Name	**Hatim Yusuf Sawai**
UID no.	**2021300108**
Experiment No.	3

AIM:	Apply the concept of functions to incorporate modularity.
Program 1	
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
ALGORITHM:	1. START 2. Int Prime(int) 3. Main: 4. Input n 5. For i=1 p = prime(i) if p==0 Ouput i* Else Output i If i%10==0 Output \n 6. Repeat step 5 till i<=n 7. STOP

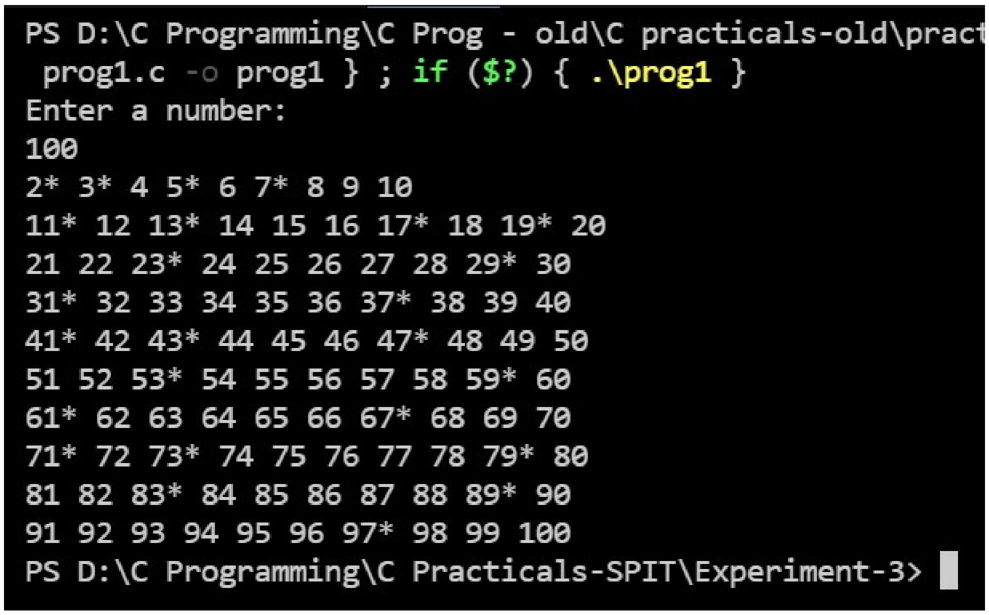
```
1. START
2. Prime(int n):
3. If n==1
   Return 1
   Else
   For i=2,i<=sqrt(n),i++
   If n%i==0
      Return 0
4. Return 0
5. STOP
```

PROGRAM:

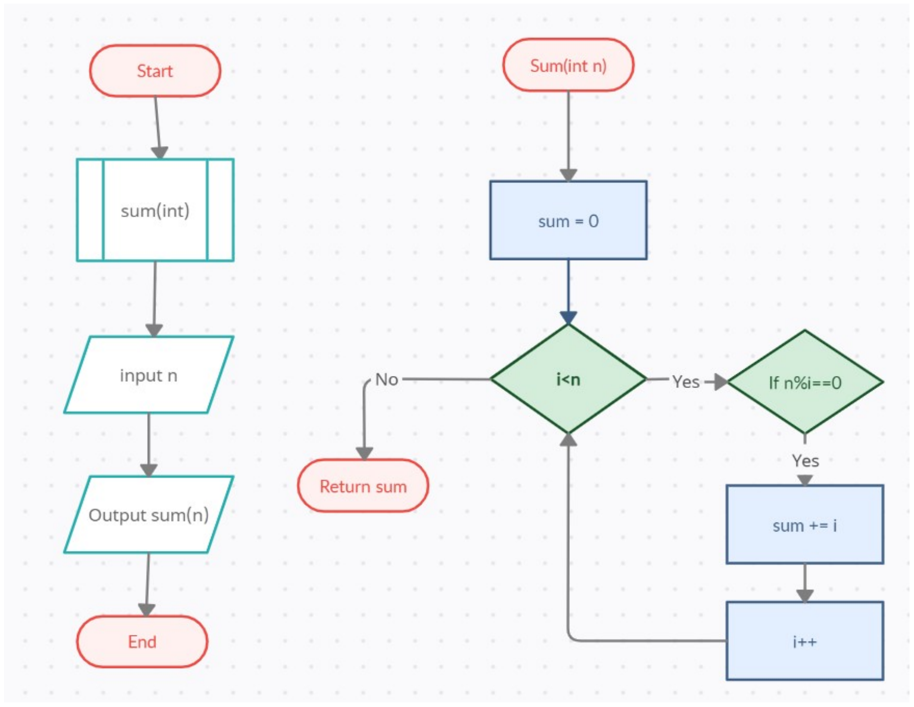
```c
#include <stdio.h>
#include <math.h>
int prime(int);
int main()
{
    int n, p,i;
    printf("Enter a number:\n");
    scanf("%d", &n);
    for(i=2;i<=n;i++)
    {
        p=prime(i);
        if(p==0)
            printf("%d* ",i);
        else
            printf("%d ",i);

        if(i%10==0)
            printf("\n");
    }
}
```

	```
	int prime(int n)
	{
	    int i;
	    if (n == 1)
	        return 1;
	    else if (n > 1)
	    {
	        for (i = 2; i <= sqrt(n); i++)
	        {
	            if (n % i == 0)
	                return 1;
	        }
	    }
	    return 0;
	}
	```

RESULT:



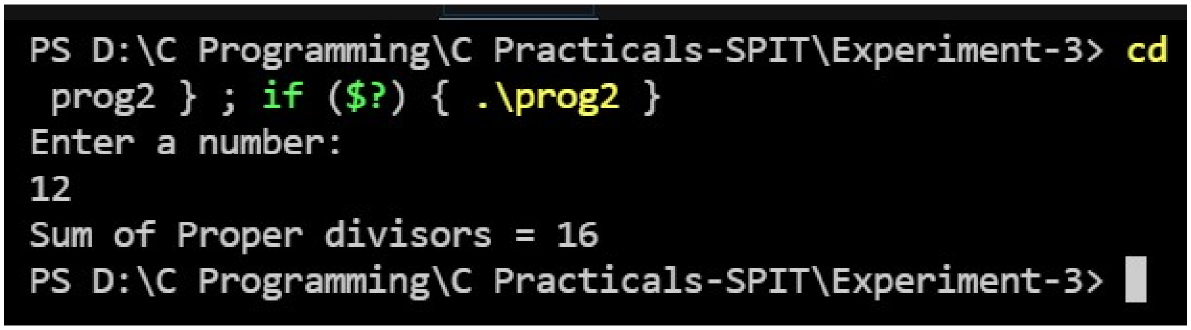
```
PS D:\C Programming\C Prog - old\C practicals-old\pract
 prog1.c -o prog1 } ; if ($?) { .\prog1 }
Enter a number:
100
2* 3* 4 5* 6 7* 8 9 10
11* 12 13* 14 15 16 17* 18 19* 20
21 22 23* 24 25 26 27 28 29* 30
31* 32 33 34 35 36 37* 38 39 40
41* 42 43* 44 45 46 47* 48 49 50
51 52 53* 54 55 56 57 58 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
PS D:\C Programming\C Practicals-SPIT\Experiment-3>
```

Program 2	
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 evenly. they do not include n itself. e.g. n=12 sum =1+2+3+4+6=16
ALGORITHM:	1. START (sum) 2. Sum=0 3. For i=1 4. If n%i==0 5. Sum +=i 6. Repeat steps 3-5 till i<n 7. Return sum 1. START (Main) 2. Input n 3. Output sum(n) 4. STOP
FLOWCHART:	 Start sum(int) input n Output sum(n) End Sum(int n) sum = 0 i<n No Return sum Yes If n%i==0 Yes sum += i i++

PROGRAM:

```
#include<stdio.h>
int sum(int);
int main()
{
   int n;
   printf("Enter a number:\n");
   scanf("%d",&n);
   printf("Sum of Proper divisors = %d",sum(n));
   return 0;
}
int sum(int n)
{
   int i,sum=0;
   for(i=1;i<n;i++)
   {
      if(n%i==0)
         sum+=i;
   }
   return sum;
}
```

RESULT:

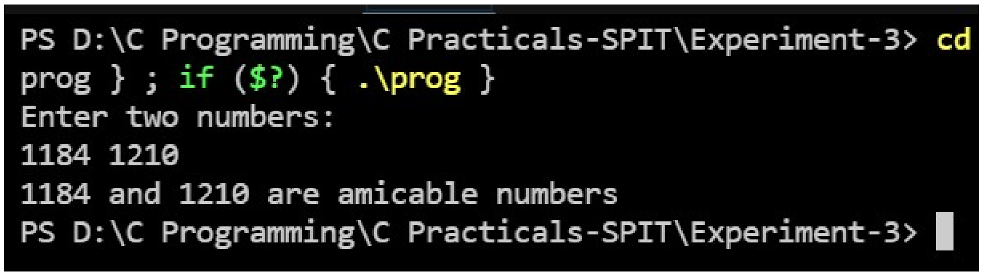


```
PS D:\C Programming\C Practicals-SPIT\Experiment-3> cd
 prog2 } ; if ($?) { .\prog2 }
Enter a number:
12
Sum of Proper divisors = 16
PS D:\C Programming\C Practicals-SPIT\Experiment-3>
```

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:	1. START(Sum) 2. Sum=0 3. For i=1 4. If n%i==0 5. Sum +=i 6. Repeat steps 3-5 till i<n 7. Return sum 1. START(Amicable) 2. If sum(a)==b and sum(b)==a Return 1 3. Return 0 1. START(Main) 2. Input a,b 3. If amicable(a,b) 4. Output Numbers are amicable 5. Else 6. Output Number are not amicable 7. STOP
FLOWCHART:	

PROGRAM:

```
#include<stdio.h>
int sum(int);
int amicable(int,int);
int main()
{
    int a,b;
    printf("Enter two numbers:\n");
    scanf("%d%d",&a,&b);
    if(amicable(a,b))
        printf("%d and %d are amicable numbers\n",a,b);
    else
        printf("%d and %d are not amicable numbers\n",a,b);
    return 0;
}
int amicable(int a,int b)
{
    if(sum(a)==b && sum(b)==a)
        return 1;
    return 0;
}
int sum(int n)
{
    int i, sum = 0;
    for (i = 1; i < n; i++)
    {
        if (n % i == 0)
            sum += i;
    }
    return sum;
}
```

RESULT: 	

```
PS D:\C Programming\C Practicals-SPIT\Experiment-3> cd
prog } ; if ($?) { .\prog }
Enter two numbers:
1184 1210
1184 and 1210 are amicable numbers
PS D:\C Programming\C Practicals-SPIT\Experiment-3>
```

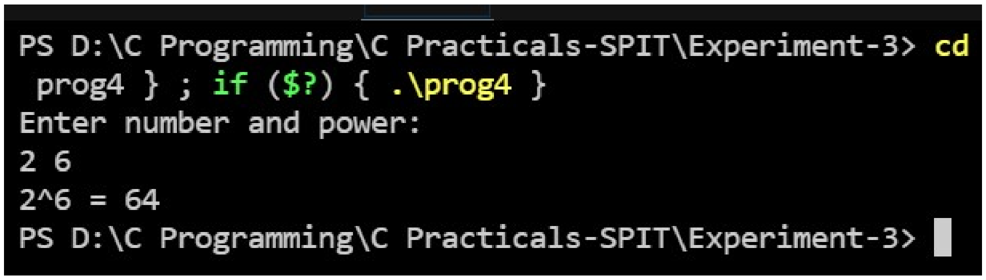
Program 4

PROBLEM STATEMENT:	Batch 3: A common method of evaluating powers is merely to perform repeated multiplications.A more efficient method of evaluating XN is: Initialize PRODUCT to 1, POWER to X and M to N. While M is non-zero repeat the following task: If M is odd then Replace PRODUCT by PRODUCT * POWER End If; Replace M by M/2; Replace POWER by POWER * POWER End task. The required result is then in POWER. Write a function which represents this procedure for calculating XN.
ALGORITHM:	1. START (power) 2. p=1 3. while: 4. if n%2==0 p = p*x 5. x = x*x 6. n = n/2 7. Repeat steps 3-6 till n>0 8. Return p 1. START (Main) 2. Input x,n 3. Output x^n = power(x,n) 4. STOP

PROGRAM:

```c
#include<stdio.h>
int power(int,int);
int main()
{
    int x,n;
    printf("Enter number and power:\n");
    scanf("%d%d",&x,&n);
    printf("%d^%d = %d",x,n,power(x,n));
    return 0;
}
int power(int x,int n)
{
    int p=1;
    while(n>0)
    {
        if (n%2==1)
            p=p*x;
        x=x*x;
        n=n/2;
    }
    return p;
}
```

RESULT:



```
PS D:\C Programming\C Practicals-SPIT\Experiment-3> cd
 prog4 } ; if ($?) { .\prog4 }
Enter number and power:
2 6
2^6 = 64
PS D:\C Programming\C Practicals-SPIT\Experiment-3>
```

CONCLUSION:

In this experiment, we saw how to declare, define and use functions to introduce modularity in our programs thereby reducing time complexity and length of code.