Name	Jhaveri Varun Nimitt
UID no.	2023800042
Experiment No.	6

AIM:	**Apply the concept of recursion to solve a given problem**
Program 1	
PROBLEM STATEMENT :	Write a recursive function to multiply 2 numbers
ALGORITHM:	1. Start 2. Define an int function mult with arguments a and b. 3. If b is 0, return 0 as the multiplication of any number with 0 is always 0. 4. Else , recursively call mult with arguments a and b-1 and store the result in variable answer. 5. Return the value of variable answer. 6. In the main function: A. Declare two integer variables num1 and num2. B. Print "Enter numbers:". C. Accept values for num1 and num2. D. Call the mult function with parameters num1 and num2 and store the result in variable ans. E. Print ans 7. End

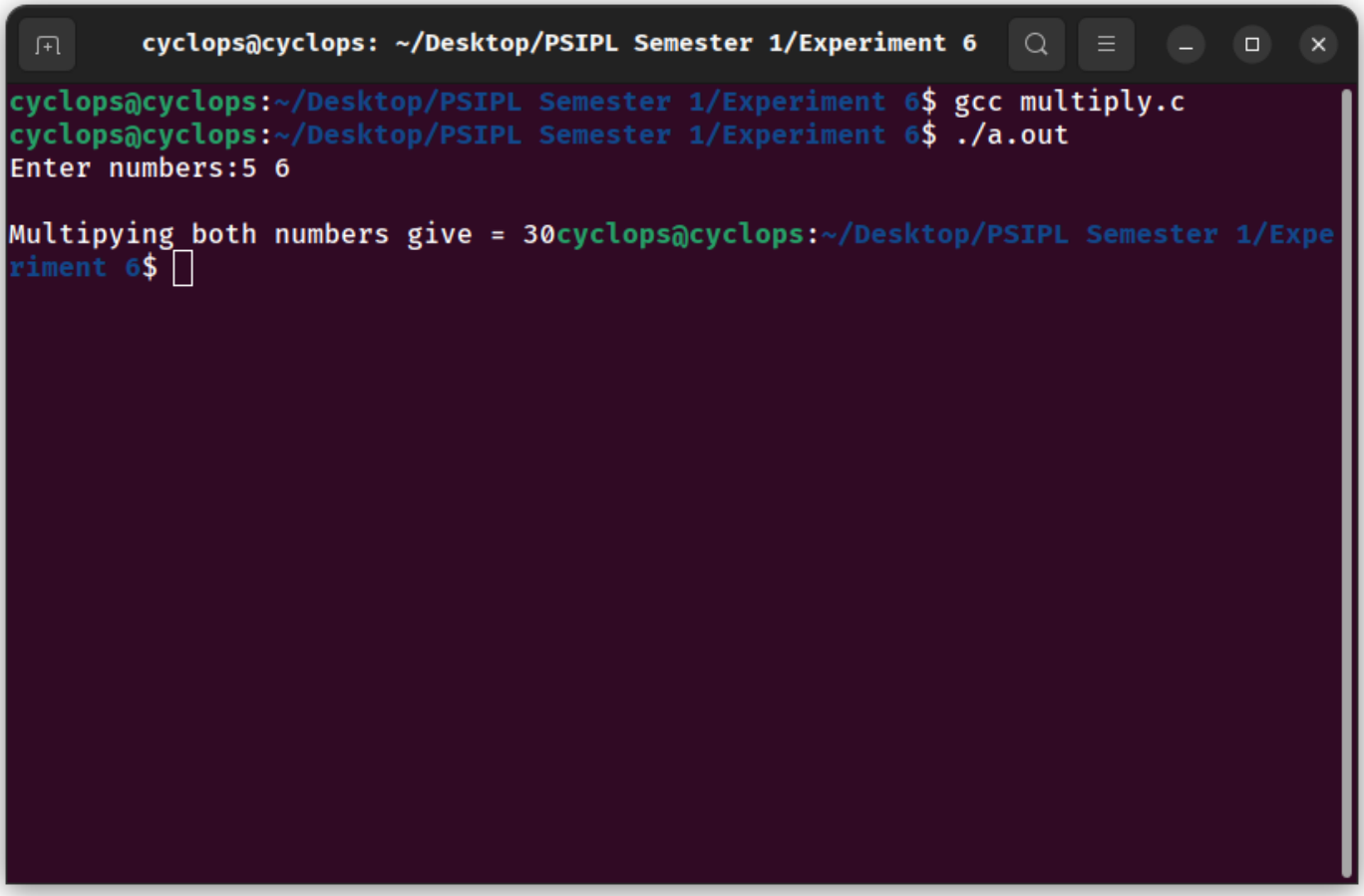
PROGRAM:

```c
#include <stdio.h>
int mult(int a,int b)
{
	//printf ("current = %d%d\n",a,b);
	int answer =0;
	if (b==0)
	{
		return 0;
	}
	else
	{
		answer=a+mult(a,b-1);
	}
	return answer;

}


int main()
{
	int num1,num2;
	printf("Enter numbers:");
	scanf("%d%d",&num1,&num2);
	int ans =mult(num1,num2);
	printf("\nMultipying both numbers give = %d",ans);
	return 0;
}
```

RESULT:

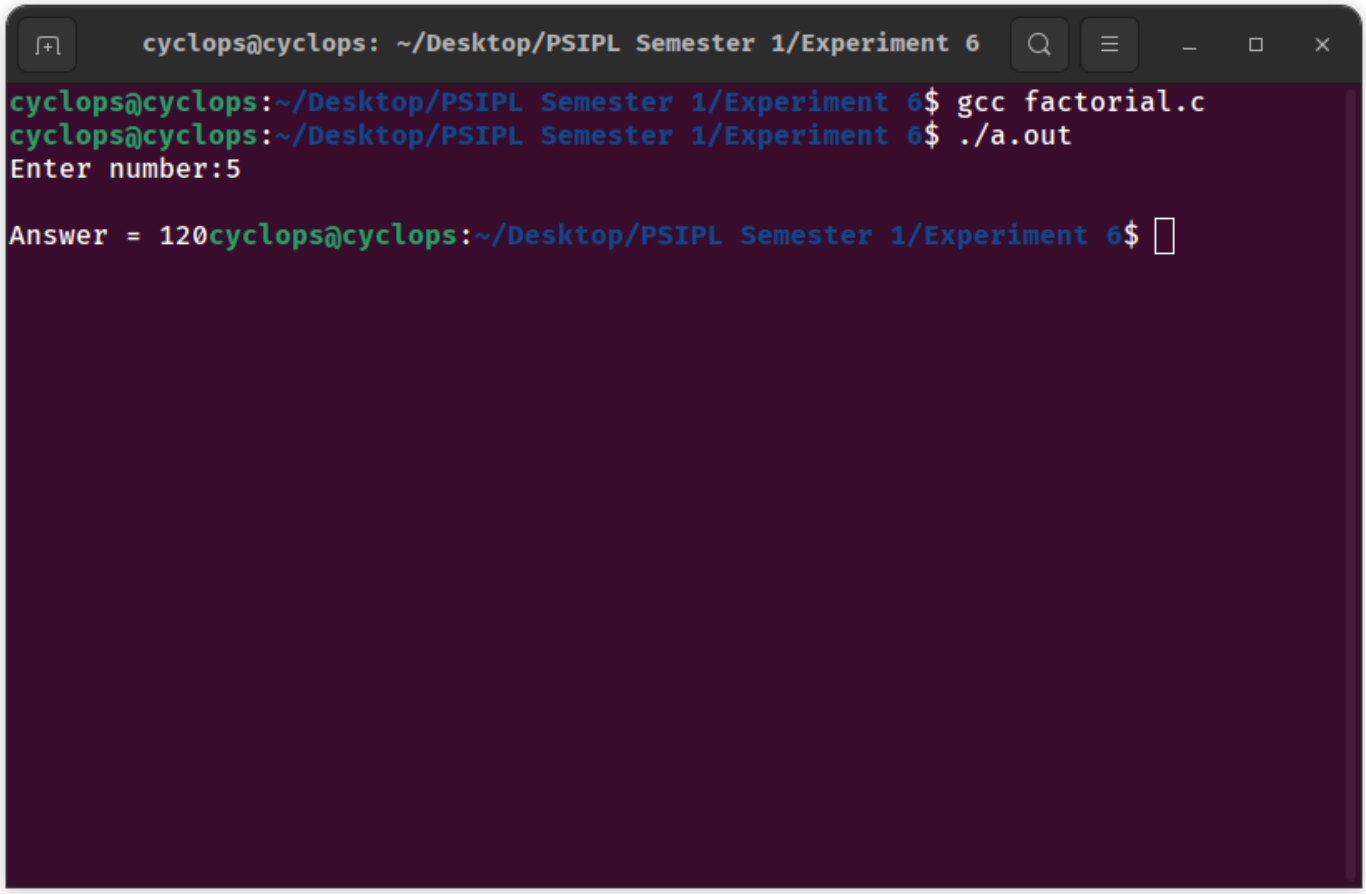

cyclops@cyclops: ~/Desktop/PSIPL Semester 1/Experiment 6
cyclops@cyclops:~/Desktop/PSIPL Semester 1/Experiment 6$ gcc multiply.c
cyclops@cyclops:~/Desktop/PSIPL Semester 1/Experiment 6$ ./a.out
Enter numbers:5 6
Multipying both numbers give = 30cyclops@cyclops:~/Desktop/PSIPL Semester 1/Expe
riment 6$

Program 2	
PROBLEM STATEMENT :	Write a recursive function to find the factorial of a number and test it.
ALGORITHM:	1. Start 2. Define an int function fact with the argument number. 3. If number is 0, return 1 4. Else, multiply number * and recursively call fact with number-1 and store the result in variable answer. 5. Return the value of answer. 6. In the main function: A. Declare an integer variable n. B. Input the value for n from the user. C. Call the fact function with parameter n and store the result in ans. D. Print ans 7. End

PROGRAM:	`#include <stdio.h>`
	`int fact(int number)`
	`{`
	`//printf ("current = %d\n",number);`
	`int answer =1;`
	`if (number == 0)`
	`{`
	`return 1;`
	`}`
	`else`
	`{`
	`answer = number*fact(number-1);`
	`}`
	`return answer;`
	`}`
	`int main()`
	`{`

```
    int n;
    printf("Enter number:");
    scanf("%d",&n);
    int ans =fact(n);
    printf("\nAnswer = %d",ans);
    return 0;
}
```

RESULT:

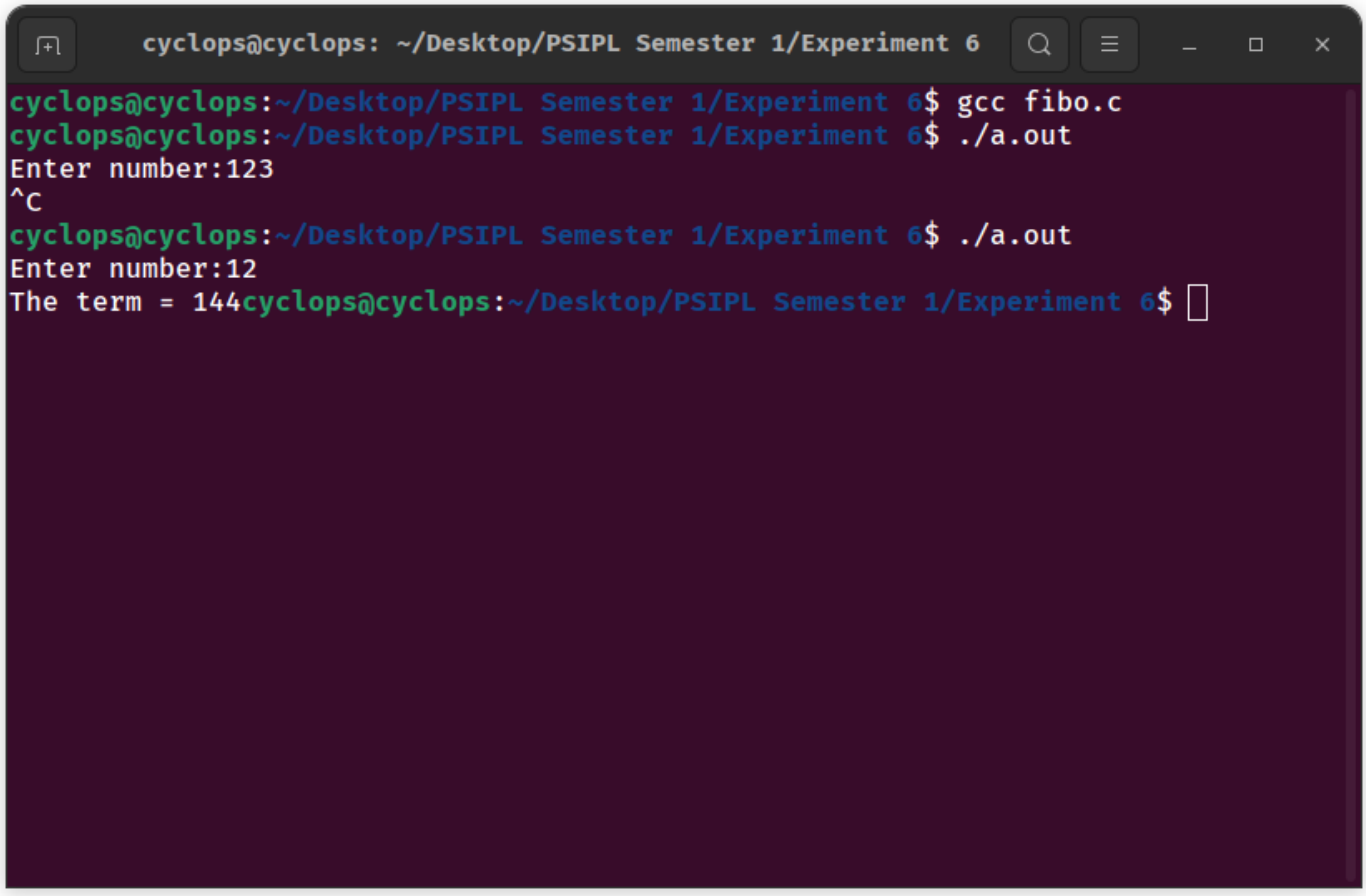


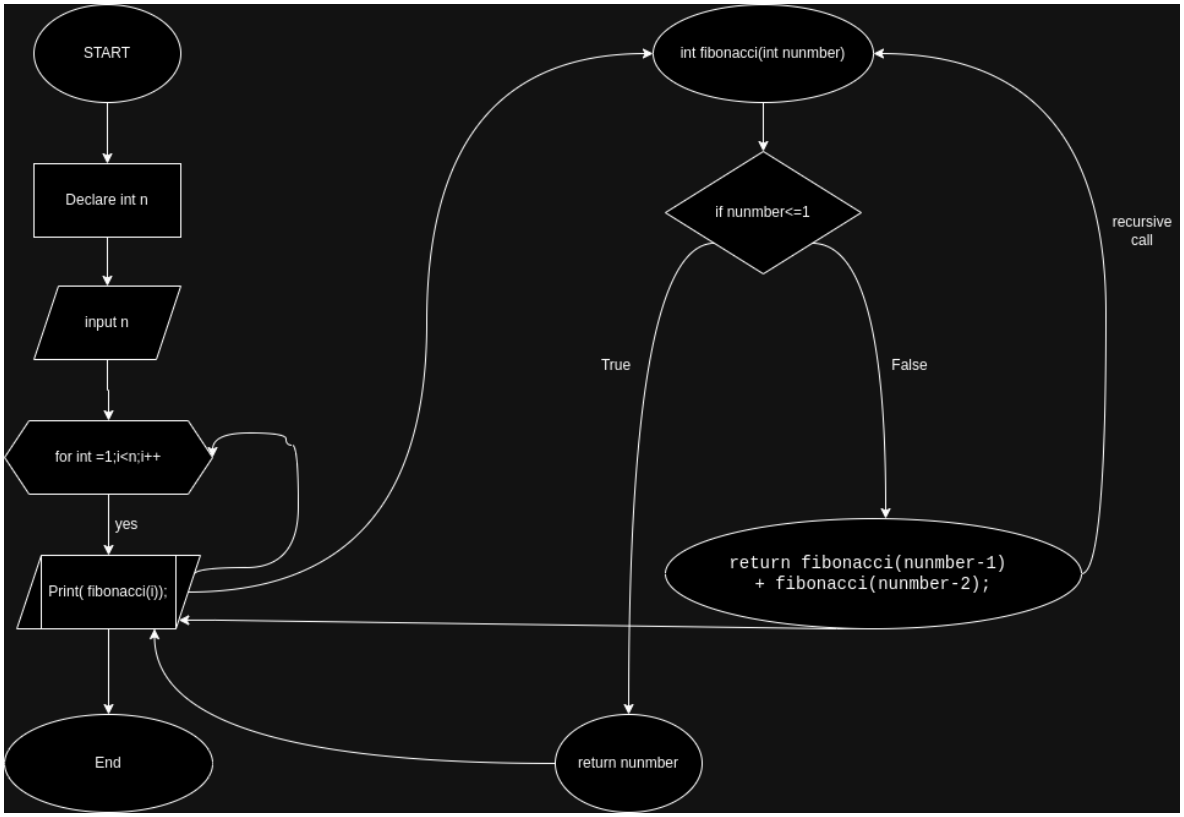
Program 3A	
PROBLEM STATEMENT:	Write a recursive function which returns the nth term of the fibonacci series.
ALGORITHM:	1. Start 2. Define an int function fibonacci with the argument number. 3. If the number is less than or equal to 1, return the number itself as the Fibonacci

	sequence for 0 and 1 is the number itself. 4. If the number is greater than 1: • return fibonacci(nunmber-1) + fibonacci(nunmber-2); (recursive) 5. In the main function: A. Declare an integer variable num. B. Accept the value for num from user C. Call the fibonacci function with parameter num and store the result in the variable ans. D. Print ans 6. End \

PROGRAM:	`#include <stdio.h>`
	`int fibonacci(int nunmber)`
	`{`
	`//printf ("current = %d\nunmber",nunmber);`
	
	`if (nunmber <= 1)`
	`{`
	`return nunmber;`
	`}`
	
	`return fibonacci(nunmber-1) + fibonacci(nunmber-2);`
	
	`}//1 1 2 3 5 8 13`
	
	`int main()`
	`{`
	`int num;`
	`printf("Enter number:");`
	`scanf("%d",&num);`
	`int ans =fibonacci(num);`
	`printf("The term = %d",ans);`
	`return 0;`
	`}`

RESULT:


cyclops@cyclops: ~/Desktop/PSIPL Semester 1/Experiment 6
cyclops@cyclops:~/Desktop/PSIPL Semester 1/Experiment 6$ gcc fibo.c
cyclops@cyclops:~/Desktop/PSIPL Semester 1/Experiment 6$ ./a.out
Enter number:123
^C
cyclops@cyclops:~/Desktop/PSIPL Semester 1/Experiment 6$ ./a.out
Enter number:12
The term = 144cyclops@cyclops:~/Desktop/PSIPL Semester 1/Experiment 6$

Program 3B	
PROBLEM STATEMENT:	Call it from main() to find the 1st n numbers of the fibonacci series.
FLOWCHART:	

PROGRAM:	`#include<stdio.h>`
	`int fibonacci(int nunmber);`
	
	`int main()`
	`{`
	`  int n;`
	`  printf("Enter till what number u want fibo series : ");`

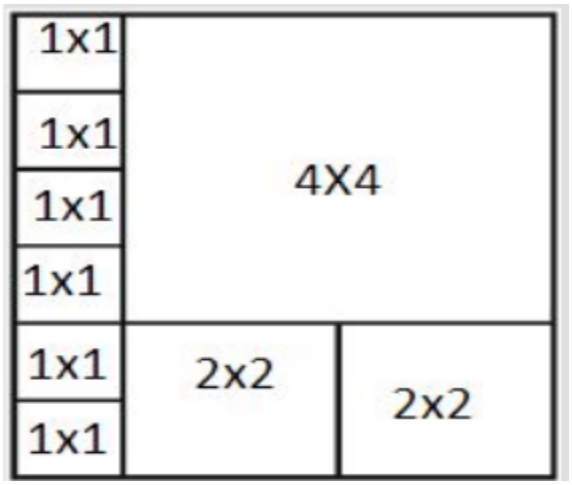
```
    scanf("%d",&n);

    printf("The Fibonacci numbers are : \n");

    for(int i=1;i<=n;i++)
    {
        printf("%d \t",fibonacci(i));
    }

    return 0;
}
int fibonacci(int nunmber)
{
    if (nunmber <= 1)
     {
          return nunmber;
    }

    return fibonacci(nunmber-1) + fibonacci(nunmber-2);
}
```

Program 4	
PROBLEM STATEMENT:	Given a room of area L x B. You have an infinite number of tiles of size 2n x 2n , where n = 0, 1,2,... so on. The task is to find the minimum number of square tiles required to fill the given area with tiles. 
ALGORITHM:	1. Start 2. Define the tilesfortheroom function with arguments length and breadth. 3. If both length and breadth are 0, return 0. 4. If both length and breadth are even then: ● Recursively call tilesfortheroom with length/2 and breadth/2 and store the result in ans. 5. If length is even and breadth is odd: ● Recursively call tilesfortheroom with length + tilesfortheroom(length/2, breadth/2) and store the result in ans. 6. If length is odd and breadth is even: ● Recursively call tilesfortheroom with breadth + tilesfortheroom(length/2, breadth/2) and store the result in ans. 7. If both length and breadth are odd: ● Recursively call tilesfortheroom with length + breadth - 1 + tilesfortheroom(length/2, breadth/2) and store the result in ans. 8. Return the value of ans. 9. In the main function: A. Declare two integer variables l and b. B. accept values for l and b from user C. Call the tilesfortheroom function with parameters l and b and store the result in ans. D. Print the ans .

	10. End
PROGRAM:	

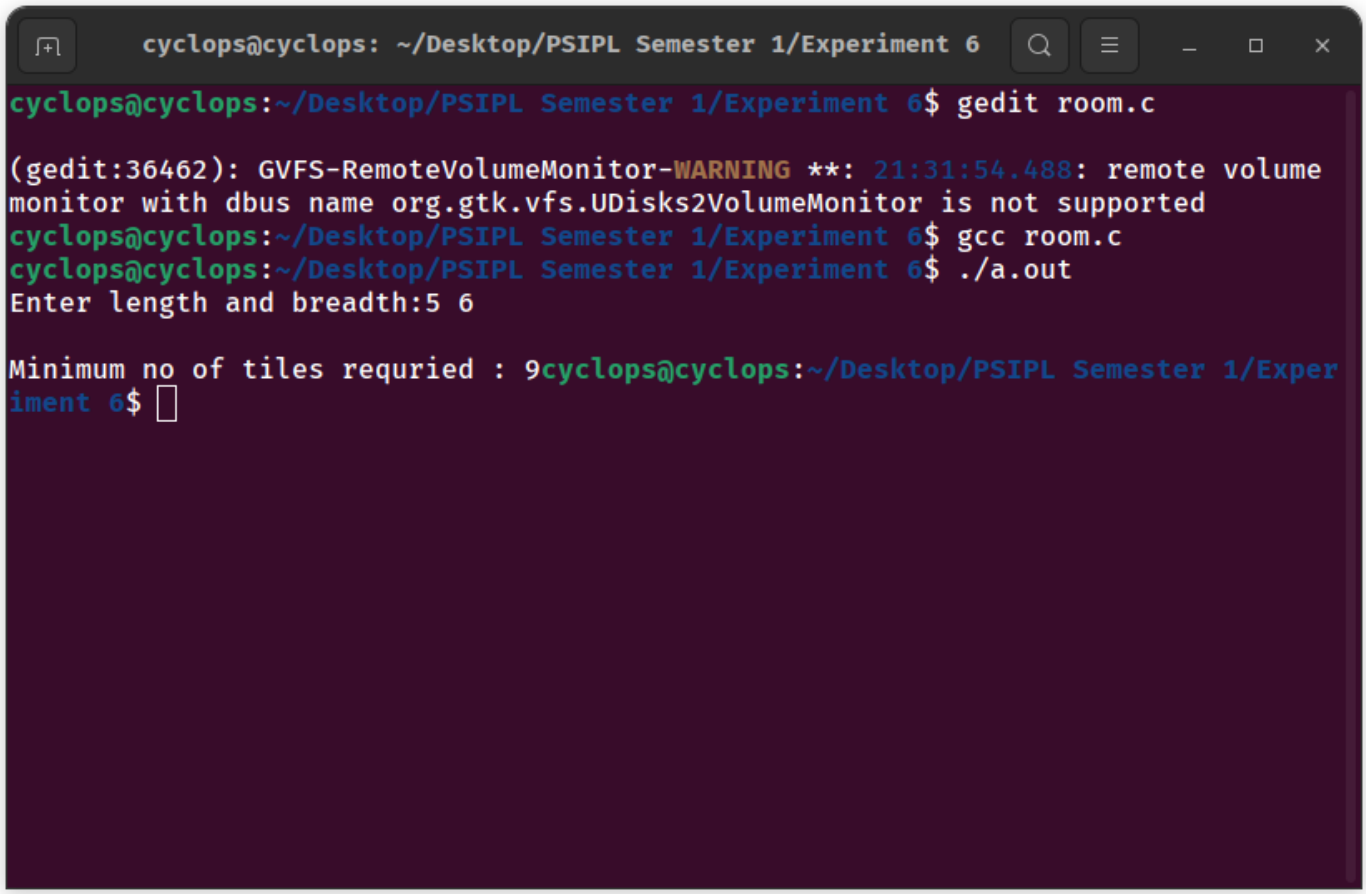
```c
#include <stdio.h>

int tilesfortheroom(int length, int breadth) // 56 23 11 00
{
      int ans=0;
    //printf("crurent : %d %d\n",n ,m);
    if ( length==0 && breadth==0)
    {
    return 0;
    }
    else if (length% 2 == 0 && breadth% 2 == 0)// n m even
    {
        ans=  tilesfortheroom(length/ 2, breadth/ 2);
    }
    else if (length% 2 == 0 && breadth% 2 == 1) // n even m odd
    {
        ans=  (length + tilesfortheroom(length/ 2, breadth/ 2));
    }
    else if (length% 2 == 1 && breadth% 2 == 0) //n odd m even
    {
        ans=  (breadth+ tilesfortheroom(length/ 2, breadth/ 2));
    }
    else //both odd
    {
        ans=  (length+ breadth - 1 + tilesfortheroom(length/ 2,
breadth/ 2));
    }
    return ans;
}

int main()
{
    int l , b ;
    printf("Enter length and breadth:");
    scanf("%d%d",&l ,&b);
    printf("\nMinimum no of tiles requried : %d",
tilesfortheroom(l, b));
```

```
    return 0;
}
```

RESULT:



CONCLUSION:	**I have understood how to use recursion to solve a given problem.**