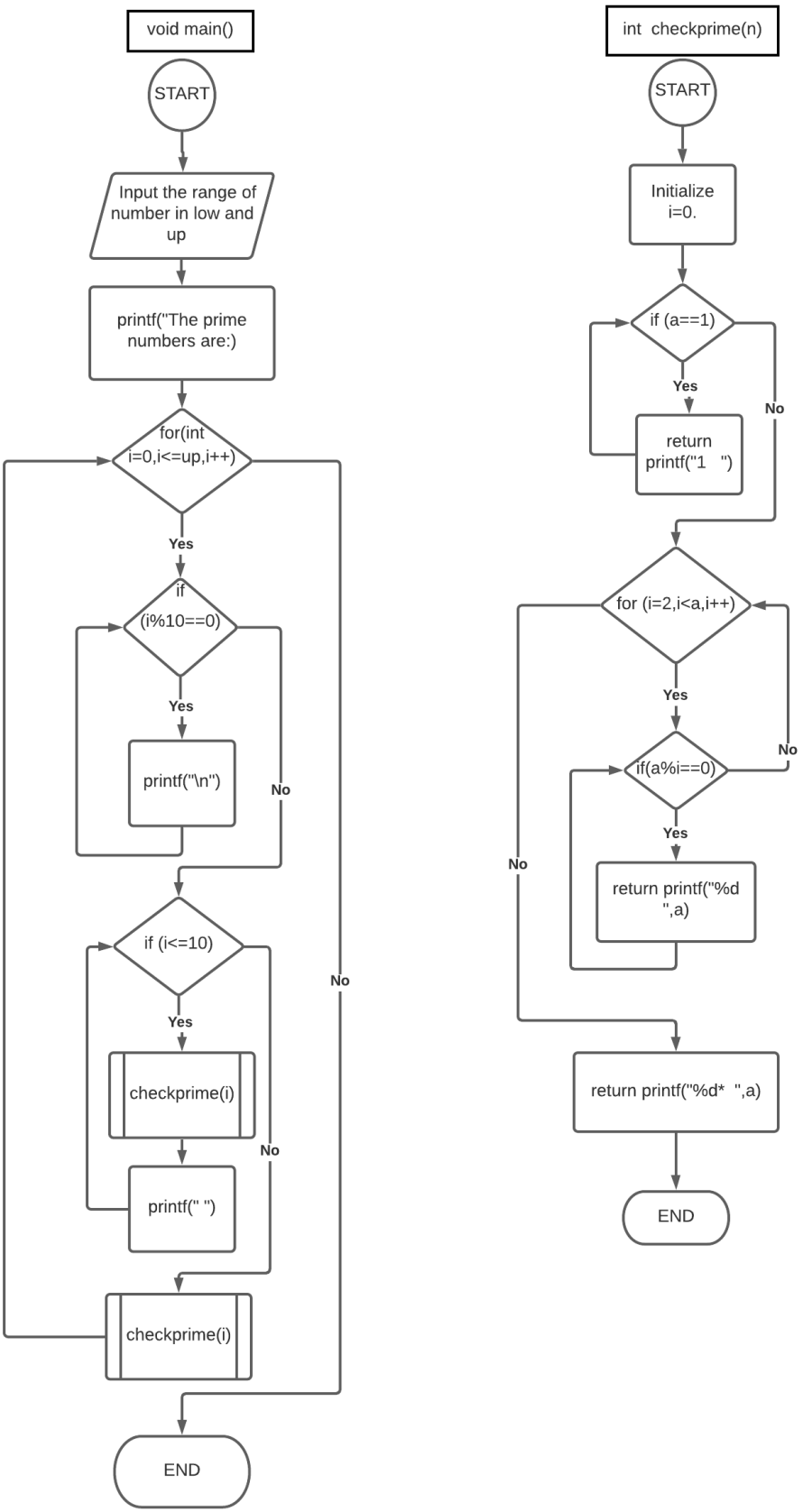
Name	**PRATIK PUJARI**
UID no.	2020300054
Experiment No.	3

AIM:	Apply the concept of functions to incorporate modularity.
Program 1	
PROBLEM STATEMENT :	**1.** Write a function which takes a range as input. Print all the numbers in the range with '*' in front of prime numbers only. All primes are starred
PROGRAM:	ALGORITHM: **main()** STEP 1: START. STEP 2: Take the range of number to be checked as input from the user and store it in variable "low","up". STEP 3: Initialize the loop counter to "i" to value of low. STEP 4: For i equal to the value of low and less than equal to up, Repeat the Steps 4.1 ,4.2and 4.3 or else if the condition is false go to step 5. STEP 4.1: If i%10 equal to zero ,do the escape sequence /n,or else go to Step 4.2. STEP 4.2: If the value of i is less than equal to 10 then call the user defined function checkprime(i) and printf(" "),else go to step 4.3. STEP 4.3: Call the user defined function check prime(i). STEP 5: END **checkprime(int n)** STEP 1:START. STEP 2 : Initialize the loop counter "i". STEP 3: If the value of n is equal to 1, the function should return printf ("1 "), else go to Step 4. STEP 4: For i equal to 2 and less than n, Repeat the steps 4.1and 4.2 or else if the condition fails go to Step 5. STEP 4.1: If a%i==0 , return the value of a or else go to Step 4.2. STEP 4.2: Increment the loop counter "i" by one. STEP 5: Return the value of n. STEP 6: END


void main()
START
Input the range of number in low and up
printf("The prime numbers are:)
for(int i=0,i<=up,i++)
Yes
if (i%10==0)
Yes
printf("\n")
No
if (i<=10)
Yes
checkprime(i)
printf(" ")
No
No
checkprime(i)
END
int checkprime(n)
START
Initialize i=0.
if (a==1)
Yes
return printf("1 ")
No
for (i=2,i<a,i++)
Yes
if(a%i==0)
Yes
return printf("%d ",a)
No
No
return printf("%d* ",a)
END

PROGRAM:

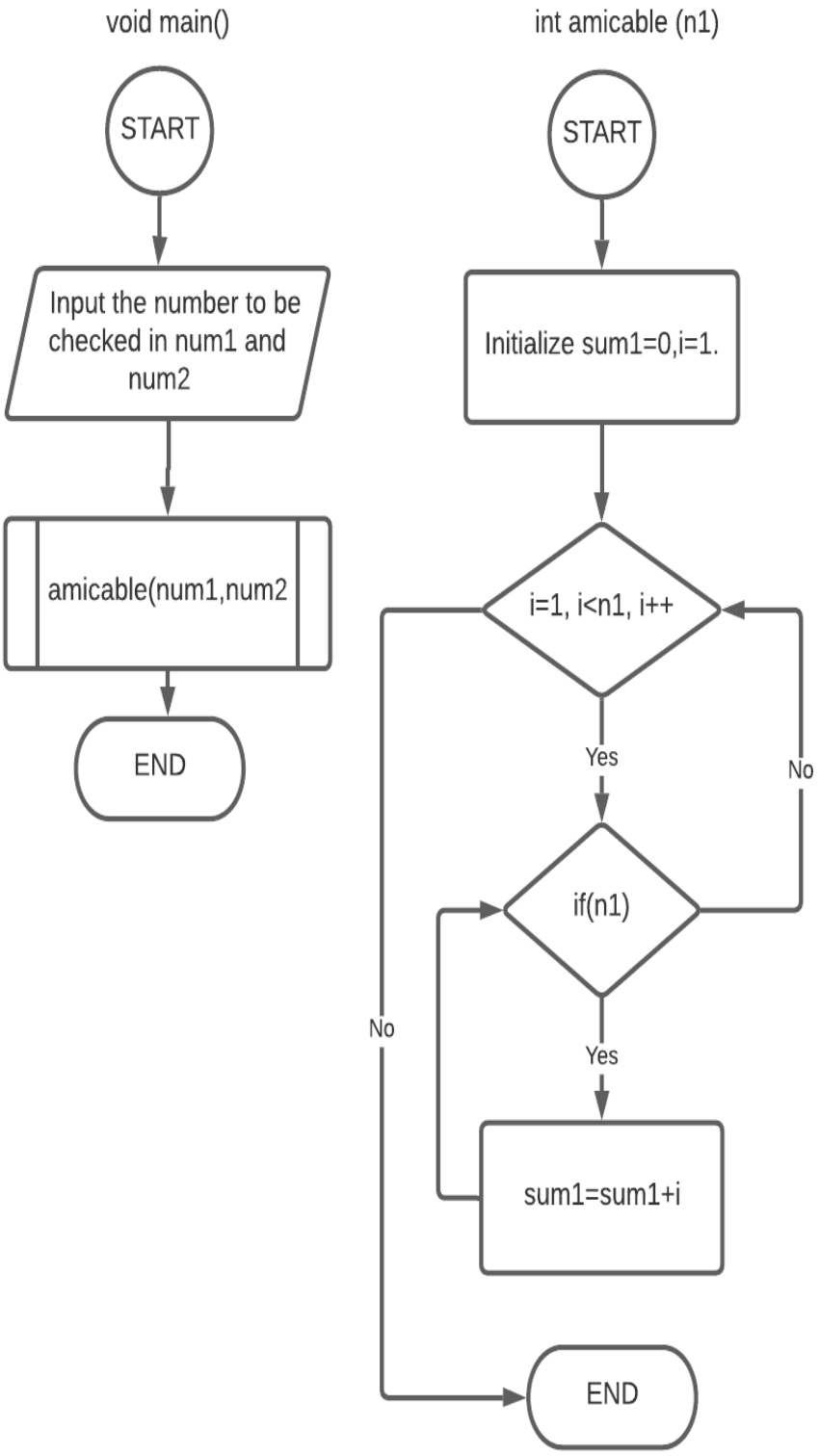
```
#include<stdio.h>
int checkprime(int);
void main()
{
  int low,up;
  printf("Enter the range of the  number for which prime numbers to be
printed:");
  scanf("%d %d",&low,&up);
  printf("The prime numbers are:\n");
  for(int i=low;i<=up;i++)
  {
     if (i%10==1)
     {
        printf("\n");
     }
     if (i<=10)
     {
        checkprime(i);
        printf(" ");
     }
     else
     {
         checkprime(i);
     }
  }
}
int checkprime(int a)
{
   int i;
   if(a == 1)
   {
      return printf("1   ");
   }

   for(i = 2; i < a; i++)
   {
      if(a % i == 0)
      {
        return printf("%d   ",a);
      }
   }

   return printf("%d*  ",a);
}
```

RESULT: All the Numbers in the range were printed with prime numbers starred.	
INPUT:	1 100
OUTPUT:	Enter the range of the number for which prime numbers to be printed:1 100 The prime numbers are: 1 2* 3* 4 5* 6 7* 8 9 10 11* 12 13* 14 15 16 17* 18 19* 20 21 22 23* 24 25 26 27 28 29* 30 31* 32 33 34 35 36 37* 38 39 40 41* 42 43* 44 45 46 47* 48 49 50 51 52 53* 54 55 56 57 58 59* 60 61* 62 63 64 65 66 67* 68 69 70 71* 72 73* 74 75 76 77 78 79* 80 81 82 83* 84 85 86 87 88 89* 90 91 92 93 94 95 96 97* 98 99 100

Program 2	
PROBLEM STATEMENT:	Write a function which takes as parameters two positive integers and returns TRUE if the numbers are amicable and FALSE otherwise. A pair of numbers is said to be amicable if the sum of divisors of each of the numbers (excluding the no. itself) is equal to the other number. Ex. 1184 and 1210 are amicable.
PROBLEM:	ALGORITHM: **main()** STEP 1:START. STEP 2:Initialize the variable num1 and num2. STEP 3:Take the two numbers to be checked and store it in variable in num1 and num2. STEP 3: Call the user defined function and pass the parameter num1 and num2. STEP 4: END **amicable(int n1,int n2)** STEP 1:START. STEP 2:Initialize the variables sum1 to zero and loop counter "i" to 1. STEP 3:For i equal to 1 and less than n1,Repeat the Steps 3.1 and 3.2 or else if the condition fails go to step 4. STEP 3.1: If n1%i is equal to 0 do sum=sum+i, else go to Step 3.2 . STEP 3.2: Increment the loop counter "i" by one . STEP 4: If sum is equal to n2 then return printf "TRUE".or else go to Step 5. STEP 5: Return Printf "FALSE". STEP 6: END.

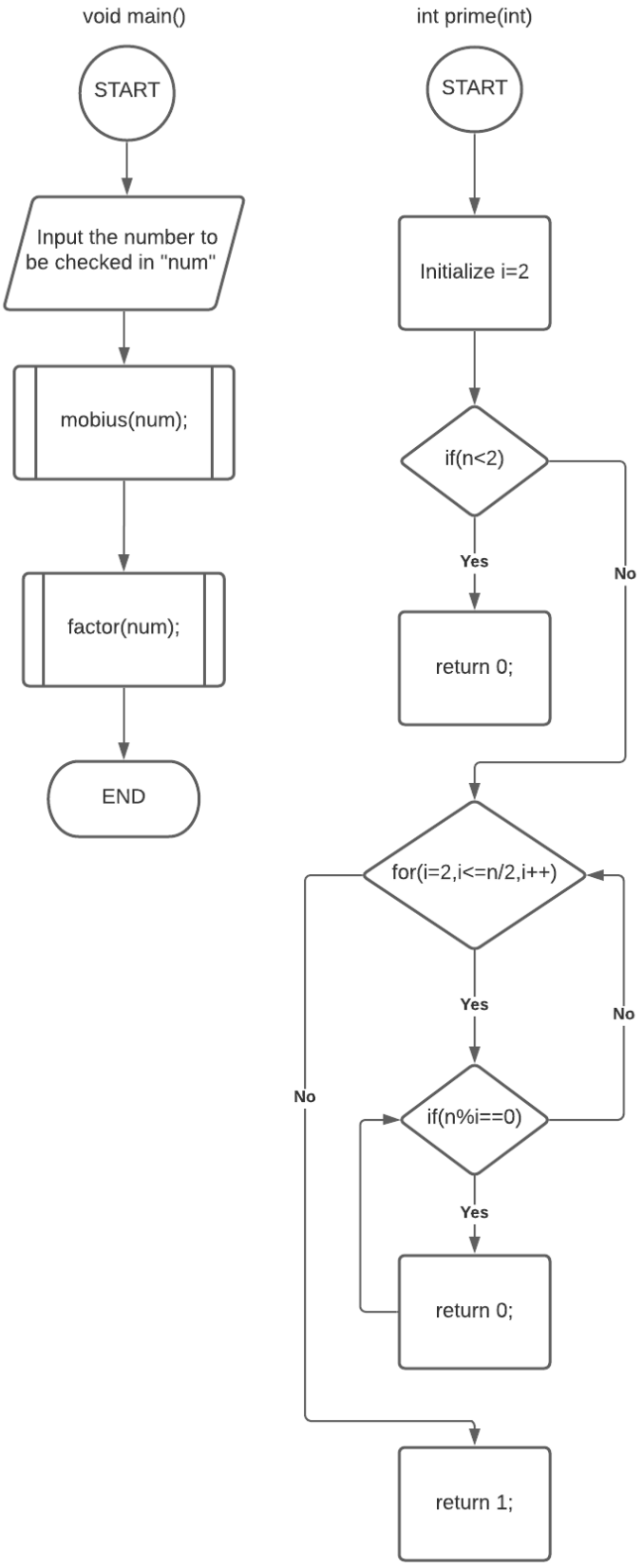

void main()
START
Input the number to be checked in num1 and num2
amicable(num1,num2
END
int amicable (n1)
START
Initialize sum1=0,i=1.
i=1, i<n1, i++
Yes
No
if(n1)
Yes
No
sum1=sum1+i
END

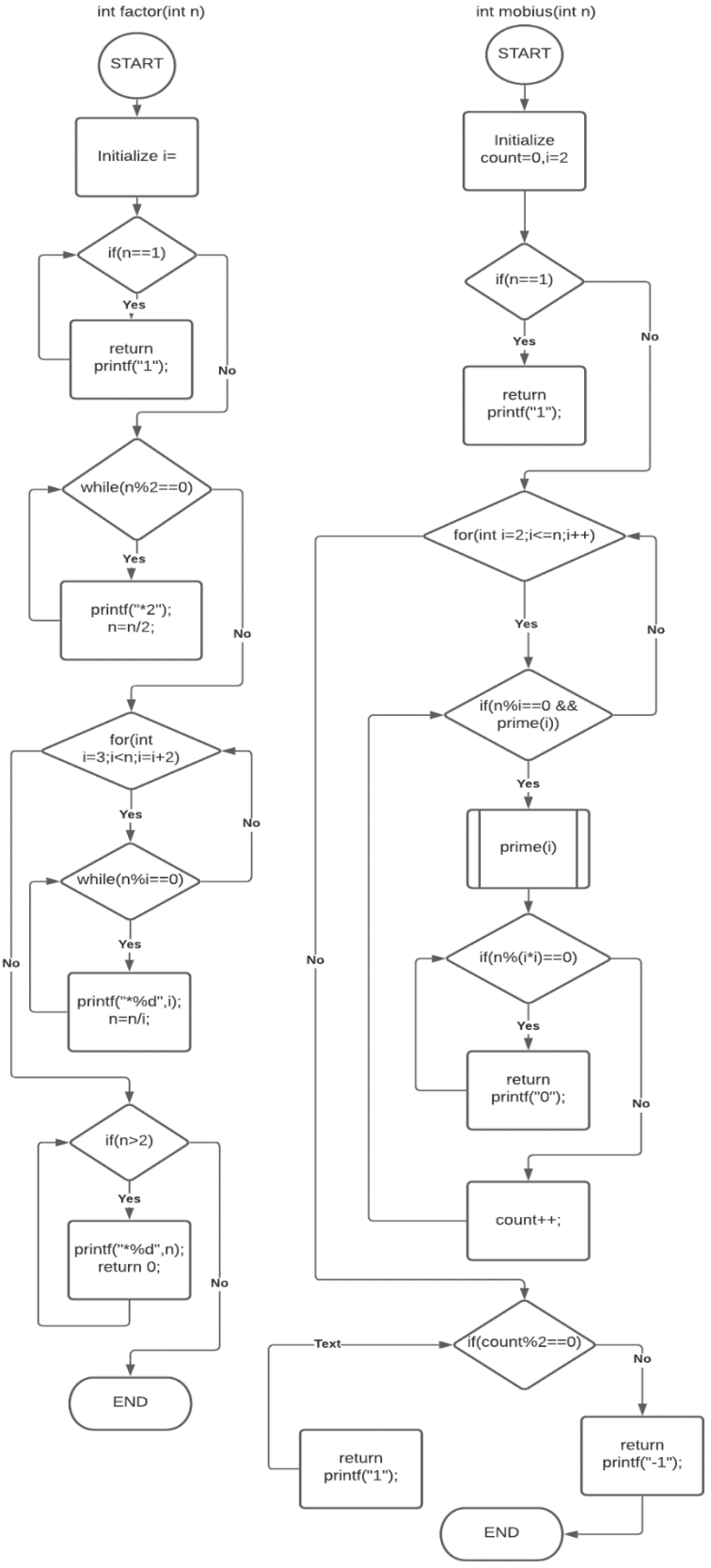
	PROGRAM:
	#include<stdio.h>
	int amicable(int,int);
	void main()
	{
	  int num1,num2;
	  printf("Enter the numbers to be checked:");
	  scanf("%d %d",&num1,&num2);
	  amicable(num1,num2);
	}
	int amicable(int n1,int n2)
	{
	  int sum1,sum2;
	  sum1=0;
	  sum2=0;
	 for(int i=1;i<n1;i++)
	  {
	   if(n1%i==0)
	  {
	  sum1=sum1+i;
	  }
	  }
	  if (sum1==n2)
	  {
	  return printf("TRUE");
	  }
	  else
	  {
	  return printf("FALSE");
	  }
	}

RESULT :If the numbers are amicable the program prints TRUE or else FALSE.	
INPUT:	1184 1210
OUTPUT:	Enter the numbers to be checked:1184 1210 TRUE

PROGRAM 3	
PROBLEM STATEMENT:	The Mobius function M (N) is defined as M (N) = 1 if N=1 = 0 if any prime factor is contained in N more than once = (-1)^p if N is the product of p different prime factors
PROBLEM:	ALGORITHM: **void main()** STEP 1: START. STEP 2: Take the number as input and store it in variable "num". STEP 3: Call the user defined function mobius(num) and factor(num). STEP 4: END **int prime(int n)** STEP 1:START**.** STEP 2:If n is less than 2 , function returns the value 0,or else it goes to step 3. STEP 3:Initialize the loop counter "i" to 2. STEP 4: For i equal to 2 and less than n/2,Repeat the steps 4.1 and 4.2 or else if the condition fails go to step 5. STEP 4.1: If n%i is equal to zero then the function returns value 0 or else go to step 4.2. STEP 4.2: Increment the loop counter "i" by one . STEP 5: The function returns the value 1. STEP 6: END

	int mobius(int n) STEP 1: START. STEP 2: Initialize the variable count to zero and loop counter “i” to 2. STEP 3: If n is equal to 1,then function returns printf(“1”) or else it goes to step 4. STEP 4: For i equal to 2 and less than equal to n Repeat 4.1, 4.2 or else if the condition fails go to Step 5. STEP 4.1: If Logical And between n%i equal to zero and prime(i) is true then go to steps 4.1.1,4.12 and 4.1.3 or else go to step 4.2. STEP 4.1.1: If n%i*i is equal to zero return printf(“0”) else go to step 4.1.2. STEP 4.1.2: Do count ++. STEP 4.2: Increment the loop counter by one. STEP 5: If count%2 equal to zero then return printf(“1”),else go to step 6. STEP 6: Function returns printf(“-1”). STEP 7: END. **int factor(int n**) STEP 1: START. STEP 2: If n equal to 1 then return printf(“1”)or else go to step 3. STEP 3: While n%2 equal to zero , Repeat the steps 3.1 and 3.2 or else if the condition go to step 3. STEP 3.1: Do Printf(“*2”) . STEP 3.2: Do n=n/2. STEP 4: Initialize the loop counter “i” to 3. STEP 5:For i equal to 3 and less than n , Repeat the steps 5.1 and 5.2 or else if the condition fails go to step 6. STEP 5.1: While n%i equal to zero do printf (“%d”,i) and n=n/i.or else if the condition fails go to Step 5.2. STEP 5.2: Increment the loop counter “i”. STEP 6: If n greater than 2 then printf (“%d”,n)or else go to Step 7. STEP 7: END.


void main()
START
Input the number to be checked in "num"
mobius(num);
factor(num);
END
int prime(int)
START
Initialize i=2
if(n<2)
Yes
No
return 0;
for(i=2,i<=n/2,i++)
Yes
No
No
if(n%i==0)
Yes
return 0;
return 1;


int factor(int n)
START
Initialize i=
if(n==1)
Yes
return printf("1");
No
while(n%2==0)
Yes
printf("*2");
n=n/2;
No
for(int i=3;i<n;i=i+2)
Yes
No
while(n%i==0)
Yes
No
printf("*%d",i);
n=n/i;
No
if(n>2)
Yes
printf("*%d",n);
return 0;
No
END
int mobius(int n)
START
Initialize count=0,i=2
if(n==1)
Yes
No
return printf("1");
for(int i=2;i<=n;i++)
Yes
No
if(n%i==0 && prime(i))
Yes
prime(i)
if(n%(i*i)==0)
Yes
No
return printf("0");
No
count++;
No
if(count%2==0)
Text
No
return printf("1");
return printf("-1");
END

PROGRAM:

```
#include<stdio.h>
int prime(int);
int factor(int);
int mobius(int);
void main()
{
  int num;
  printf("Enter the number to check if it is Mobius:");
  scanf("%d*",&num);
  printf("M(%d)=(",num);
  mobius(num);
  printf(") [%d=",num);
  factor(num);
  printf("]");
}
int prime(int n)
{
          if(n<2)
    return 0;
          for(int i=2;i<n/2;i++)
          {
        if(n%i==0)
                    return 0;
          }
          return 1;
}
int mobius(int n)
{
          int count=0;
          if(n==1)
          return printf("1");
          for(int i=2;i<=n;i++)
          {
                    if(n%i==0 && prime(i))
                    {
                              if(n%(i*i)==0)
                              {
                                        return printf("0");
                              }
                              else
                              count++;
                    }
          }
          if(count%2==0)
    return printf("1");
          else
          return printf("-1");}
```

```
int factor(int n)
{
        if(n==1)
        {
                return printf("1");
        }
        while(n%2==0)
        {
                printf("*2");
                n=n/2;
        }
        for(int i=3;i<n;i=i+2)
        {
                while(n%i==0)
                {
                        printf("*%d",i);
                        n=n/i;
                }
        }
        if(n>2)
        printf("*%d",n);
        return 0;
}
```

RESULT: The mobius value of the number was printed according to the case.

INPUT:	69
OUTPUT:	Enter the number to check if it is Mobius:69 M(69)=(1) [69=*3*23]
CONCLUSION:	Successfully applied the concept of functions to incorporate modularity.

\