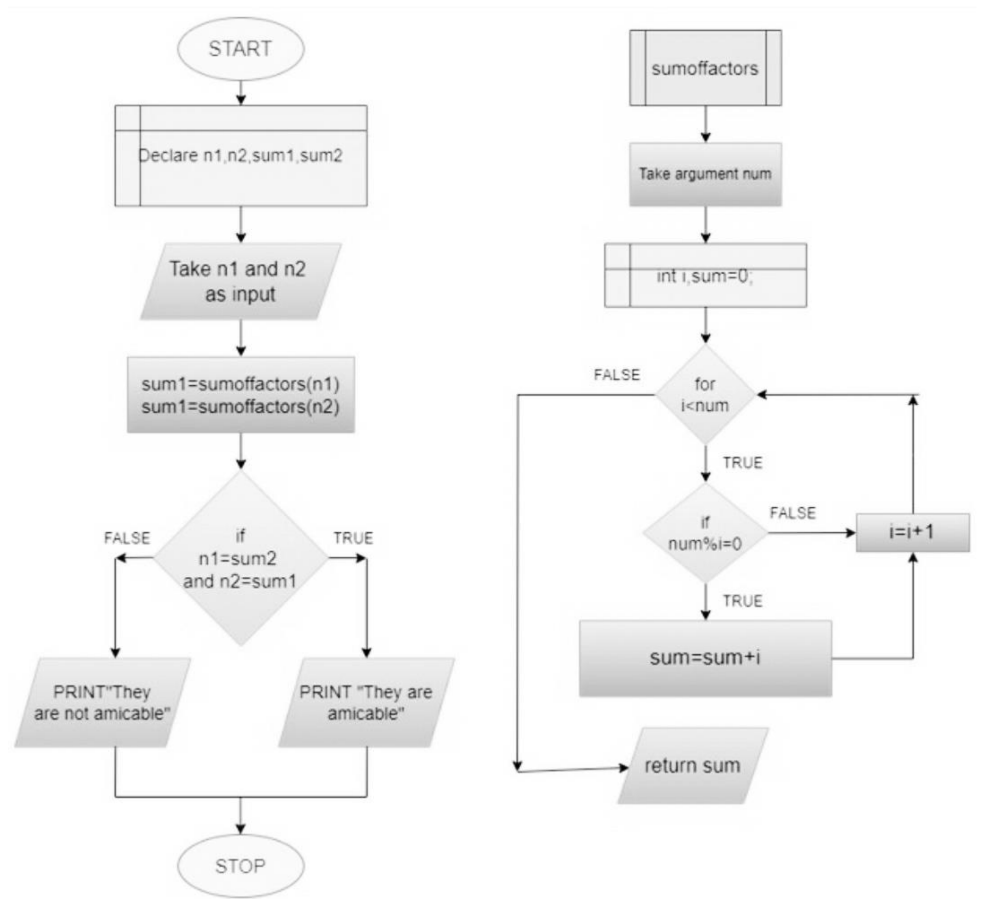
Name	Lekh Sanatan Nayak
UID no.	2023800068
Experiment No.	3

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. Define a function “sum_of_factors” which takes an interger ”n” as agruement and returns sum of its proper divisors. 2. In the function , use a loop to iterate from 1 to n-1. 3. Check if n is divisible by ‘i’, if it is add ‘i’ to ‘sum’. 4. Return sum form the function . 5. Define a function “main” and declare variable n1,n2,sum1,sum2. 6. Take user input for first and second number and calculate the sum of its factors. 7. Check if the sum of factors of n1 is equal to n2 and check the sum of factors of n2 is equal to n1. If both conditions are met print “They are amicable” if not print “They are not amicable numbers”. 8. Return 0 to end the program

FLOWCHART:

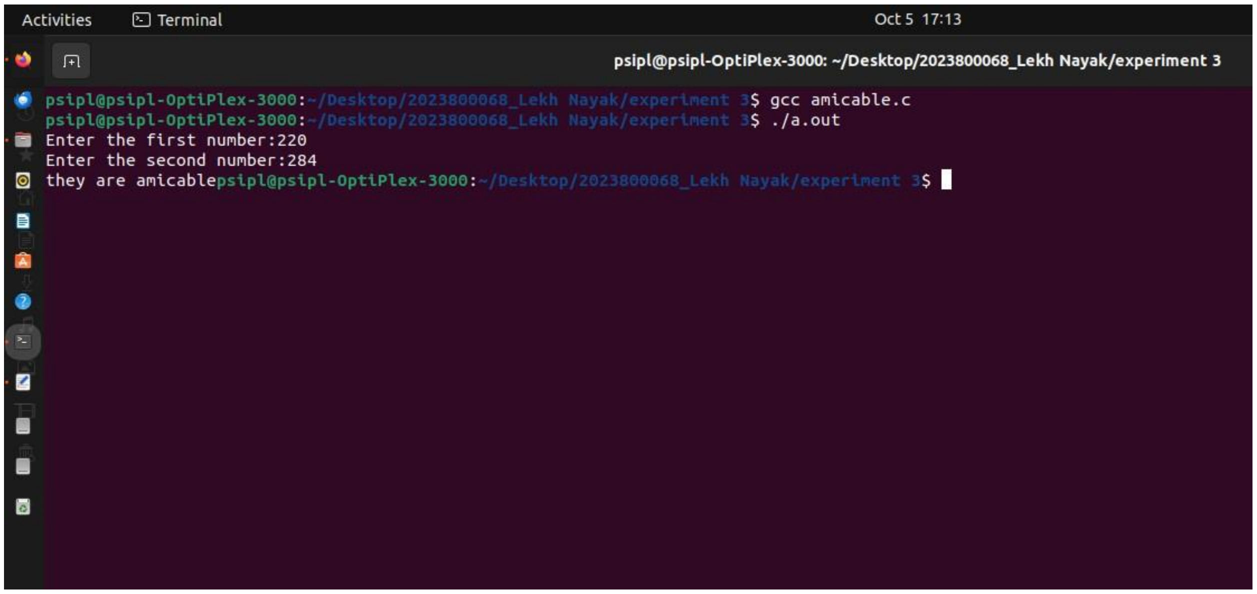


PROGRAM:

```
#include<stdio.h>

int sum_of_factors(int n){
        int i,sum=0;
        for(i=1;i<n;i++){
                if(n%i==0)
                sum=sum+i;
        }
        return sum;
}

int main (){
        int n1,n2,sum1,sum2;

        printf("Enter the first number:");
        scanf("%d",&n1);
        sum1=sum_of_factors(n1);


        printf("Enter the second number:");
```

```
            scanf("%d",&n2);
            sum2=sum_of_factors(n2);


                        if (sum1==n2 &&sum2==n1){
                        printf("they are amicable");
                        }
                        else{
                        printf("they are not amicale");
                        }
return 0;
}
```

RESULT:


Activities
Terminal
Oct 5 17:13
psipl@psipl-OptiPlex-3000: ~/Desktop/2023800068_Lekh Nayak/experiment 3

```
psipl@psipl-OptiPlex-3000:~/Desktop/2023800068_Lekh Nayak/experiment 3$ gcc amicable.c
psipl@psipl-OptiPlex-3000:~/Desktop/2023800068_Lekh Nayak/experiment 3$ ./a.out
Enter the first number:220
Enter the second number:284
they are amicablepsipl@psipl-OptiPlex-3000:~/Desktop/2023800068_Lekh Nayak/experiment 3$
```

Program 2

PROBLEM STATEMENT :

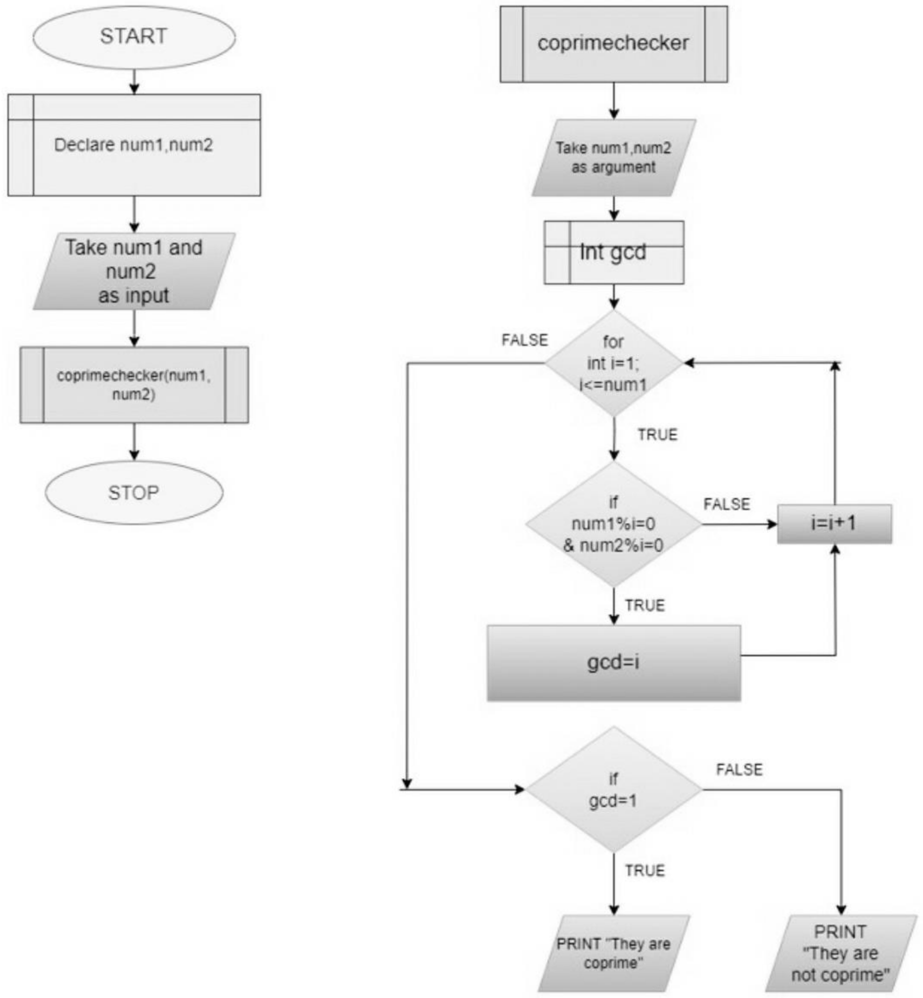
Write a function to find out whether given numbers are relatively prime (co-prime) or not. A number is relatively prime if the ‘1’ is the only common factor between the two numbers. For example: 9 and 8 are relatively prime. (9 =1x3x3 and 8=1x2x2x2).

ALGORITHM:

1. Start
2. Define a function “coprime” that takes two integers ‘n1’ and ‘n2’ as arguments.
3. In the coprime function declare the variable ‘gcd’ to store the greatest common divisor
4. For each iteration of the loop, check if both n1 and n2 are divisible by i.

5. If they are update the gcd to be equal to i.
6. After the loop check if the gcd is equal to 1.
7. If it is, print “The numbers are coprime” otherwise print “The numbers are not coprime”
8. In the “main” function declare to integer variables ‘num1’ and ‘num2’ to store the value of the numbers.
9. Take user input form the user for the value of num1 and num2
10. Call the “coprime” function with arguments num1 and num2 to check if they are coprime
11. Return 0 and end the program if succesfull.

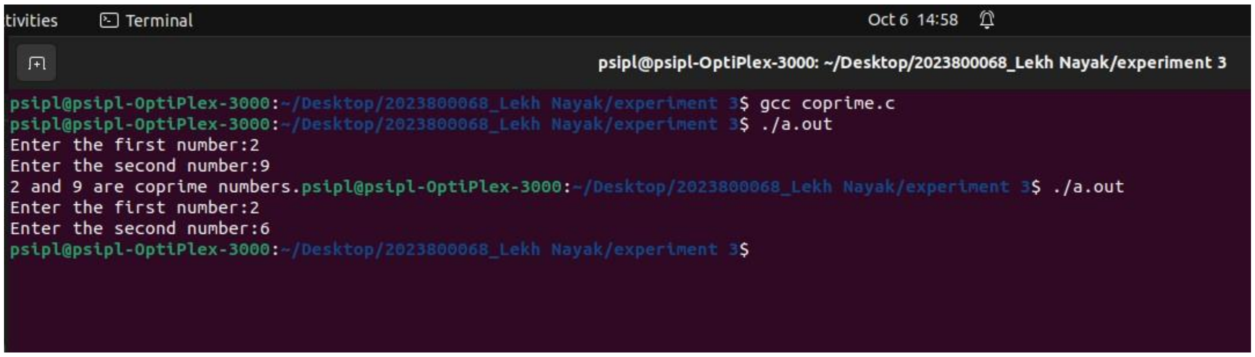
FLOWCHART:



PROGRAM:

```
#include<stdio.h>

void coprime(int n1,int n2){
    int gcd;
    for(int i=1;i<=n1;i++){
        if(n1%i==0 && n2%i==0){
            gcd=i;
        }
    }
    if(gcd==1){
        printf("%d and %d are coprime numbers.",n1,n2);
    }else{
        printf("%d and %d are not coprime numbers.",n1,n2);
    }

}

int main(){
    int num1,num2;

    printf("Enter the first number:");
    scanf("%d",&num1);
    printf("Enter the second number:");
    scanf("%d",&num2);

    coprime(num1,num2);
    return 0;
}
```

RESULT:



```
psipl@psipl-OptiPlex-3000:~/Desktop/2023800068_Lekh Nayak/experiment 3$ gcc coprime.c
psipl@psipl-OptiPlex-3000:~/Desktop/2023800068_Lekh Nayak/experiment 3$ ./a.out
Enter the first number:2
Enter the second number:9
2 and 9 are coprime numbers.psipl@psipl-OptiPlex-3000:~/Desktop/2023800068_Lekh Nayak/experiment 3$ ./a.out
Enter the first number:2
Enter the second number:6
psipl@psipl-OptiPlex-3000:~/Desktop/2023800068_Lekh Nayak/experiment 3$
```

Program 3

PROBLEM STATEMENT:	Write a function which takes a range as input. Print all the prime numbers in the range.
ALGORITHM:	1: Start 2: Include stdbool.h 3: Declare start,finish and take them as input 4: Call primeinrange(start,finish) function 5: Stop

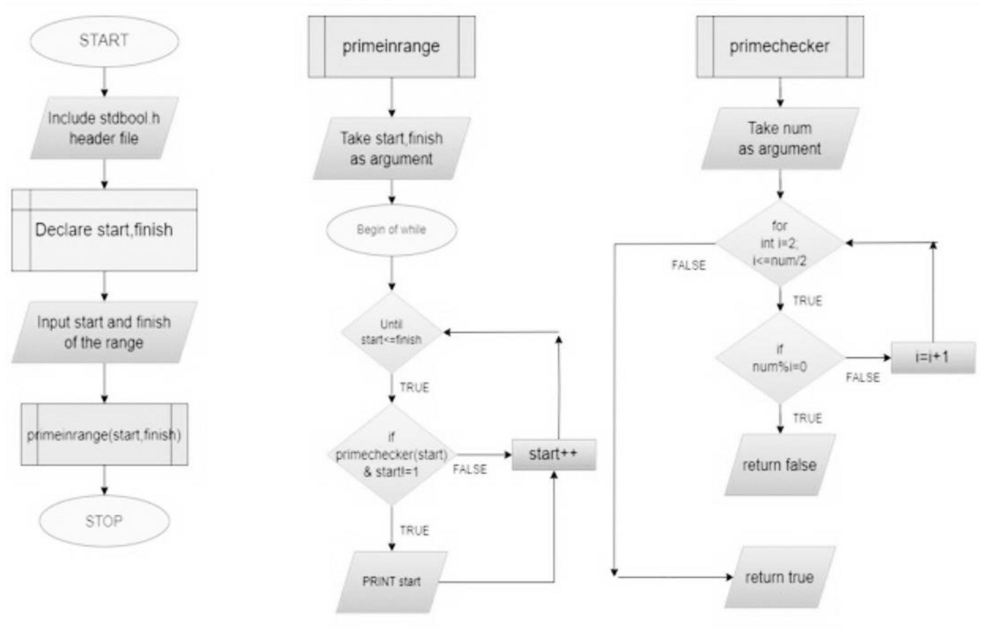
<u>Algorithm for primeinrange:</u>

```
1: Take two integers as argument start,finish
2: [BEGIN OF WHILE]
        Until start<=finish
        [START OF IF]
       If(primechecker(start)=true & start!=1)
         PRINT start
       [END OF IF]
       start=start+1
       [END OF WHILE LOOP]
```

<u>Algorithm for primechecker:</u>

```
1: Take a integer as argument num
2: [START OF FOR LOOP]
        Int i=2;
        i<=num/2
        [START OF IF]
         If(num%i=0)
           return false
        [END OF IF]
        i=i+1
       [END OF FOR LOOP]
3: return true
```

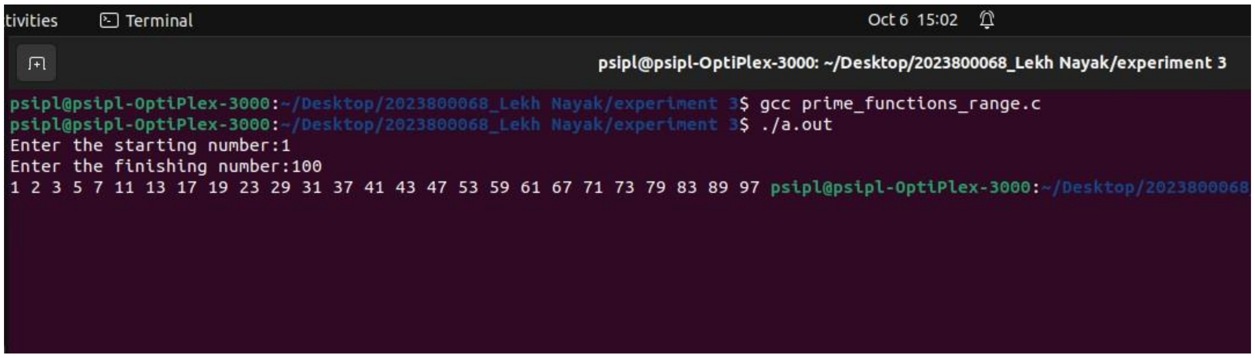
FLOWCHART:



PROGRAM:

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

bool prime_checker(int n);
void prime_range(int n1,int n2);

int main() {
   int start,finish;
    printf("Enter the starting number:");
      scanf("%d",&start);
    printf("Enter the finishing number:");
      scanf("%d",&finish);
      prime_range(start,finish);
    return 0;
}

bool prime_checker(int n){
    for(int i=2;i<=n/2;i++){
        if(n%i==0){
            return false;
        }
    }
    return true;
}
void prime_range(int n1,int n2){
    while(n1<=n2){
        if(prime_checker(n1)){
            printf("%d ",n1);
        }
        n1++;
    }
}
```

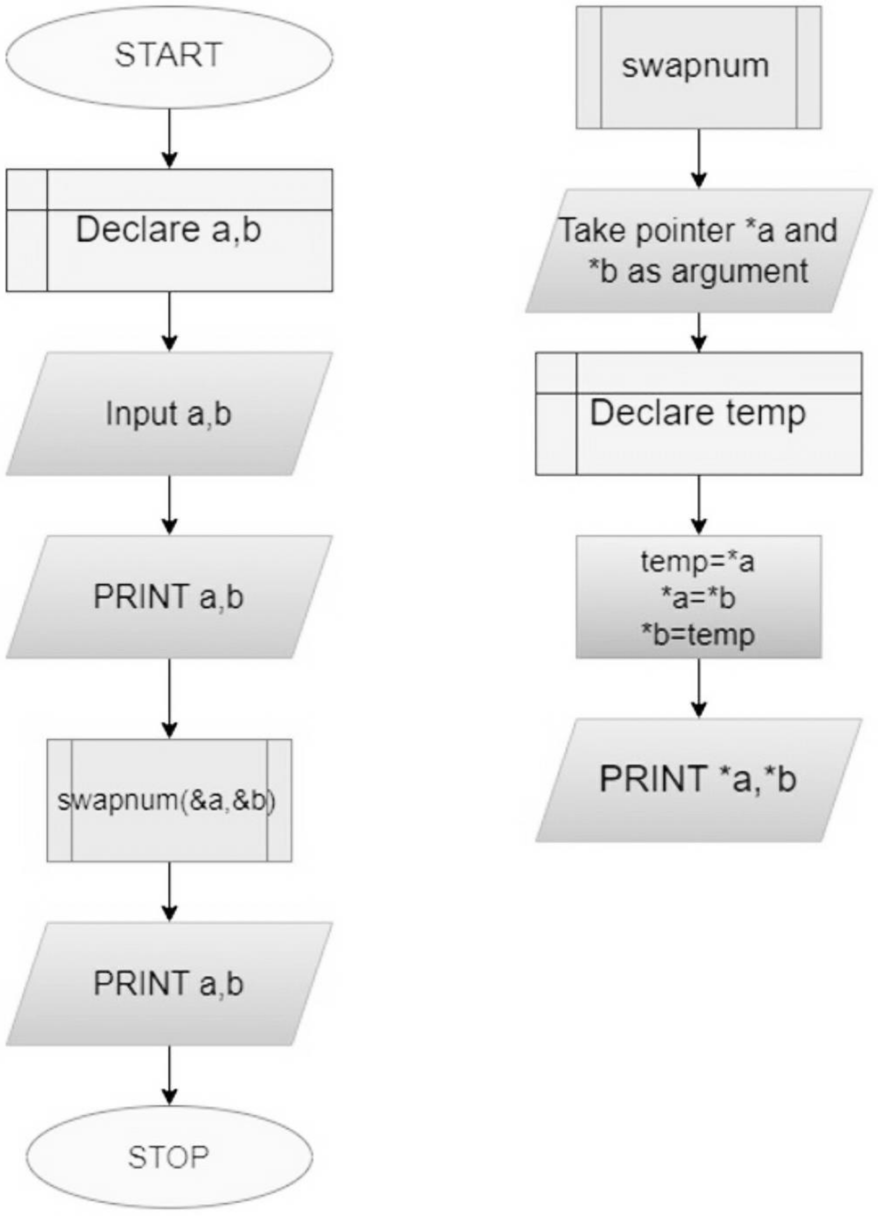
RESULT:


tivities
Terminal
Oct 6 15:02
psipl@psipl-OptiPlex-3000: ~/Desktop/2023800068_Lekh Nayak/experiment 3

```
psipl@psipl-OptiPlex-3000:~/Desktop/2023800068_Lekh Nayak/experiment 3$ gcc prime_functions_range.c
psipl@psipl-OptiPlex-3000:~/Desktop/2023800068_Lekh Nayak/experiment 3$ ./a.out
Enter the starting number:1
Enter the finishing number:100
1 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 psipl@psipl-OptiPlex-3000:~/Desktop/2023800068
```

Program 4

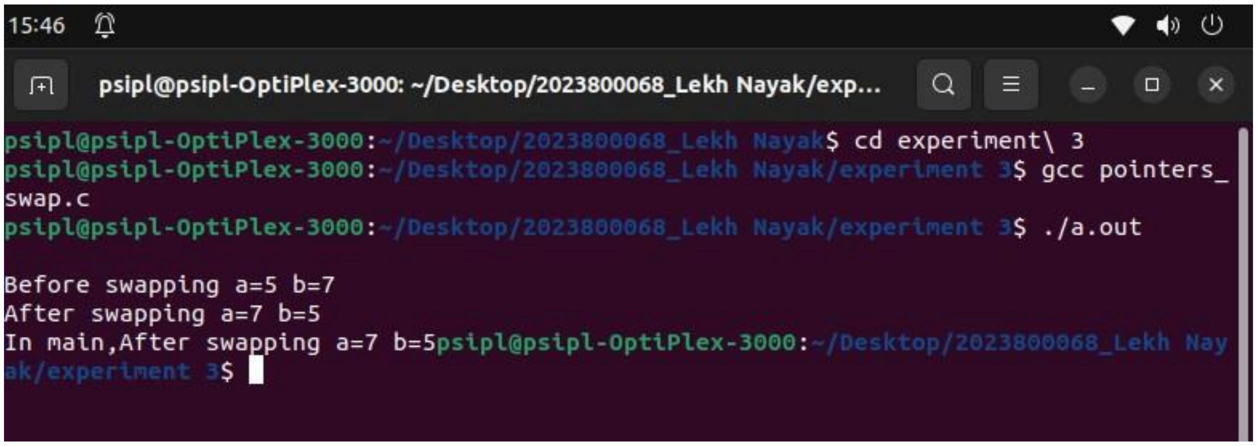
PROBLEM STATEMENT:	Write a function which swaps two integers using pointers as parameters.
ALGORITHM:	1. Define a function “swap” which takes 2 interger pointers as arguments. 2. Declare variable temp,a,b. 3. Swap the value of ‘a’ and ‘b’. 4. Print the swapped values 5. Define the function main and declare variable ‘a’ and ‘b’ and initialize them to 5 and 7 respectively. 6. Print the values of ‘a’and ‘b’ before swapping. 7. Call the “swap” function . 8. Print the values of ‘a’ and ‘b’ after swapping. 9. Return 0 to end the program.

FLOWCHART:	 START Declare a,b Input a,b PRINT a,b swapnum(&a,&b) PRINT a,b STOP swapnum Take pointer *a and *b as argument Declare temp temp=*a *a=*b *b=temp PRINT *a,*b

PROGRAM:	#include<stdio.h>
	void swap(int *a,int *b){
	    int temp;
	    temp=*a;
	    *a=*b;
	    *b=temp;
	    printf("\nAfter swapping a=%d b=%d",*a,*b);
	}
	
	int main(){
	    int a=5,b=7;
	    printf("\nBefore swapping a=%d b=%d",a,b);

	```
			swap(&a,&b);
			printf("\nIn main,After swapping a=%d b=%d",a,b);
			/*int *p;
			p=&a;
			printf("\n%p",p);
			printf("\np*=%d",*p);*/
	return 0;
	}
	```

RESULT:



```
15:46
psipl@psipl-OptiPlex-3000: ~/Desktop/2023800068_Lekh Nayak/exp...
psipl@psipl-OptiPlex-3000:~/Desktop/2023800068_Lekh Nayak$ cd experiment\ 3
psipl@psipl-OptiPlex-3000:~/Desktop/2023800068_Lekh Nayak/experiment 3$ gcc pointers_
swap.c
psipl@psipl-OptiPlex-3000:~/Desktop/2023800068_Lekh Nayak/experiment 3$ ./a.out

Before swapping a=5 b=7
After swapping a=7 b=5
In main,After swapping a=7 b=5psipl@psipl-OptiPlex-3000:~/Desktop/2023800068_Lekh Nay
ak/experiment 3$
```

CONCLUSION:	In this experiment I learnt how to apply the concept of functions to incorporate modularity