AIM: Apply the concept of functions to incorporate modularity

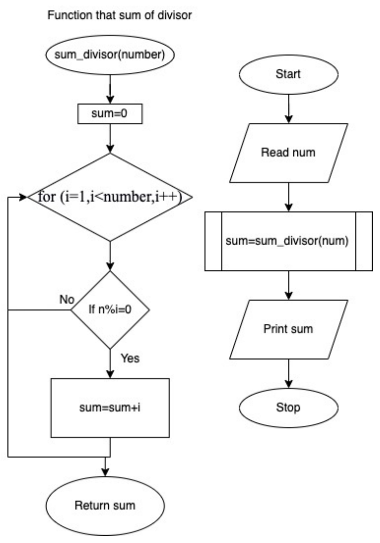
Program 1

PROBLEM STATEMENT : Write a function to find the sum of the proper divisors of a given number ‘n’. The proper divisors of a number ‘n’ are the numbers less than n that divide it; they do not include n itself. e.g. n=12 sum =1+2+3+4+6=16

ALGORITHM:

```
Function to sum the divisor of a number
Step 1. Take a number from parameter of function
Step 2. sum=0
Step 3. for i=1, i<number ,i++:
        Step 1: if n%i=0:
                    sum=sum+i
                else:
                    continue
                end if
        end for loop
Step 4. Return sum
Main Function
Step 1. Read num
Step 2. Use Function sum_divisor(num) and store it in sum
Step 3. Print sum
Step 4. Stop
```

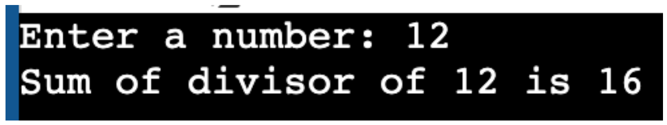
FLOWCHART:



PROGRAM:

```c
#include <stdio.h>
int sum_divisor(int n);
int main()
{
    int num;
    printf("Enter a number: ");
    scanf("%d",&num);//taking number from user
    int sum=sum_divisor(num);
    printf("Sum of divisor of %d is %d",num,sum);
    return 0;
}
// used this fuction to find Sum of divisor
int sum_divisor(int n){
    int sum=0;
    for (int i=1; i<n;i++){
        if (n%i==0){
            sum+=i;//finding the sum of divisor
        }
    }
    return sum;
}
```

RESULT:



```
Enter a number: 12
Sum of divisor of 12 is 16
```

Program 2

PROBLEM STATEMENT :

Write a function which takes a range as input. Print all the numbers in the range with '*'
in front of prime numbers only.
Example:
Print a table as follows
1 2* 3* 4 5* ... 10
11* 12 13* 14 15 ... 20
upto 100. All primes are starred.

ALGORITHM:

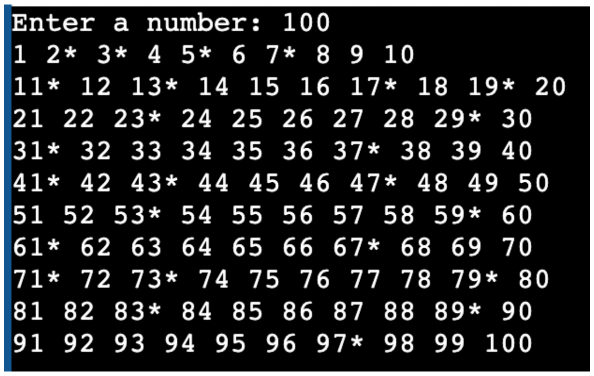
```
Function to check a number is prime or not (is_prime(number))
Step 1: Take a number from parameter of function
Step 2: if number =1:
            return 0
Step 3: for i=2, i<number, i++:
            if number%i=0:
                return 0
      end for loop
```

```
Step 4: then return 1 if all the condition above is false
Function to print the desire result(num_print())
Step 1: Read a number from user
Step 2:  for i =1, i<number, i++:
            if is_prime(i)=1:
                    print “i* ”
            else:
                    print “i ”
            end if
Main function
Step 1: Call the num_print() function
Step 2: Stop
```

PROGRAM:

```c
#include <stdio.h>
//Declaration of Fuction
int is_prime(int num);
void num_print();
int main()
{
   num_print();
   return 0;
}
//Check whether a number is prime or not
int is_prime(int num){
   if (num==1)
      return 0;
   for (int i=2; i<num;i++){
      if (num%i==0)
         return 0;
   }
   return 1;
}
//Print the desire result
void num_print(){
   int num;
   printf("Enter a number: ");
   scanf("%d",&num);
   for (int i =1;i<num;i++){
      if (is_prime(i)==1)
         printf("%d* ",i);
      else
         printf("%d ",i);
      if (i%10==0)
         printf("\n");
   }
}
```

RESULT:



```
Enter a number: 100
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 3

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.

ALGORITHM:

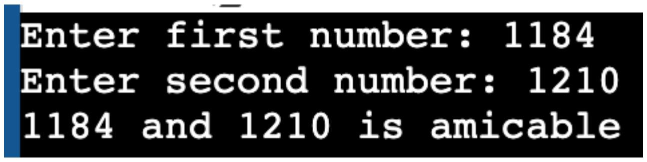
```
Function to sum the divisor of a number (sum_divisor(number))
Step 1. Take a number from parameter of function
Step 2. sum=0
Step 3. for i=1, i<number ,i++:
            Step 2:  if n%i=0:
                         sum=sum+i
                     else:
                         continue
                     end if
            end for loop
Step 4. Return sum
Function to find whether a pair of number is amicable or
not(is_amicable(num1,num2)
Step 1.  if sum_divisor(num1)==num2 && sum_divisor(num2)==num1:
                  return1
Step 2. return 0
Main Function
Step 1. Read two number num1 and num2
Step 2. if is_amicable(num1,num2)=1:
                print "num1 and num2 is amicable "
        else:
                print "num1 and num2 is not amicable "
        end if
Step 3. Stop
```

PROGRAM:

```
#include <stdio.h>
int sum_divisor(int num);
int is_amicable(int num1, int num2);
```

```c
int main()
{
    int num1, num2;
    //Taking numbers from user
    printf("Enter first number: ");
    scanf("%d",&num1);
    printf("Enter second number: ");
    scanf("%d",&num2);
    //Checking if two number are amicable
    if (is_amicable(num1,num2))
        printf("%d and %d is amicable",num1,num2);
    else
        printf("%d and %d is not amicable",num1,num2);
    return 0;
}
//Fuction to find sum of divisor
int sum_divisor(int num){
    int sum=0;
    for (int i=1; i<num;i++){
        if (num%i==0){
            sum+=i;
        }
    }
    return sum;
}
//Fuction to Check whether two number are amicable
int is_amicable(int num1, int num2){
    if (sum_divisor(num1)==num2 && sum_divisor(num2)==num1)
        return 1;
    return 0;
}
```

RESULT:



```
Enter first number: 1184
Enter second number: 1210
1184 and 1210 is amicable
```

Program 4

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
Thus, for example
M (78) = -1 [78 = 2 * 3 * 13]
M (34) = 1 [34 = 2 * 17]
M (45) = 0 [45 = 3 * 3 * 5]
Write a function MOBIUS as specified above.

ALGORITHM:

```
 Function to check a number is prime or not (is_prime(number))
Step 1. Take a number from parameter of function
Step 2. if number =1:
            return 0
Step 3. for i=2, i<number, i++:
           if number%i=0:
                    return 0
        end for loop
Step 4. then return 1 if all the condition above is false
Fuction to check a number is Mobius or not (mobius(num))
Step 1. Read num from parameter
Step 2. count=0
Step 3. for i=2, i<num, i++:
            if num%i==0 && is_prime(i):
              if (num%(i*i)==0):
                   return 0
              end if
              else:
                   count=count+1
            end if
      end for loop
Step 4. if count%2=0:
            return 1
        else:
            return  -1
        end if
Step 5. Stop
```

PROGRAM:

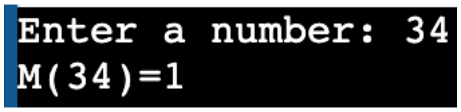
```
#include <stdio.h>

//Function delcaration
int mobius(int num);
int is_prime(int num);
int main()
{
   int a,num;
```

```
    //Taking a number from User
    printf("Enter a number: ");
    scanf("%d",&num);
    a=mobius(num);
    printf("M(%d)= %d",num,a);
    return 0;
}
//Check a number is prime or not
int is_prime(int num){
    if (num<2)
        return 0;
    for (int i = 2; i < num; i++)
        if (num % i == 0)
            return 0;
    return 1;
}
//Function to check a number is Mobius or not
int mobius(int num){
    int count=0;
    for(int i=2;i<num;i++){
        if (num%i==0 && is_prime(i)){
            if (num%(i*i)==0)//to return 0 if number is repeated
                return 0;
            else
                count++;
        }
    }
    return (count%2==0)?1:-1;
}
```

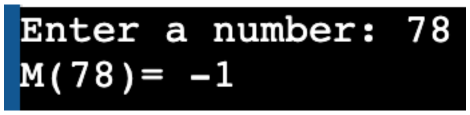
RESULT:

Case 1:



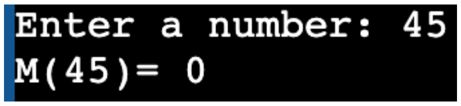
```
Enter a number: 34
M(34)=1
```

Case 2:



```
Enter a number: 78
M(78)= -1
```

Case 3:



```
Enter a number: 45
M(45)= 0
```

CONCLUSION:	We learned to apply the concept of functions to incorporate modularity