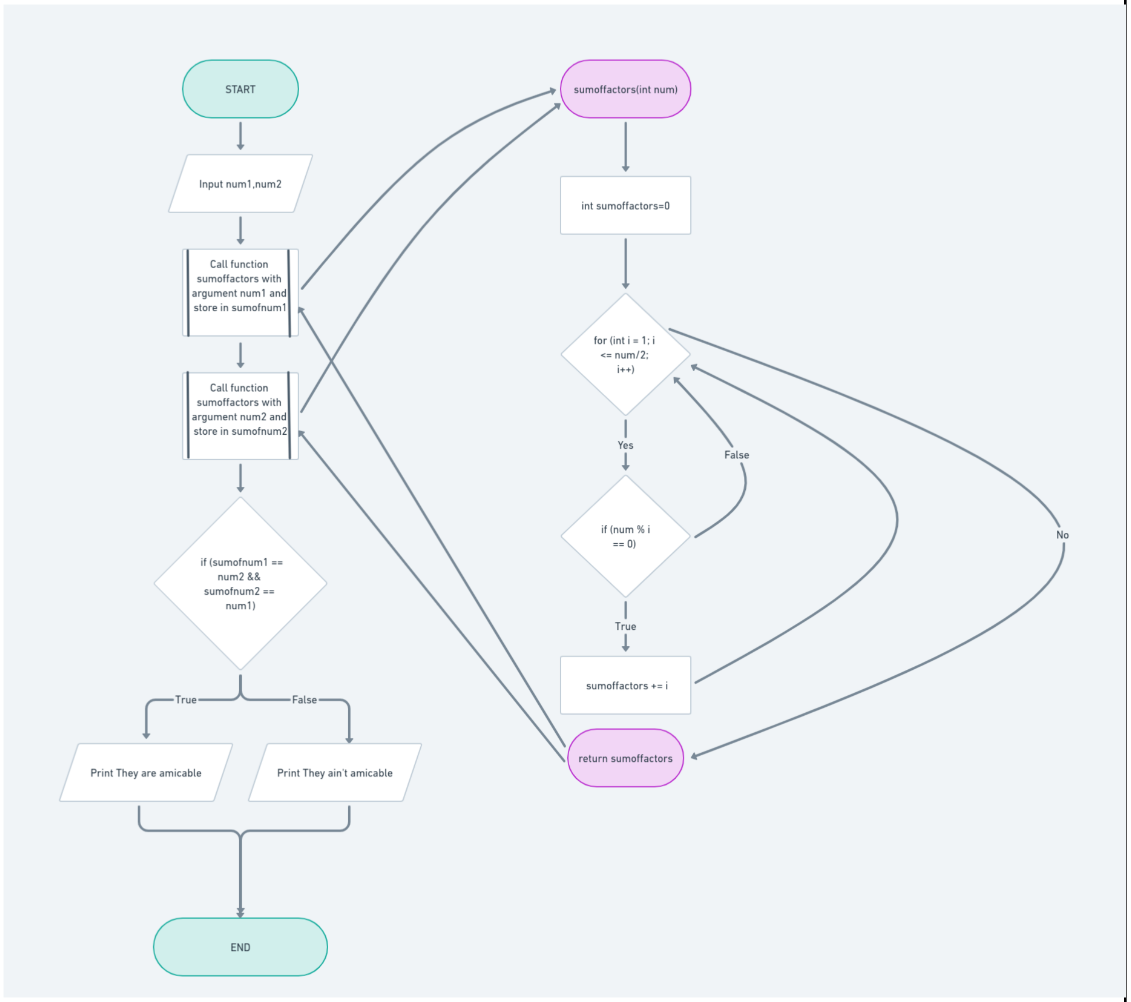
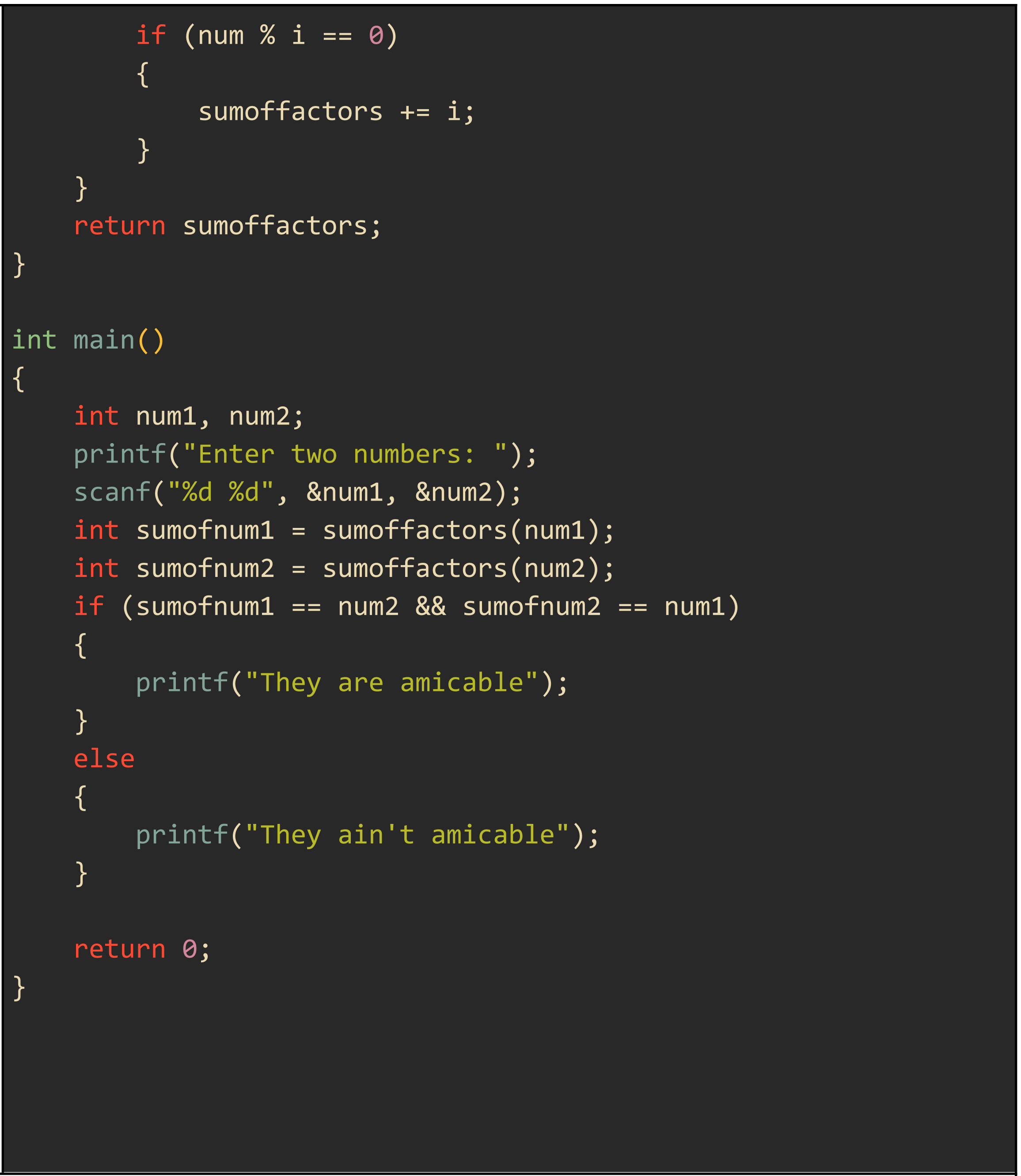
Name	Jhaveri Varun Nimitt
UID no.	2023800042
Experiment No.	3

AIM:	**Apply the concept of functions to incorporate modularity**
Program 1	
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 main function 2. Declare two integer variables num1 and num2 3. Print "Enter two numbers: " 4. Scan two integers into num1 and num2 5. Call sumoffactors function for num1 and store the result in sumofnum1 6. Call sumoffactors function for num2 and store the result in sumofnum2 7. Check if sumofnum1 is equal to num2 and sumofnum2 is equal to num1 a. If true, print "They are amicable" b. If false, print "They ain't amicable" 8. End main function (5 and 6) ● Define a function sumoffactors that takes an integer num ● Initialize a variable sumoffactors to 0 ● Start a loop from i=1 to num/2 and increment i in each iteration ● a. Check if num modulus i = 0 ● b. If it is, add i to sumoffactors ● End loop ● Return sumoffactors ● End function sumoffactors
FLOWCHART:	


START
Input num1,num2
Call function sumoffactors with argument num1 and store in sumofnum1
Call function sumoffactors with argument num2 and store in sumofnum2
if (sumofnum1 == num2 && sumofnum2 == num1)
True
False
Print They are amicable
Print They ain't amicable
END
sumoffactors(int num)
int sumoffactors=0
for (int i = 1; i <= num/2; i++)
Yes
False
No
if (num % i == 0)
True
sumoffactors += i
return sumoffactors

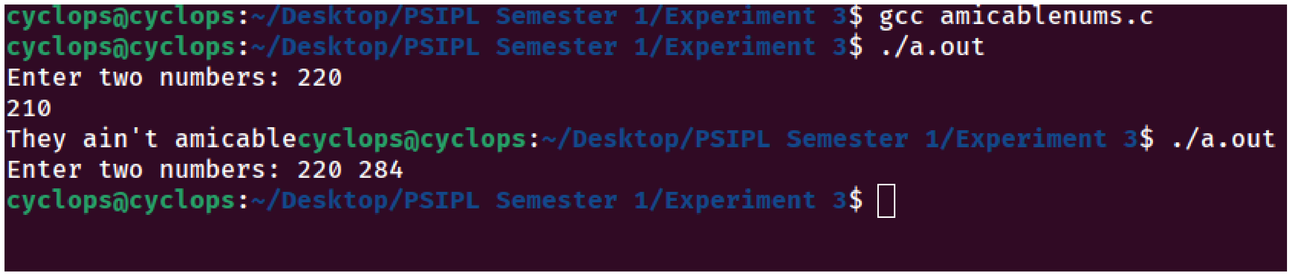
PROGRAM:

```
#include <stdio.h>
int sumoffactors(int num)
{
    int sumoffactors = 0;
    for (int i = 1; i <= num/2; i++)
    {
```



```c
        if (num % i == 0)
        {
            sumoffactors += i;
        }
    }
    return sumoffactors;
}

int main()
{
    int num1, num2;
    printf("Enter two numbers: ");
    scanf("%d %d", &num1, &num2);
    int sumofnum1 = sumoffactors(num1);
    int sumofnum2 = sumoffactors(num2);
    if (sumofnum1 == num2 && sumofnum2 == num1)
    {
        printf("They are amicable");
    }
    else
    {
        printf("They ain't amicable");
    }

    return 0;
}
```

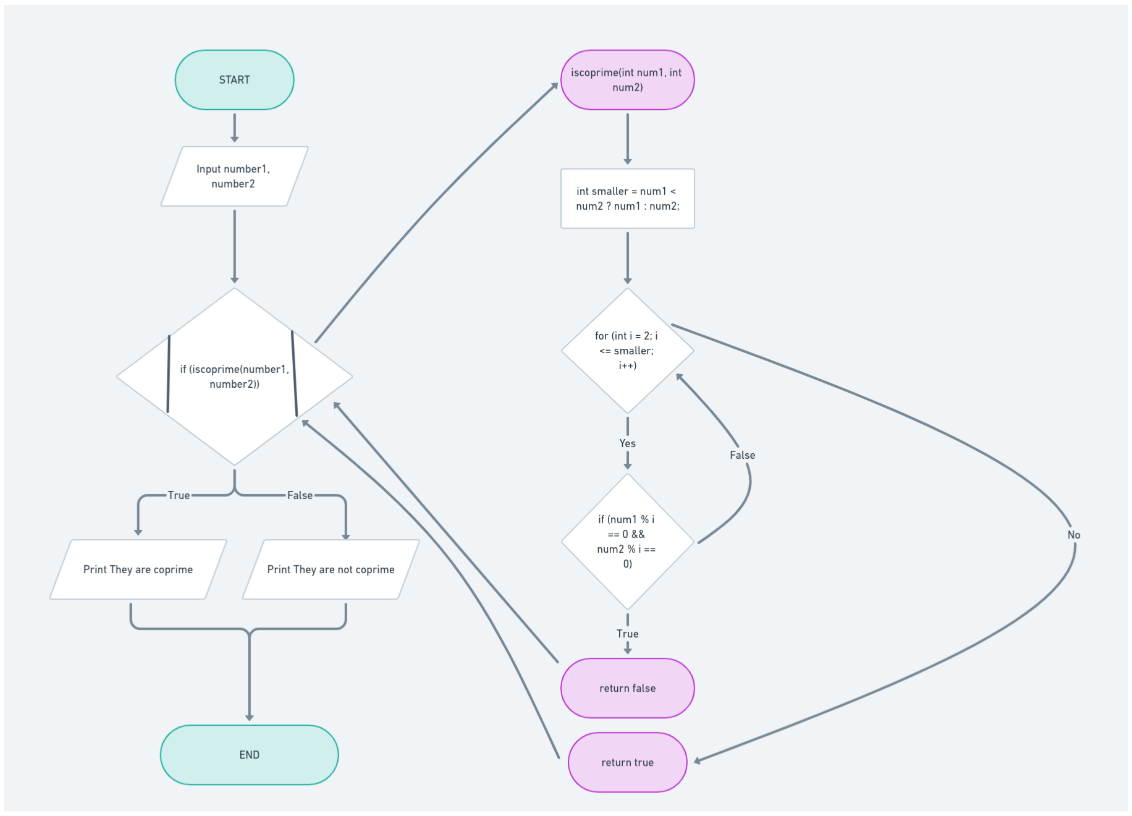
RESULT:



```
cyclops@cyclops:~/Desktop/PSIPL Semester 1/Experiment 3$ gcc amicablenums.c
cyclops@cyclops:~/Desktop/PSIPL Semester 1/Experiment 3$ ./a.out
Enter two numbers: 220
210
They ain't amicablecyclops@cyclops:~/Desktop/PSIPL Semester 1/Experiment 3$ ./a.out
Enter two numbers: 220 284
cyclops@cyclops:~/Desktop/PSIPL Semester 1/Experiment 3$ 
```

Program 2	
PROBLEM STATEMENT :	Write a function to find out whether given numbers are relatively prime (co-prime) or not. A number is relatively prime if the ‘1’ is the only common factor between the two numbers. For example: 9 and 8 are relatively prime. (9 =1x3x3 and 8=1x2x2x2).
ALGORITHM:	1. Start main function 2. Declare two integer variables number1 and number2 and initialize number2 to 0 3. Print Enter 1st number: 4. Scan an integer into number1 5. Print Enter 2nd number: 6. Scan an integer into number2 7. If iscoprime(number1, number2) returns true a. Print The numbers number1 and number2 are coprime. 8. Else a. Print The numbers number1 and number2 are not coprime. 9. End main function (7) ● Define a function iscoprime that takes two integer parameters num1 and num2 and returns a boolean ● Declare an integer variable smaller and initialize it to the smaller of num1 and num2 ● Start a loop from i=2 to smaller and increment i by 1 each iteration a. Check if both num1 and num2 are divisible by i and return no remainder i. If they are, return false ● End loop ● Return true ● End function iscoprime

FLOWCHART:

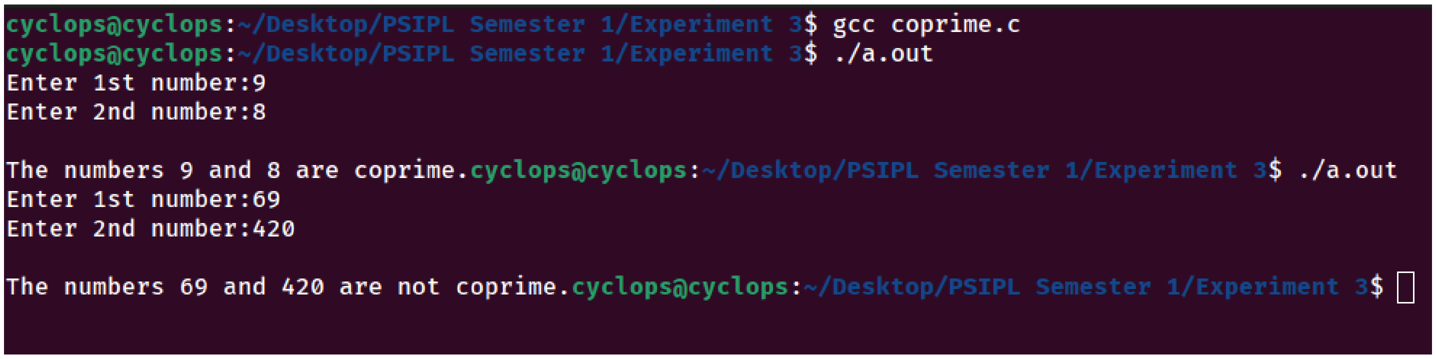

START
Input number1, number2
if (iscoprime(number1, number2))
True
False
Print They are coprime
Print They are not coprime
END
iscoprime(int num1, int num2)
int smaller = num1 < num2 ? num1 : num2;
for (int i = 2; i <= smaller; i++)
Yes
No
if (num1 % i == 0 && num2 % i == 0)
False
True
return false
return true

PROGRAM:

```c
#include <stdio.h>
#include <stdbool.h>

bool iscoprime(int num1, int num2);

int main()
{
      int number1, number2 = 0;
```

```
        printf("Enter 1st number:");
        scanf("%d", &number1);
        printf("Enter 2nd number:");
        scanf("%d", &number2);
        if (iscoprime(number1, number2))
        {
                printf("\nThe numbers %d and %d are coprime.",
number1, number2);
        }
        else
        {
                printf("\nThe numbers %d and %d are not coprime.",
number1, number2);
        }
}

bool iscoprime(int num1, int num2)
{
        int smaller = num1 < num2 ? num1 : num2;
        //dont need to run till bigger num coz any factor of the
two numbers cannot be greater than the smaller of the two
numbers
        for (int i = 2; i <= smaller; i++)
        {
                if (num1 % i == 0 && num2 % i == 0)
                {
                        return false;
                }
        }

        return true;
}
```

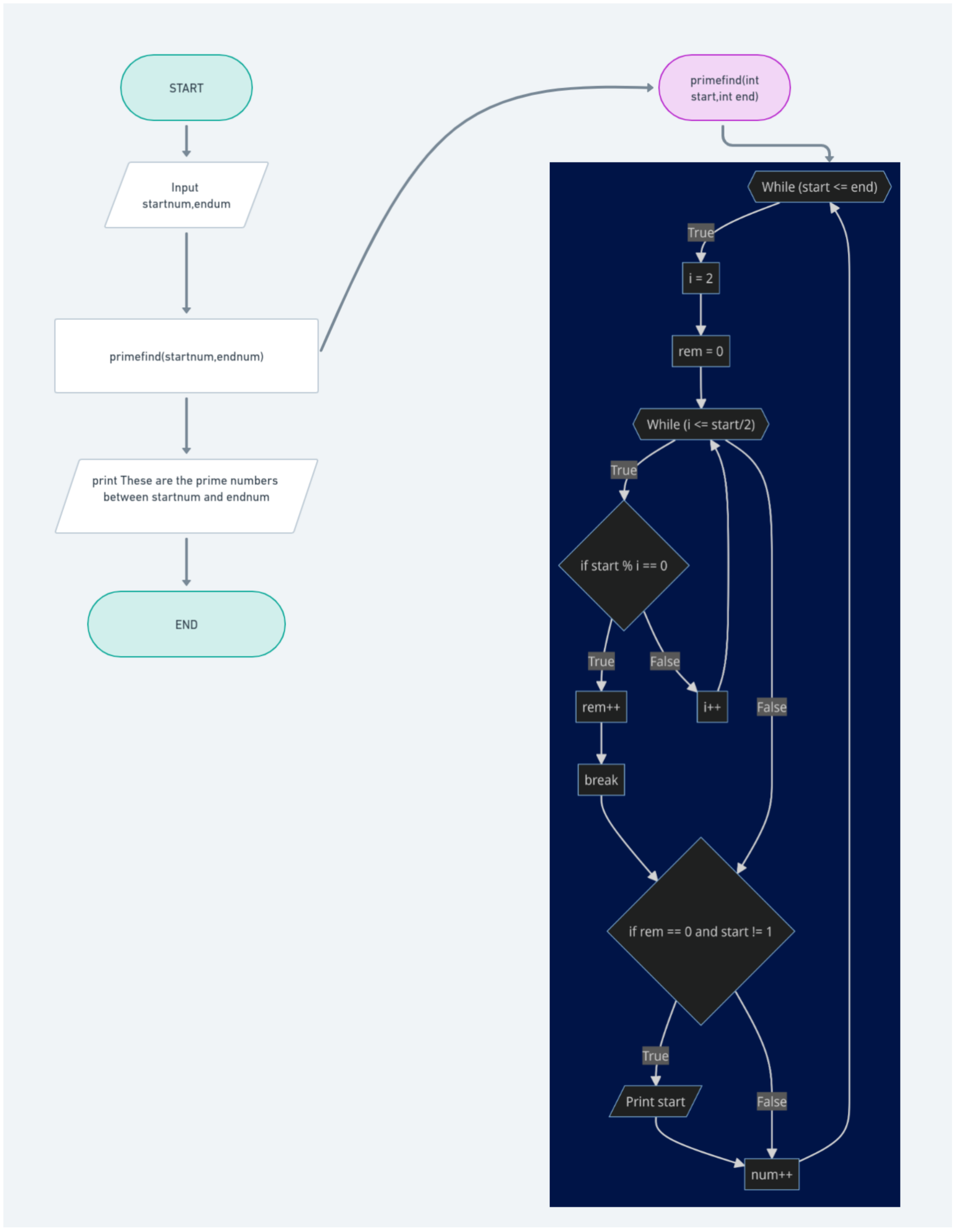
RESULT:



```
cyclops@cyclops:~/Desktop/PSIPL Semester 1/Experiment 3$ gcc coprime.c
cyclops@cyclops:~/Desktop/PSIPL Semester 1/Experiment 3$ ./a.out
Enter 1st number:9
Enter 2nd number:8

The numbers 9 and 8 are coprime.cyclops@cyclops:~/Desktop/PSIPL Semester 1/Experiment 3$ ./a.out
Enter 1st number:69
Enter 2nd number:420

The numbers 69 and 420 are not coprime.cyclops@cyclops:~/Desktop/PSIPL Semester 1/Experiment 3$
```

Program 3

PROBLEM STATEMENT:	Write a function which takes a range as input. Print all the prime numbers in the range.
ALGORITHM:	• Define a function primefind that takes two integer parameters start and end and returns void • Initialize integer variables rem and i to 0 • Start a while loop with the condition start <= end a. Set i to 2 b. Set rem to 0 c. Start a nested while loop with the condition i <= start/2 i. Check if start is divisible by i with remainder 0 ii. If it is, increment rem and break the loop d. Check if rem is 0 and start is not 1 i. If true, print start followed by a space e. Increment start by 1 • End while loop • End function primefind 1. Start main function 2. Declare two integer variables startnum and endnum and initialize them to 0 3. Print "Enter start number:" 4. Scan an integer into startnum 5. Print "Enter end number:" 6. Scan an integer into endnum 7. Call primefind function with arguments startnum and endnum 8. Print "These are the prime numbers between" 9. End main function

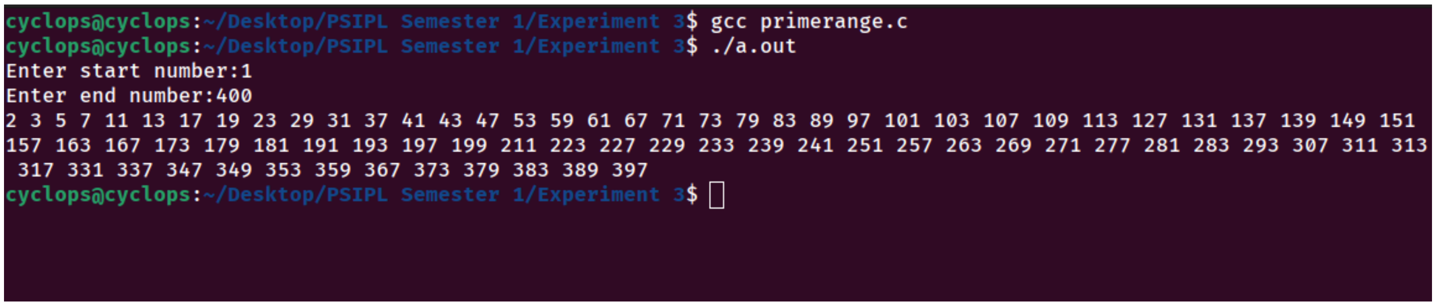
FLOWCHART:	

START
Input startnum,endum
primefind(startnum,endnum)
print These are the prime numbers between startnum and endnum
END
primefind(int start,int end)
While (start <= end)
True
i = 2
rem = 0
While (i <= start/2)
True
if start % i == 0
True
False
rem++
i++
False
break
if rem == 0 and start != 1
True
False
Print start
num++

PROGRAM:

```c
#include <stdio.h>



void primefind(int start,int end);
int main()
{
        int startnum=0;
        int endnum=0;
        printf("Enter start number:");
        scanf("%d", &startnum);
        printf("Enter end number:");
        scanf("%d", &endnum);
        primefind(startnum,endnum);
        printf("\nThese are the prime numbers between %d and 
%d.",startnum,endnum);

        return 0;
}



void primefind(int start,int end)
{
      int rem = 0;
      int i =0;
      while (start <= end)
      {
               i = 2;
               rem=0;
                while (i <= start/2)
                {
                        if (start % i == 0)
                        {
                                rem++;
                                break;
                        }
                        i++;
                }
                if (rem == 0 && start != 1)
                {
                        printf("%d ", start);
                }
                start++;
        }
}
```

RESULT:



```
cyclops@cyclops:~/Desktop/PSIPL Semester 1/Experiment 3$ gcc primerange.c
cyclops@cyclops:~/Desktop/PSIPL Semester 1/Experiment 3$ ./a.out
Enter start number:1
Enter end number:400
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97 101 103 107 109 113 127 131 137 139 149 151
157 163 167 173 179 181 191 193 197 199 211 223 227 229 233 239 241 251 257 263 269 271 277 281 283 293 307 311 313
 317 331 337 347 349 353 359 367 373 379 383 389 397
cyclops@cyclops:~/Desktop/PSIPL Semester 1/Experiment 3$ 
```

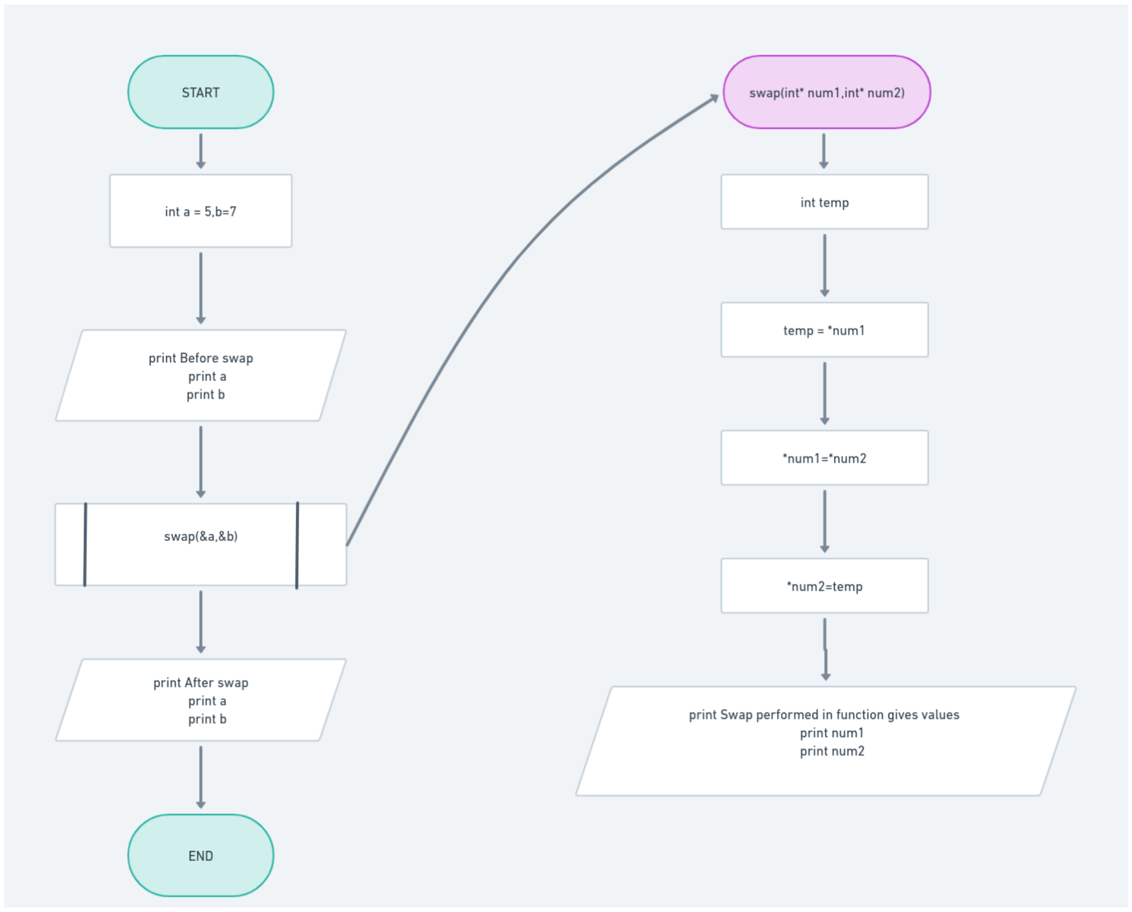
Program 4

PROBLEM STATEMENT: Write a function which swaps two integers using pointers as parameters.

ALGORITHM:

- Define a function named swap that takes two integer pointers, num1 and num2, and returns void
- Declare an integer temp
- Set temp to the value pointed by num1
- Set the value pointed by num1 to the value pointed by num2
- Set the value pointed by num2 to temp
- Print "Swap performed in function gives values : "
- Print the value pointed by num1
- Print the value pointed to by num2
- End function swap

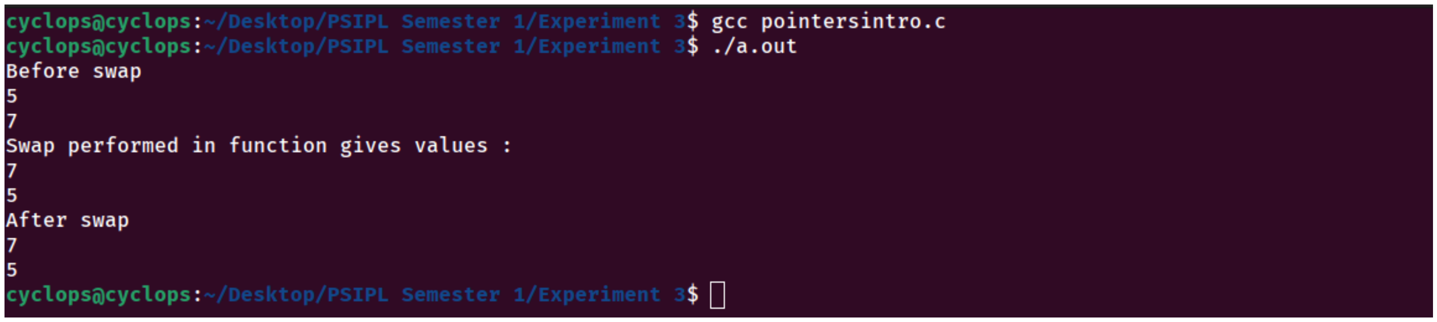
1. Start main function
2. Declare two integer variables, a and b, and initialize them to 5 and 7, respectively
3. Print "Before swap"

	4. Print the value of a 5. Print the value of b 6. Call the swap function with the addresses of a and b as arguments 7. Print "After swap" 8. Print the value of a 9. Print the value of b 10. End main function
FLOWCHART:	 START int a = 5,b=7 print Before swap print a print b swap(&a,&b) print After swap print a print b END swap(int* num1,int* num2) int temp temp = *num1 *num1=*num2 *num2=temp print Swap performed in function gives values print num1 print num2

PROGRAM:

```c
//call by reference

#include <stdio.h>
void swap(int* num1,int* num2);
int main()
{
	int a = 5,b=7;
	printf("Before swap\n");
	printf("%d \n",a);
	printf("%d \n",b);
	swap(&a,&b);
	printf("After swap\n");
	printf("%d \n",a);
	printf("%d \n",b);
	return 0;
}

void swap(int* num1,int* num2)
{
	int temp;
	temp = *num1;
	*num1=*num2;
	*num2=temp;
	printf("Swap performed in function gives values : \n");
	printf("%d \n",*num1);
	printf("%d \n",*num2);

}
```

RESULT:



```
cyclops@cyclops:~/Desktop/PSIPL Semester 1/Experiment 3$ gcc pointersintro.c
cyclops@cyclops:~/Desktop/PSIPL Semester 1/Experiment 3$ ./a.out
Before swap
5
7
Swap performed in function gives values :
7
5
After swap
7
5
cyclops@cyclops:~/Desktop/PSIPL Semester 1/Experiment 3$ 
```

CONCLUSION: **I have understood the way to utilize functions to incorporate**

	modularity.