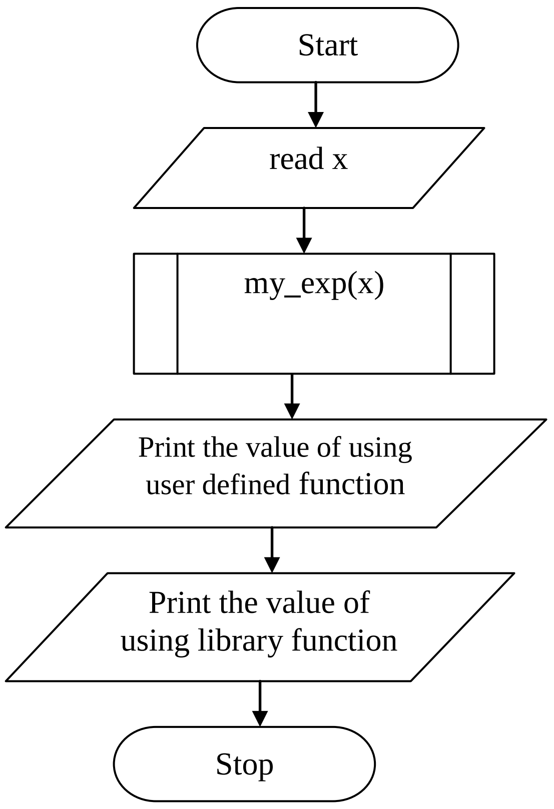
**Design, develop and execute a program in C to calculate the approximate value of exp (0.5) using the Taylor Series expansion for the exponential function. Use the terms in the expansion until the last term is less than the machine epsilon defines as FLT_EPSILON in the header file <float.h>. Print the value returned by the Mathematical function exp ( ) also.**

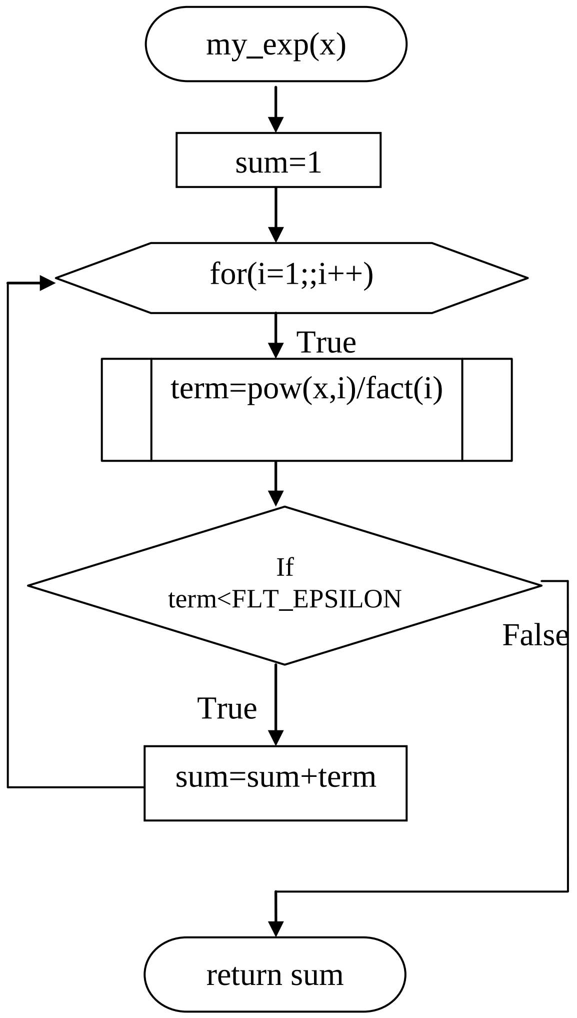
**Algorithm:**

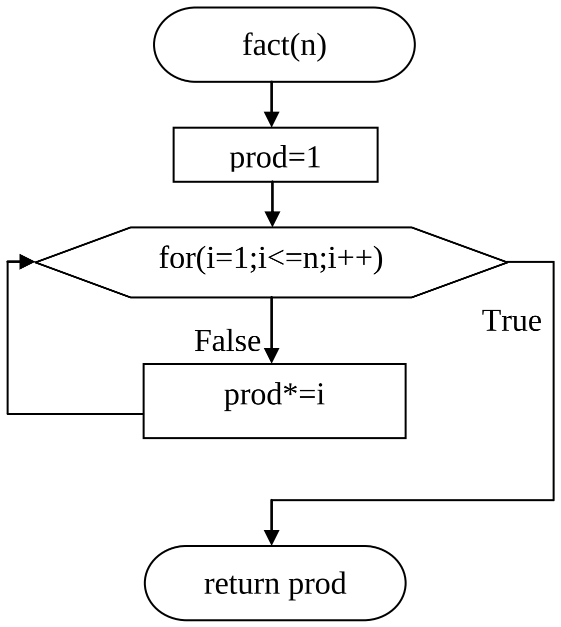
```
Step 1:enter the value of x

Step 2:[logic to find the exp(x)]
           for i=1 to infinite
                term=pow(x,i)/fact(i);
                     if(term<FLT_EPSILON)
                        break
                     end if
             sum=sum+term
            end for
              return sum
Step 3:print the value of exp(x) using user defined function
           Print the  value of exp(x) using mathematical function exp(x)
Step 4:[Finished]
           Stop
```

**Flowchart:**


Start
read x
my_exp(x)
Print the value of using user defined function
Print the value of using library function
Stop


my_exp(x)
sum=1
for(i=1;;i++)
True
term=pow(x,i)/fact(i)
If term<FLT_EPSILON
False
True
sum=sum+term
return sum


fact(n)
prod=1
for(i=1;i<=n;i++)
True
False
prod*=i
return prod

**Program:**

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

/*function to find the factorial*/
int fact(int n)
{
  int i,prod;
  prod=1;
  for(i=1;i<=n;i++)
  {
    prod*=i;
  }
  return prod;
}
/*function to find the exponential of a number*/
double myexp(double x)
{
  int i;
  double sum,term;
  sum=1;
  for(i=1;;i++)
  {
    term=pow(x,i)/fact(i);
    if(term<FLT_EPSILON) /*if the term is less than the  FLT_EPSILON value*/
       {
        break;
       }   /*terminate the for loop*/
    sum=sum+term;
  }
 return sum;
}

int main()
{
  float x;
  double sum;
  clrscr();
  printf("enter value of x:");
  scanf("%f",&x);
  sum=myexp(x);
  printf("using user defined function:\n");
  printf("e^%f=%lf",x,sum);
  printf("\n using library function:\n");
  printf("e^%f=%lf",x,exp(x));
  return 0;
}
```

**Output:**

```
Enter value of x
0.5
Using user defined function
e^0.500000=1.648721
Using library function
e^0.500000=1.648721
```

## Program: 10

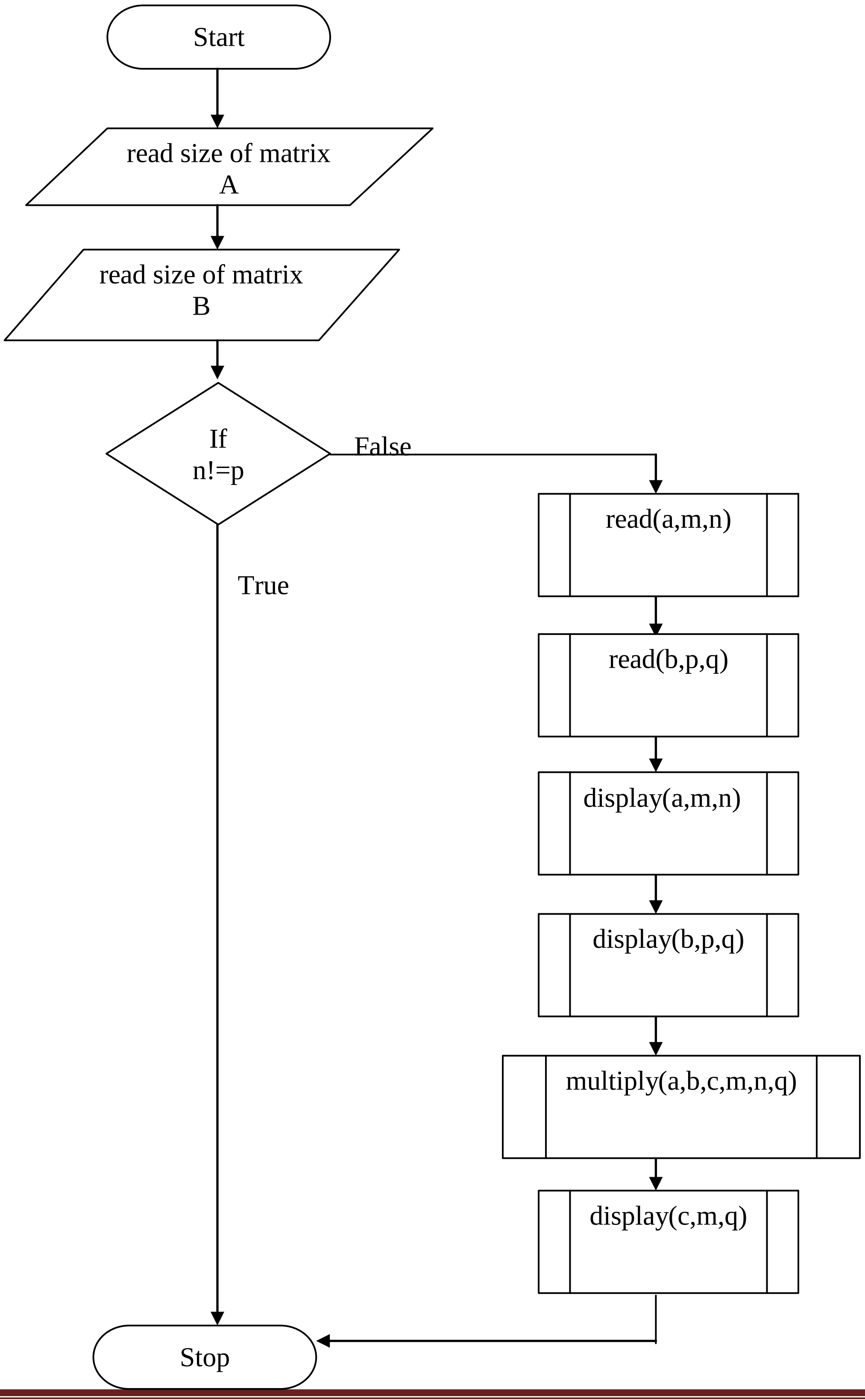
**Design, develop and execute a program in C to read two matrices A (M x N) and B (P x Q) and to compute the product of A and B if the matrices are compatible for multiplication. The program is to print the input matrices and the resultant matrix with suitable headings and format if the matrices are compatible for multiplication, otherwise the program must print a suitable message. (For the purpose of demonstration, the array sizes M, N, P, and Q can all be less than or equal to 3)**

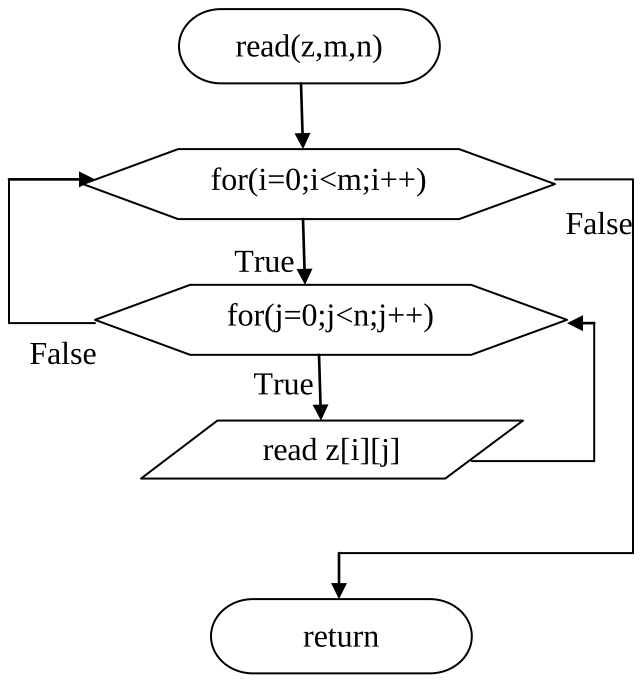
**Algorithm:**

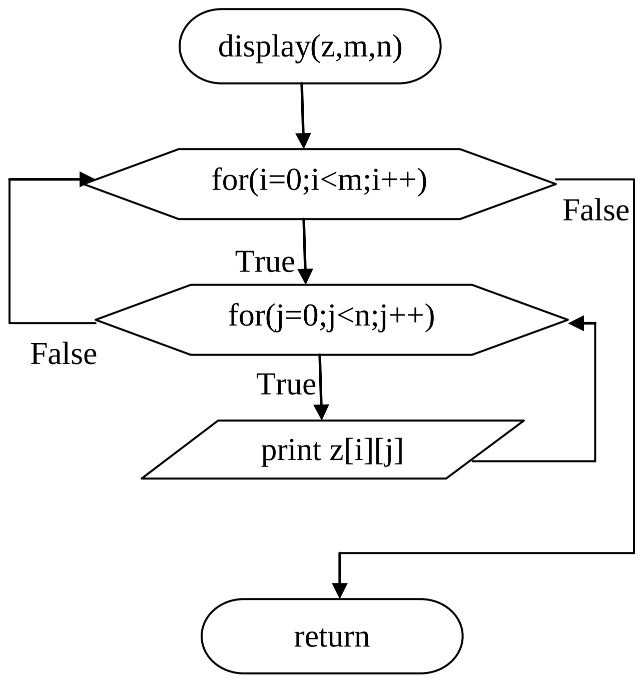
```
Step 1:[Input the size of matrics A]
        read m,n
Step 2:[Input the size of matrics B]
        read p,q
step 3:[If possible multiply two matrices]
        if(n!=p)
          print "multiplication not possible"
          stop
        endif
Step 4:[Input elements of matrix A]
        for i=0 to m-1
           for j=0 to n-1
                read a[i][j]
           end for
        end for
Step 5:[Input elements of matrix A]
        for i=0 to m-1
           for j=0 to n-1
                read a[i][j]
           end for
        end for
Step 6:[Find the product]
     for i=0 to i<m
        for j=0 to j<q
          sum=0
            for k=0 to k<n
              sum=sum+a[i][k]*b[k][j]
            end for
           c[i][j]=sum
          end for
        end for
step 7:[Output the product matrix]
        for i=0 to m-1
           for j=0 to n-1
                print c[i][j]
           end for
        end for
Step 8:[Finish]
```

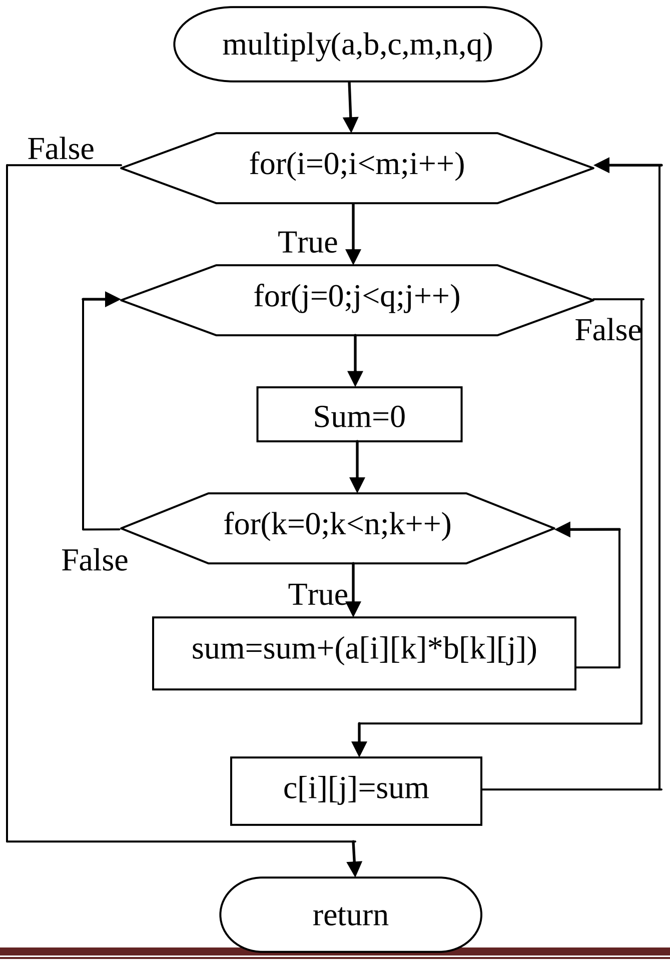
Stop

**Flowchart:**




read(z,m,n)
for(i=0;i<m;i++)
True
False
for(j=0;j<n;j++)
True
False
read z[i][j]
return


display(z,m,n)
for(i=0;i<m;i++)
True
False
for(j=0;j<n;j++)
True
False
print z[i][j]
return


multiply(a,b,c,m,n,q)
False
for(i=0;i<m;i++)
True
for(j=0;j<q;j++)
False
Sum=0
for(k=0;k<n;k++)
False
True
sum=sum+(a[i][k]*b[k][j])
c[i][j]=sum
return

**Program:**

```
  #include<stdio.h>
  void read(int z[10][10],int m,int n);
  void display(int z[10][10],int m,int n);
  void multiply(int a[10][10],int b[10][10],int c[10][10],int m,int n,int q);

  int main()
 {
   int m,n,p,q,a[10][10],b[10][10],c[10][10];
   clrscr();

/*enter the size of matrix a and b from keyboard*/
   printf("enter size of matrix A\n");
   scanf("%d%d",&m,&n);
   printf("enter size of matrix B\n");
   scanf("%d%d",&p,&q);

  if(n!=p) /*check if n is not equal to p*/
  {
   printf("multiplication is not possible\n");
   return 0;
  }

/*read the elements of matrix a and b from keyboard*/
   printf("enter the elements of matrix A\n");
   read(a,m,n);
   printf("enter elements of matrix B\n");
   read(b,p,q);

/*display the matrix a and b*/
   printf("Matrix A\n");
   display(a,m,n);
   printf("Matrix B\n");
   display(b,p,q);

   multiply(a,b,c,m,n,q);
   printf("product is:\n");
   display(c,m,q);/*display the product of two matrices*/
   return 0;
 }
```

**/*fuction to read the matrix from keyboard*/**

```
 void read(int z[10][10],int m,int n)
 {
   int i,j;
   for(i=0;i<m;i++)
   {
     for(j=0;j<n;j++)
    {
      scanf("%d",&z[i][j]);
    }
   }
 }
```

**/*function to print the matrix*/**

```
  void display(int z[10][10],int m,int n)
  {
    int i,j;
    for(i=0;i<m;i++)
    {
      for(j=0;j<n;j++)
     {
       printf("%d\t",z[i][j]);
     }
    printf("\n");
    }
  }
```

**/*function to multiply two matrices*/**

```
  void multiply(int a[10][10],int b[10][10],int c[10][10],int m,int n,int q)
  {
    int i,j,k,sum;
   for(i=0;i<m;i++)
   {
     for(j=0;j<q;j++)
     {
       sum=0;
       for(k=0;k<n;k++)
       {
              sum=sum+a[i][k]*b[k][j];
       }
       c[i][j]=sum;
     }
   }
 }
```

**Output:**

**Case 1:**

```
Enter the size of matrix A
2        2
Enter size of matrix B
2       3
Enter elements of matrix A
1        1       1        1
Enter elements of matrix B
2       2       2       2       2       2
Matrix A
1       1
1       1
Matrix B
2       2       2
2       2       2
The product matrix C
4       4       4
4       4       4
```

**Case 2:**

```
Enter the size of matrix A
2       2
Enter size of matrix b
1       1
Multiplication not possible
```

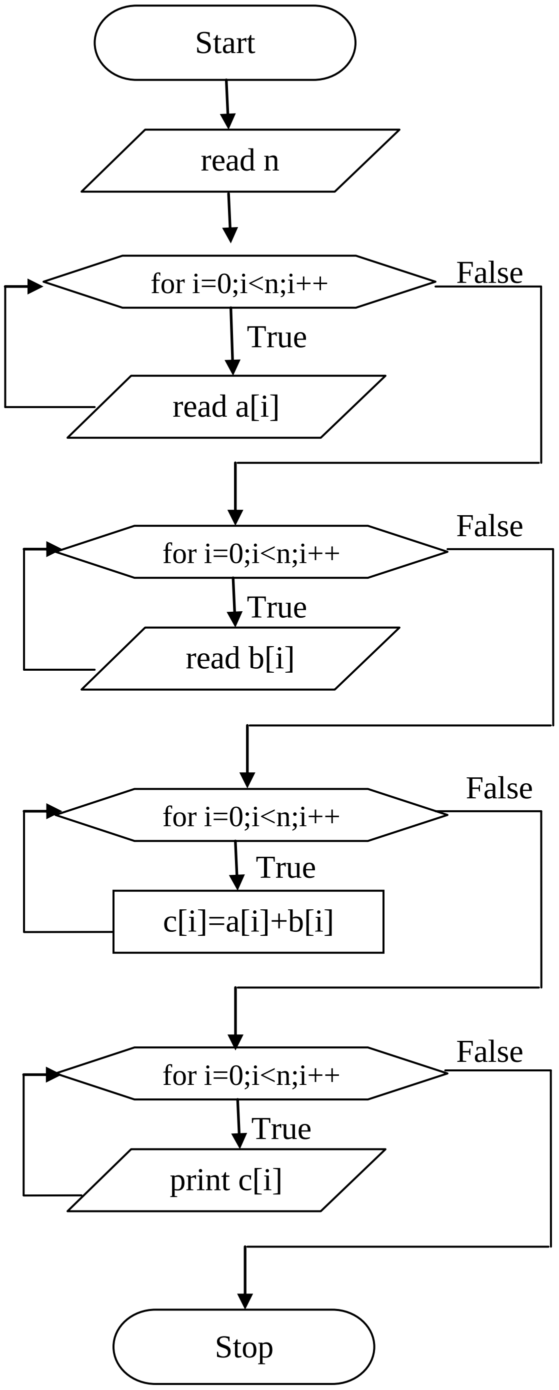
**Program: 11**

**Design, develop and execute a parallel program in C to add, element-wise, two one-dimensional arrays A and B of N integer elements and to store the result in another one-dimensional array C of N integer elements.**

**Algorithm:**

```
Step 1:[Input size of array]
          read the value of n
Step 2: [Enter the elements of array A]
        for i=0 to n-1
          Read elements a[i]
        end for
Step 3: [Enter the elements of array B]
        for i=0 to n-1
           read elements b[i]
       end for
Step 4: [parallelize following code]
          for i=0 to n-1
             c[i]=a[i]+b[i]
          end for
Step 5:[display array c]
         for i=0 to n-1
             print c[i]
         endfor
Step 6:[finish]
        Stop
```

**Flowchart:**


Start
read n
for i=0;i<n;i++
False
True
read a[i]
for i=0;i<n;i++
False
True
read b[i]
for i=0;i<n;i++
False
True
c[i]=a[i]+b[i]
for i=0;i<n;i++
False
True
print c[i]
Stop

**Program:**
```
#include<stdio.h>
#include<omp.h>

int main()
{

int a[10],b[10],c[10],i, n;
omp_set_num_threads(5); /*set the thread ids range from 0-4*/

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

printf("\nEnter the element of 1st array");
for(i=0;i<n;i++)
scanf("%d",&a[i]);

printf("Enter the elements of 2nd array");
for(i=0;i<n;i++)
scanf("%d",&b[i]);

printf("\nThe contents of array A\n");
for(i=0;i<n;i++)
{
printf("\na[%d]=%d",i,a[i]);
}

printf("\nThe contents of array B\n");
for(i=0;i<n;i++)
{
printf("\nb[%d]=%d",i,b[i]);
}

#pragma omp parallel for shared(c)    /*parallelize the for loop*/
for(i=0;i<n;i++)
{
#pragma omp critical(c_lock) /*allows single thread to enter*/
c[i]=a[i]+b[i];
printf("\nc[%d]=%d,thread id=%d\n",i,c[i],omp_get_thread_num());
}

printf("\nThe new array is====\n");
for(i=0;i<n;i++)
{
printf("c[%d]=[%d]\n",i,c[i]);
}
```

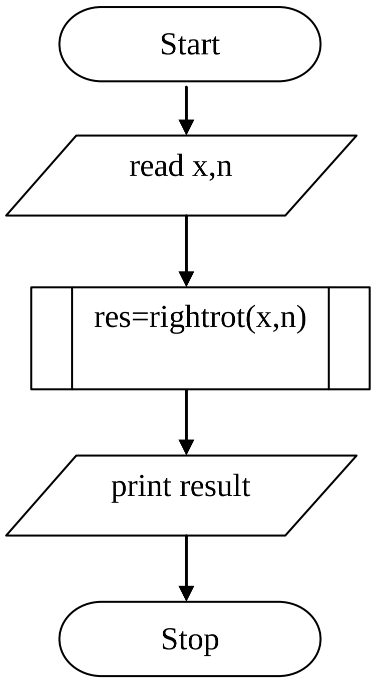
**Program: 12**

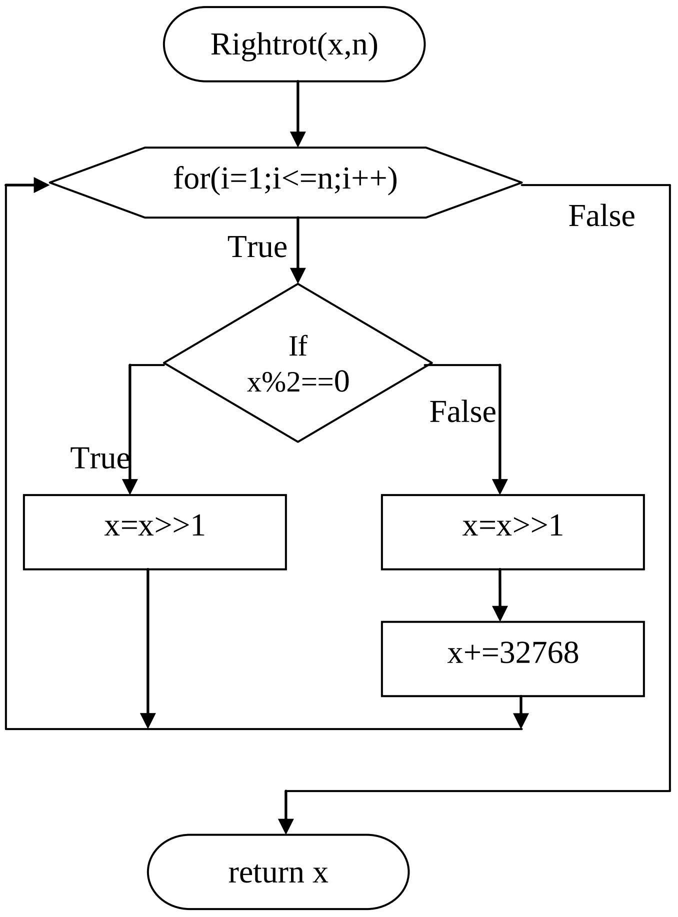
**Design and develop a function rightrot (x, n) in C that returns the value of the integer x rotated to the right by n bit positions as an unsigned integer. Invoke the function from the main with different values for x and n and print the results with suitable headings.**

**Algorithm:**

```
Step 1:[Input the unsiged integer]
        read x
Step 2:[Input an integer to indicate how many times to rotate]
        read n
Step 3:[Rotate the unsigned integer]
        for i=1 to n
          if(x%2==0)
            x=x>>1
          else
            x=x>>1
            x+=32768
          end if
        end for
Step 4:[Display the result]
        print x
Step 5:[Finished]
        Stop
```

**Flowchart:**


Start
read x,n
res=rightrot(x,n)
print result
Stop


Rightrot(x,n)
for(i=1;i<=n;i++)
True
False
If x%2==0
True
False
x=x>>1
x=x>>1
x+=32768
return x

**Program:**

```
 #include<stdio.h>
 unsigned int rightrot(unsigned int x,int n);

int main()
 {
   unsigned int x,res;
   int n;
   clrscr();
   printf("enter value of x:");
   scanf("%u",&x);
   printf("enter how many times to rotate:");
   scanf("%d",&n);
   res=rightrot(x,n);
   printf("result=%u",res);
   return 0;
}

unsigned int rightrot(unsigned int x,int n)
{
 int i;
 for(i=1;i<=n;i++)
 {
   if(x%2==0) /*checks if last bit is zero*/
   {
    x=x>>1; /*right shift operation by one bit*/
   }
   else     /*if last bit has one else part will be executed*/
   {
    x=x>>1;
    x+=32768;
   }
 }
 return x;
}
```

**Output:**
**Case 1:**

```
Enter value of x:4
Enter how many times to rotate:2
Result=1
```

**Case 1:**

```
Enter value of x:3
Enter how many times to rotate:3
Result=32769
```

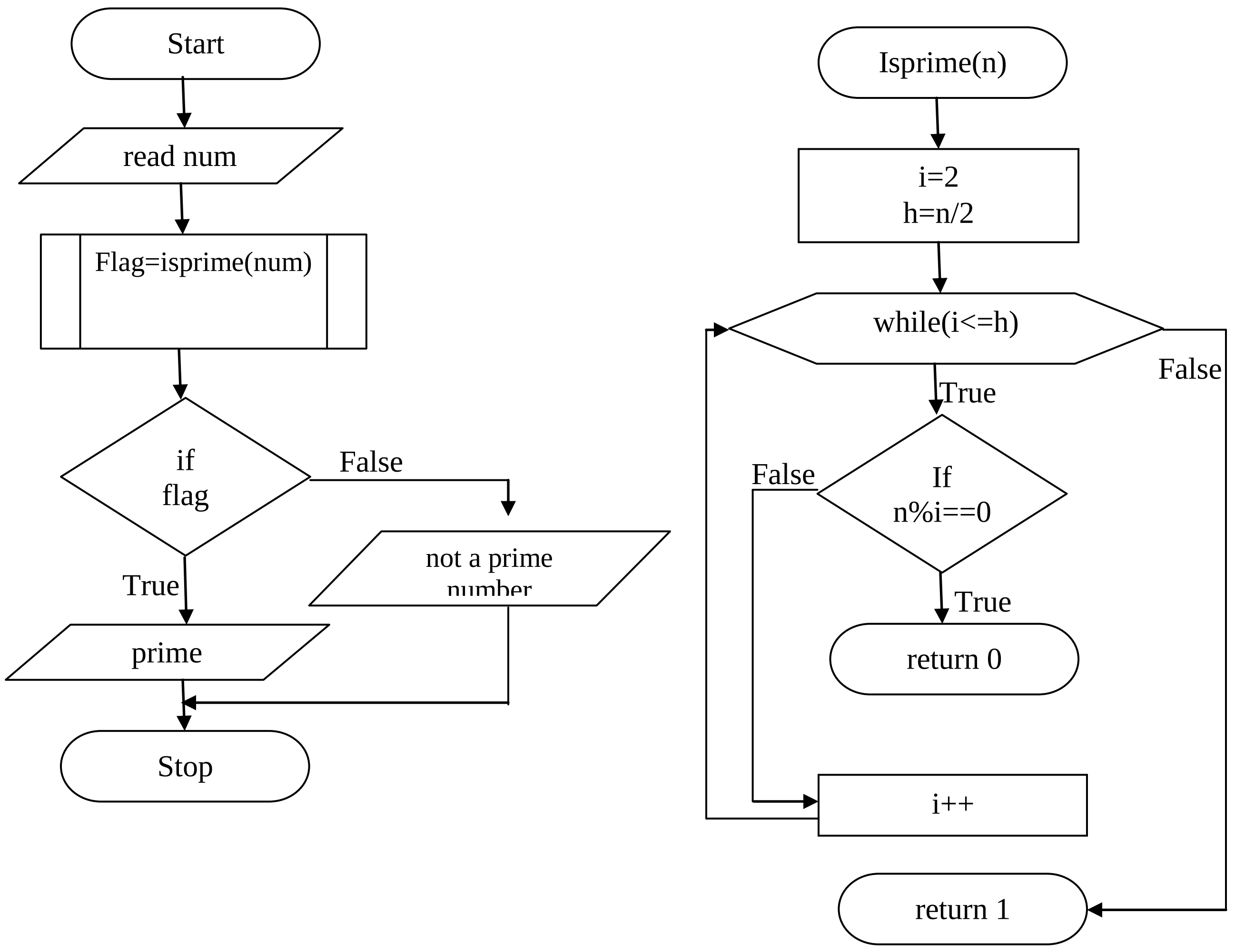
**Program: 13**

**Design and develop a function isprime (x) that accepts an integer argument and returns 1 if the argument is prime and 0 otherwise. The function is to use plain division checking approach to determine if a given number is prime. Invoke this function from the main with different values obtained from the user and print appropriate messages.**

**Algorithm:**

```
Step 1:[Input the number]
        read n
Step 2:[Initialize]
        i=2
        h=n/2
Step 3:[Find whether the number prime or not]
        while(i<=h)
          if(n%i==0)
            return 0
          end if
           i++
        end while
          return 1
Step 4:[Display the result]
        print the number
Step 5:[Finished]
        Stop
```

**Flowchart:**


Start
read num
Flag=isprime(num)
if flag
False
True
not a prime number
prime
Stop
Isprime(n)
i=2
h=n/2
while(i<=h)
False
True
False
If n%i==0
True
return 0
i++
return 1

**Program:**

```
  #include<stdio.h>
  int isprime(int n);

/*function to check whether the nuber is prime or not*/
  int isprime(int n)
 {
   int i=2,h=n/2;
   while(i<=h)
   {
      if(n%i==0)
      {
            return 0; /*return zero to the calling function if the number is divisible by i*/
      }
    i++;
   }
   return 1; /*return one to the calling function*/
 }

 int main()
 {
   int n,flag,q=0,choice;
   while(q!=1)
   {
   clrscr();
   printf("\nEnter 1. To check whether the number is prime or not\n");
   printf("Enter 2. To quit\n");
   printf("enter your choice\n");
   scanf("%d",&choice);
    switch(choice)
    {
     case 1:printf("enter the number\n");
                scanf("%d",&n);
                flag=isprime(n);
                if(flag)
                         printf("%d is prime",n);
                else
                         printf("%d is not prime",n);
                break;
    case 2:printf("quitting from the program\n");
                  q=1;
                  break;
    default:printf("invalid choice\n");
                  break;
    }
    return 0;
   }
}
```

**Output:**

**Case 1:**

```
Enter 1. To check whether given nu mber is prime or not
Enter 2. To quit
Enter your choice
1
Enter the number: 11
11 is a prime number
```

**Case 2:**

```
Enter 1. To check whether given nu mber is prime or not
Enter 2. To quit
Enter your choice
1
Enter the number: 4
4 is not prime number
```

**Case 3:**

```
Enter 1. To check whether given nu mber is prime or not
Enter 2. To quit
Enter your choice
2
Quitting from the program
```

**Case 4:**

```
Enter 1. To check whether given nu mber is prime or not
Enter 2. To quit
Enter your choice
6
Invalid choice
```

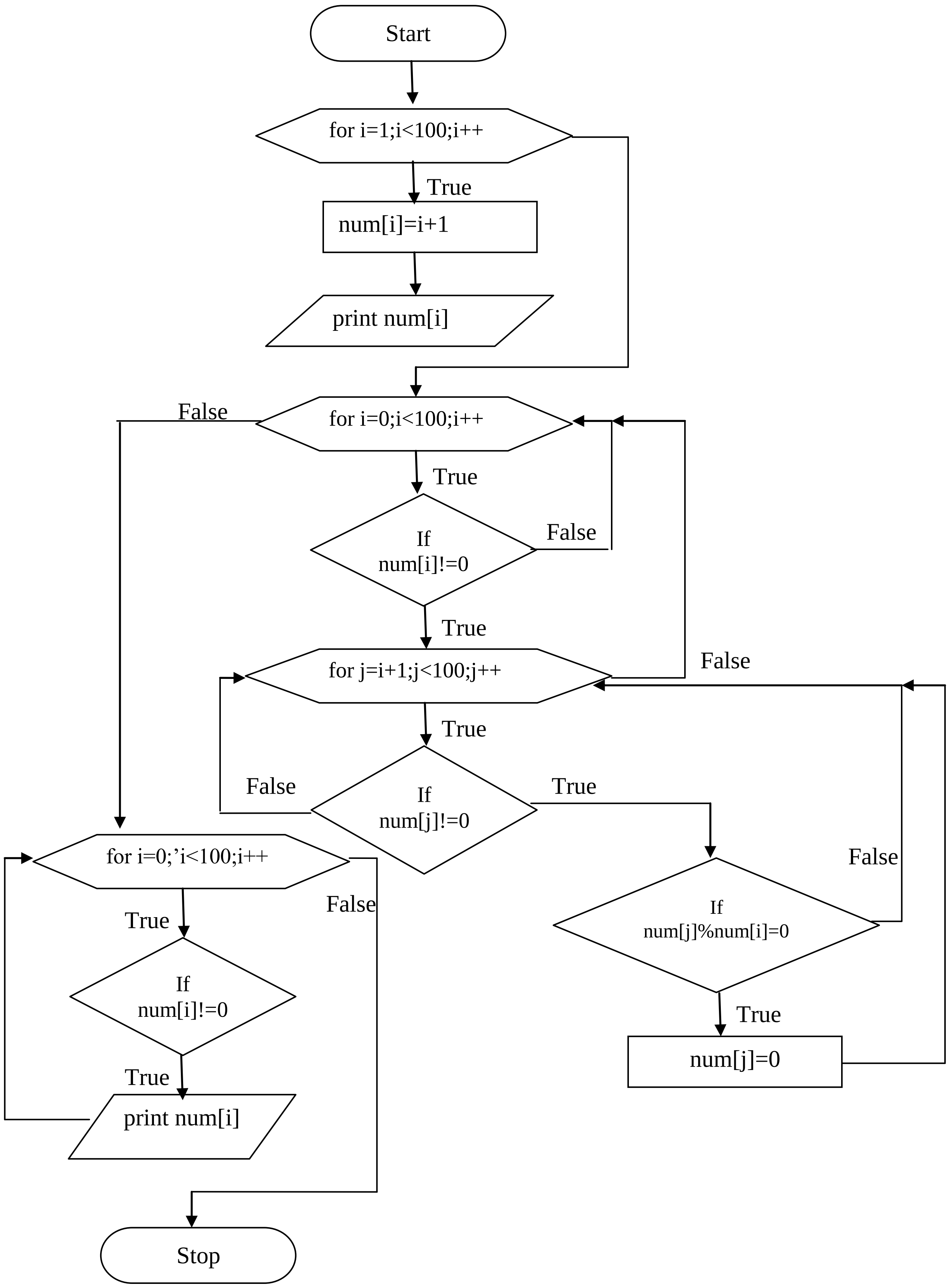
## Program: 14

**Design, develop and execute a parallel program in C to determine and print the prime numbers which are less than 100 making use of algorithm of the Sieve of Eratosthenes.Using these function.**

**Algorithm:**

```
Step 1:[print the value from 1 to 100]
             For i=1 to i<100
               num[i]=i+1;
              print num[i]
            end for
Step 2: [find the prime elements]
           for(i=1;i<100;i++)
                 if(num[i]!=0)
                          for(j=(i+1);j<100;j++)
                                 if(num[j]!=0)
                                    if((num[j]%num[i])==0)
                                    num[j]=0
                                    end if
                               end if
                        endfor
                    endif
               endfor
step 3:[display the prime numbers from 1 to n100]
            for(i=0;i<100;i++)
                  if(num[i]!=0)
                       print num[i]
                 endif
            endfor
Step 4:[finish]
           Stop
```

**Flowchart:**


Start
for i=1;i<100;i++
True
num[i]=i+1
print num[i]
False
for i=0;i<100;i++
True
If num[i]!=0
False
True
for j=i+1;j<100;j++
False
True
If num[j]!=0
False
True
If num[j]%num[i]=0
False
True
num[j]=0
for i=0;'i<100;i++
True
False
If num[i]!=0
True
print num[i]
Stop

**Program:**

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

int main()
{

    int num[100],i,j;
    omp_set_num_threads(10); /*set the thread ids range from 0-9*/

    #pragma omp parallel for  /*parallelize the for loop*/
           for(i=0;i<100;i++)
           {
           num[i]=i+1;
        printf("\nnum[%d]=%d,thread id=%d\n",i,num[i],omp_get_thread_num());
           }

 for(i=1;i<100;i++)
        {
        if(num[i]!=0)
        {
                    for(j=(i+1);j<100;j++)
                    {
                    if(num[j]!=0)
                          {
                             if((num[j]%num[i])==0)
                             num[j]=0;

                          }

                    }
        }

        }

    printf(" \nThe prime numbers between 1 to 100 is \n");
    for(i=0;i<100;i++)
    {
        if(num[i]!=0)
            printf("  %d",num[i]);
    }
printf("\n");
}
```

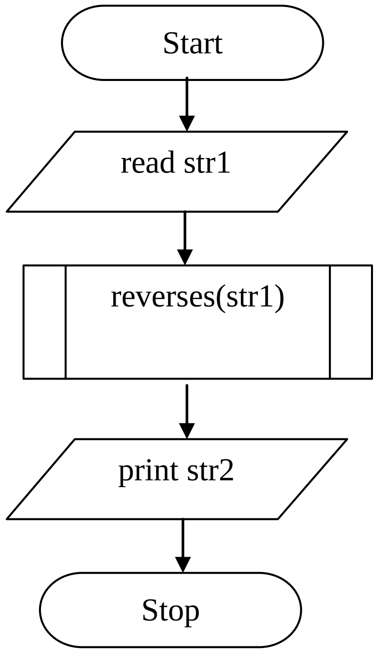
**Program: 15**

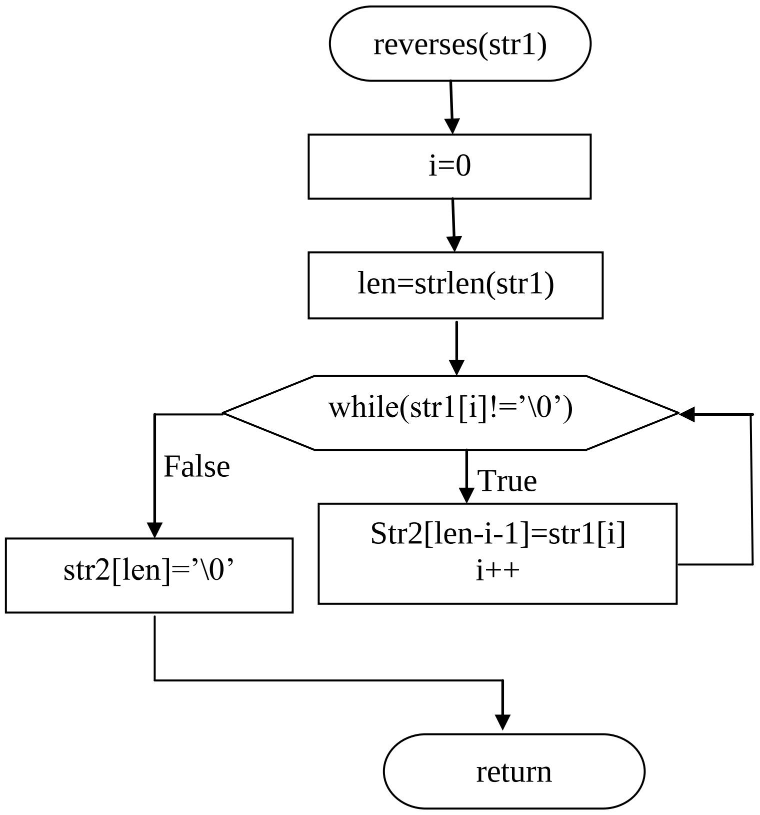
**Design and develop a function reverses (s) in C to reverse the string s in place. Invoke this function from the main for different strings and print the original and reversed strings.**

**Algorithm:**

```
Step 1:[Input the string]
            read str1
Step 2:[Reverse the string]
            for i=0 to i<len
                str2[len-1-i]=str1[i]
            end for
        str2[len]='\0'
Step 3:[Display the reversed string]
            print str2
Step 4:[Finished]
            Stop
```

**Flowchart:**





**Program:**

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

 void reverses(char str1[]);
 char str2[40];
 /*function to reverse string*/
 void reverses(char str1[])
 {
   int i=0,len;
   len=strlen(str1); /*len varible stores the length of the string returned from the function
                                      strlen(str1)*/
while(str1[i]!='\0')
   {
     str2[len-i-1]=str1[i];
     i++;
   }
   str2[len]='\0';
   return;
 }

int main()
 {
  char str1[40];
  int q=0,choice;
  while(q!=1)
   {
    printf("enter 1. to reverse the string\n");
    printf("enter 2. to quit\n");
    printf("enter your choice\n");
    scanf("%d",&choice);
    switch(choice)
     {
      case 1:printf("enter a string\n");
             scanf("%s",str1);
             reverses(str1);
             printf("\nthe original string is:%s\n",str1);
             printf("the reversed string is:%s\n",str2);
             break;
      case 2:printf("quitting from the program\n");
             q=1;
             break;
      default:printf("invalid choice\n");
              break;
      }
    return 0;
    }
  }
```

**Output:**

**Case 1:**

```
Enter 1. TO reverse a string
Enter 2. To quit
Enter your choice
1
Enter a string: computer
The original string is computer
The reversed string is retupmoc
```

**Case 2:**

```
Enter 1. TO reverse a string
Enter 2. To quit
Enter your choice
2
Quitting from the program
```

**Case 2:**

```
Enter 1. TO reverse a string
Enter 2. To quit
Enter your choice
7
Invalid choice
```

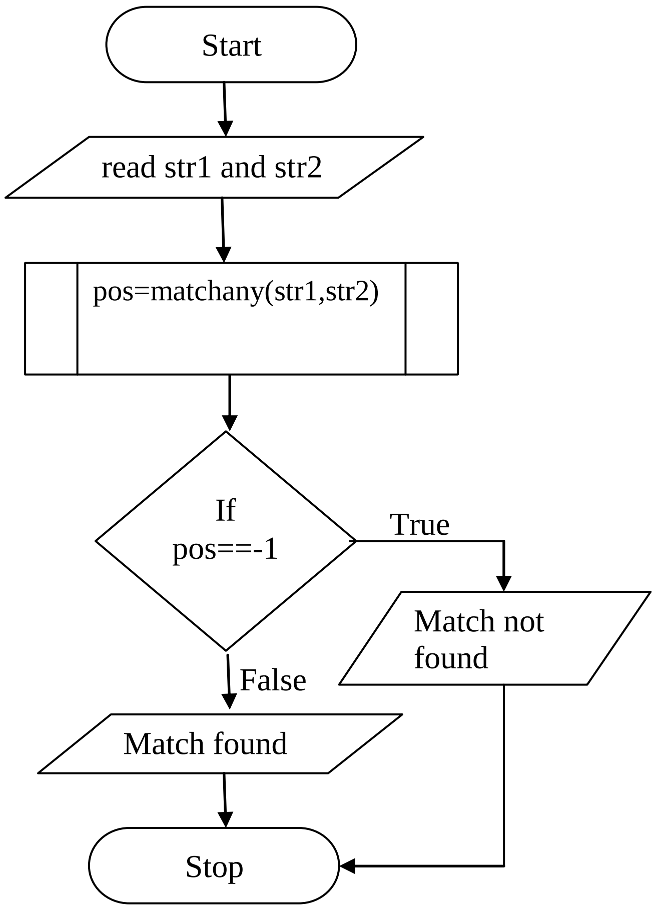
## Program: 16

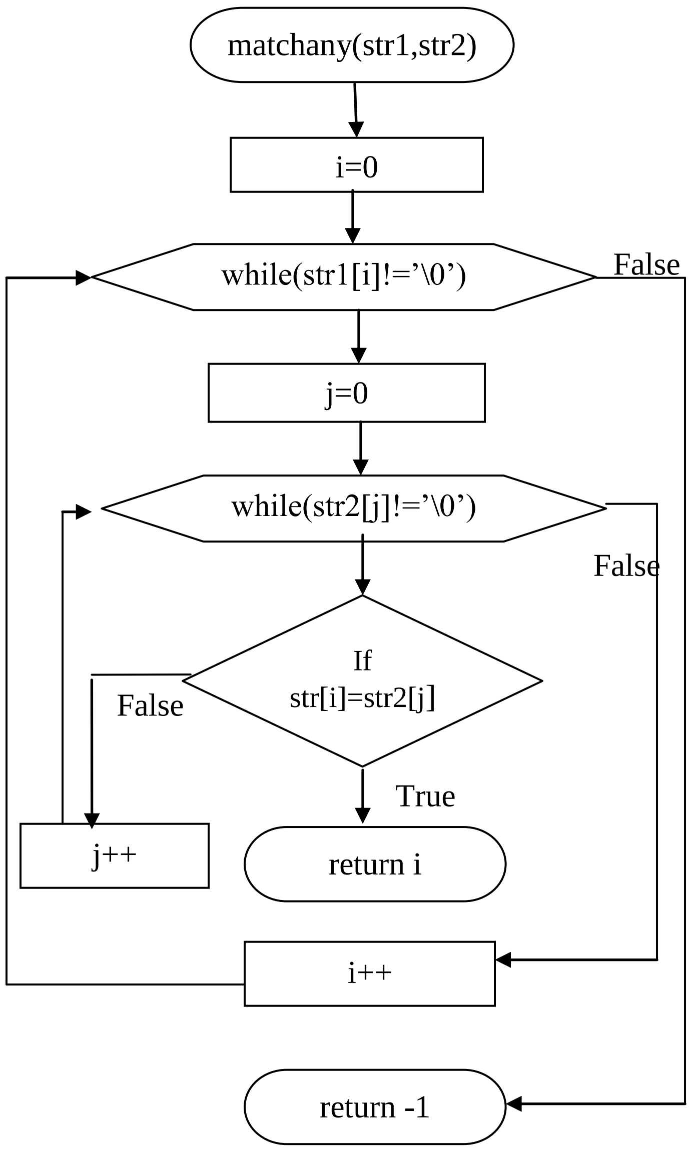
**Design and develop a function matchany (s1, s2) which returns the first location in the string s1 where any character from the string s2 occurs, or -1 if s1 contains no character from s2. Do not use the standard library function which does a similar job! Invoke the function matchany (s1. s2) from the main for different strings and print both the strings and the return value from the function matchany (s1,s2).**

**Algorithm:**

```
Step 1:[Input two strings]
        read str1
        read str2
Step 2:[Initialize]
        i=0
        j=0
Step 3:[Find the match in the string]
        while(str1[i]!='\0')
          j=0
          if(str1[i]==str2[j])
               return i
               stop
          j++
         i++
        end while
Step 4:[Display the position]
        print i
Step 5:[Finished]
        Stop
```

**Flowchart:**


Start
read str1 and str2
pos=matchany(str1,str2)
If pos==-1
True
Match not found
False
Match found
Stop


matchany(str1,str2)
i=0
while(str1[i]!='\0')
False
j=0
while(str2[j]!='\0')
False
If str[i]=str2[j]
False
True
j++
return i
i++
return -1

**Program:**

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

int matchany(char str1[],char str2[])
 {
   int i=0,j=0;
   while(str1[i]!='\0') /*loop is executed untill null character is found*/
   {
    j=0;
    while(str2[j]!='\0') /*loop is executed untill null character is found*/
     {
      if(str1[i]==str2[j]) /*checks if the characters are equal*/
       return i; /*if condition is true, returns i as position*/
      j++;
     }
    i++;
   }
 return -1; /*if match is not found,returns -1*/
 }

int main()
{
 char str1[40],str2[40];
 int q=0,choice,pos;
 while(q!=1)
 {
  clrscr();
  printf("enter 1. to find matching character in the string\n");
  printf("enter 2. to quit\n");
  printf("enter your choice\n");
  scanf("%d",&choice);
  switch(choice)
   {
    case 1:printf("enter string1\n");
           scanf("%s",str1);
           printf("\nenter string2\n");
           scanf("%s",str2);
           pos=matchany(str1,str2);

           if(pos==-1)
             printf("match not found\n");
           else
             printf("match found at position=%d",pos);

           break;
    case 2:printf("quitting from the program");
               q=1;
               break;
    default:printf("Invalid choice\n");
                break;
   }
  return 0;
 }
}
```

**Output:**

**Case 1:**

```
Enter 1. TO find the matching character in the string
Enter 2. To quit
Enter your choice
1
Enter a string1: computer
Enter a string2: hello
Match found at position 1
```

**Case 2:**

```
Enter 1. To find the matching character in the string
Enter 2. To quit
Enter your choice
1
Enter a string1: book
Enter a string2: string
Match not found
```

**Case 3:**

```
Enter 1. To reverse a string
Enter 2. To quit
Enter your choice
2
Quitting from the program
```

**Case 4:**

```
Enter 1. To check whether given number is prime or not
Enter 2. To quit
Enter your choice
6
Invalid choice
```

## ***CCP VIVA-VOCE QUESTIONS***

1. What do you mean by Hardware and Software?
2. Mention the main components of computer and their functions.
3. What is operating system (OS)?
4. What is Algorithms?
5. What is Flowchart?
6. Name the four basic data types in "C" language.
7. Describe at least five different format specifiers.
8. Define and explain scanf() function.
9. Define and explain printf() function.
10. What are the maximum and minimum possible ranges of values for long and short type?
11. What is Preprocessors?
12. What exactly is a 'variable scope', 'local variables', and global variables?
13. What are signed values?
14. Define reserved words.
15. What is the purpose of type declaration in C?
16. What is identifier?
17. Mention the different types of operators used in C.
18. What is loop control statement?
19. Explain *while* loop.
20. Explain *for* loop.
21. What do you mean by network?
22. List a few unconditional control statements in C.
23. What is an array?
24. What is multidimensional array?
25. Define string.

26. Define and explain about logical Operator.
27. What is operator precedence?
28. Explain about the functions strcat() and strcmp().
29. Define functions.
30. Differentiate built-in functions and user-defined functions.
31. Distinguish between actual and formal arguments.
32. Explain the concept and use of type void.
33. What is recursion?
34. Mention the types of network.
35. What are library functions?
36. How does the type float differ from double in C language?
37. What is an operator and operand?
38. Differentiate between RAM and ROM.
39. Define system software.
40. Define application software.
41. What are control systems?
42. What is Parallel Computations?
43. Why use Parallel Computations?
44. What does openMP stands for? Explain the compiling of openMP programs.
45. Explain increment and decrement operators.
46. Mention the types of memory.
47. What are input and output devices?