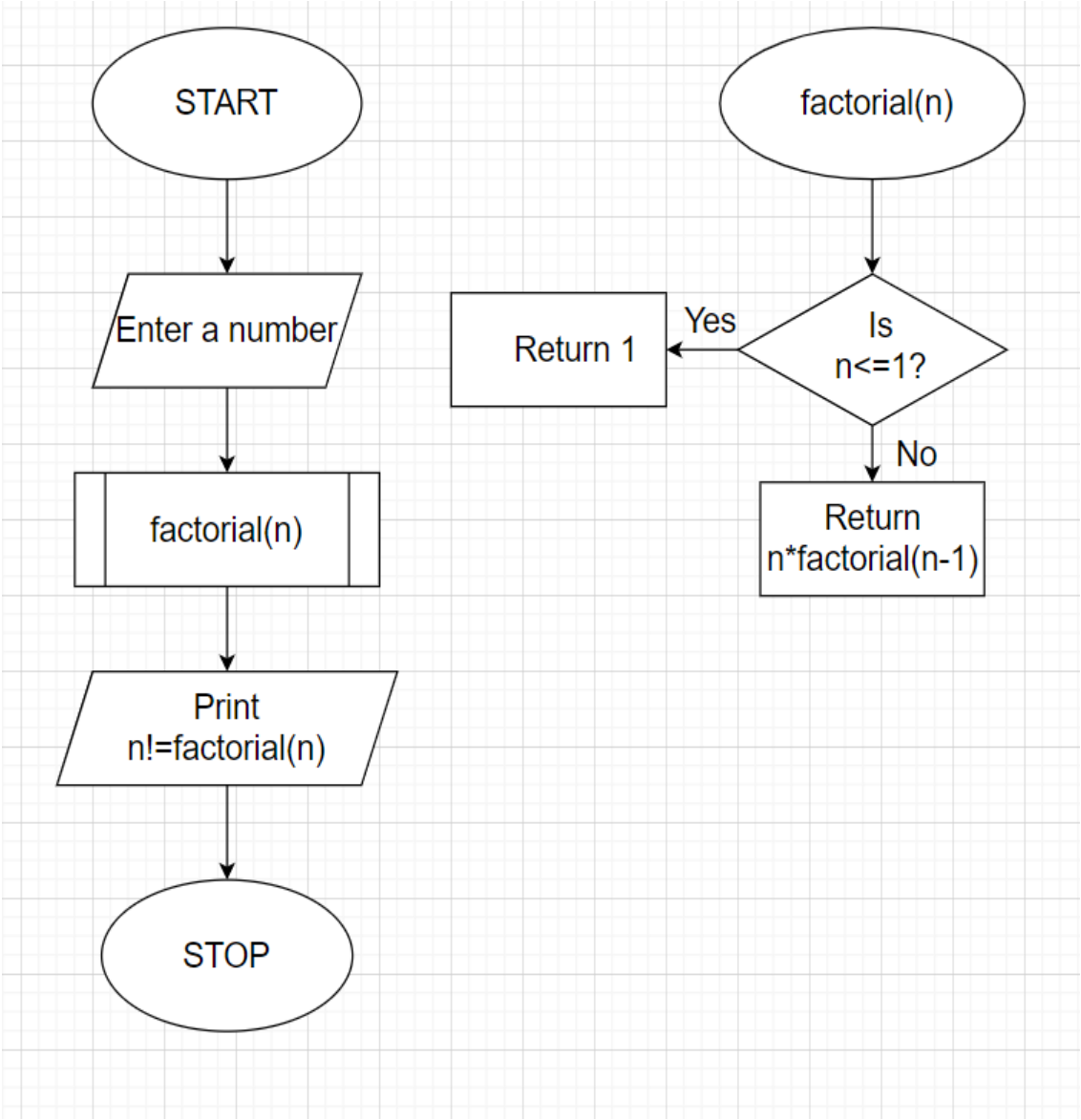
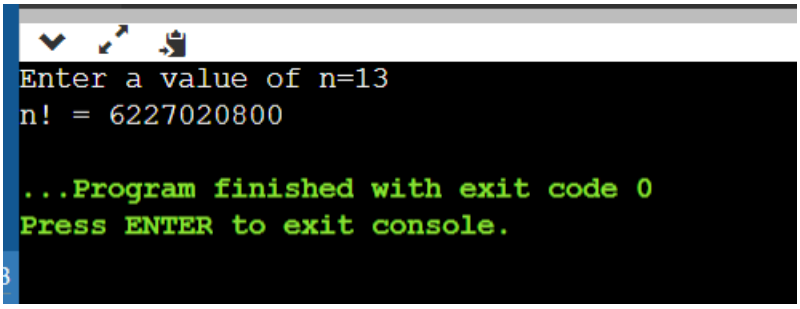
Name	Adwait S Purao
UID no.	2021300101
Experiment No.	4

AIM:	Apply the concept of recursion to solve a given problem.
Program 1	
PROBLEM STATEMENT :	Write a recursive function to find the factorial of a number and test it.
ALGORITHM:	1.START 2.Define function factorial with integer parameter n 3.If(n<=1) then return 1,else return n*factorial(n-1) 4.In main function,input number n 5.Call function factorial n 6.print value of n factorial 7.STOP

FLOWCHART:


START
Enter a number
factorial(n)
Print
n!=factorial(n)
STOP
factorial(n)
Is
n<=1?
Yes
Return 1
No
Return
n*factorial(n-1)

PROGRAM:	#include <stdio.h>
	long int factorial(int n);
	int main()
	{int n;
	printf("Enter a value of n=");
	scanf("%d",&n);
	printf("n! = %ld ",factorial(n));
	
	return 0;
	}
	long int factorial(int n){
	if(n<=1){
	return(1);
	
	}else{return(n*factorial(n-1));}
	}

RESULT: 	

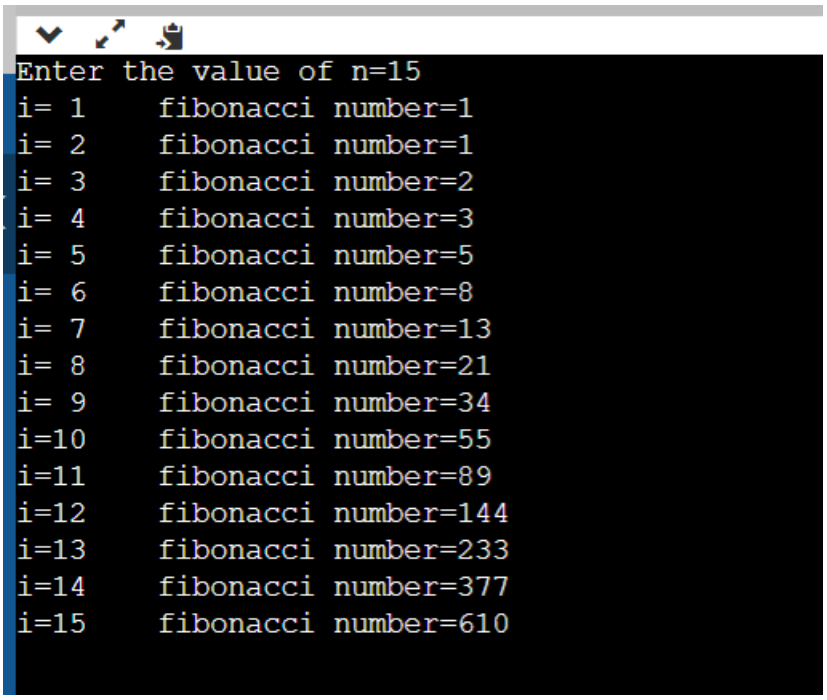
Program 2	
PROBLEM STATEMENT :	Write a recursive function which returns the nth term of the fibonacci series. Call it from main() to find the 1st n numbers of the fibonacci series.
ALGORITHM:	1.START 2.Define function fib with integer parameter count 3.If count=0 , return 0 4.If count=1 , return 1 5. else, return(fib(count-1)+fib(count-2)) 6.Define function main 7.Input number n 8. for(count=1;count<=n;++count)

	Print i= count, fibonacci number=fib(count),(Here the function fib(count) is being called) 9.STOP
FLOWCHART:	

PROGRAM:	#include<stdio.h>
	long int fib(int count);
	int main(){
	int n,count,i;
	printf("Enter the value of n=");
	scanf("%d",&n);
	
	for(count=1;count<=n;++count){
	printf("i=%2d fibonacci number=%ld \n",count,fib(count));
	}
	
	Return 0;
	}
	long int fib(int count){
	if(count==0) {return 0;}
	if(count==1){return 1;}

```
    else {
        return(fib(count-1)+fib(count-2));
    }
}
```

RESULT:

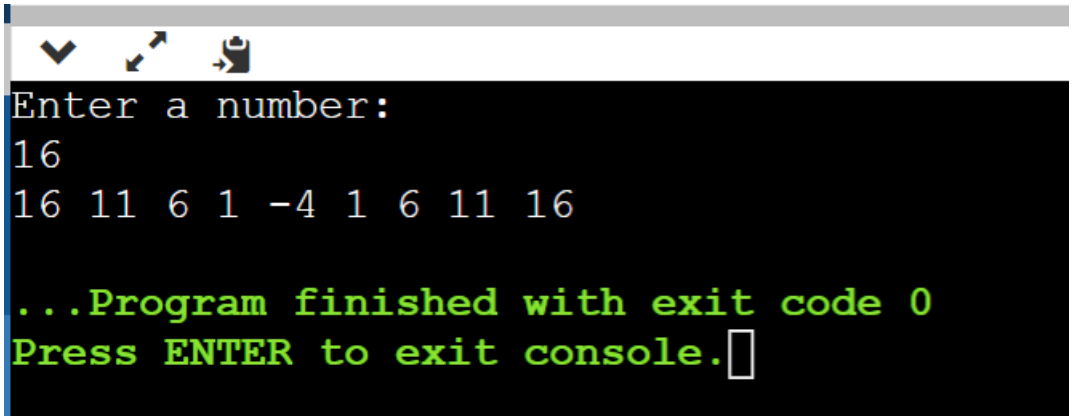


Program 3	
PROBLEM STATEMENT:	Given a number n, print following a pattern without using any loop. Example: Input: n = 16 Output: 16, 11, 6, 1, -4, 1, 6, 11, 16 Input: n = 10 Output: 10, 5, 0, 5, 10
ALGORITHM:	1.START 2.Define function series with integer parameter n 3.Define a variable num 4. if(n<=0),print n else,print n 5. num = 5 + series(n-5); Print num 6.In main function,Input a number n 7.Call function series(n) 8.STOP
FLOWCHART:	

PROGRAM:	#include <stdio.h>
	int series(int);
	int main()
	{int i,n;
	printf("Enter a number:\n");
	scanf("%d",&n);
	series(n);

```
    return 0;
}
int series(int n)
{
    int num;
    if(n<=0)
    {
        printf("%d ",n);
        return n;
    }
    else
    {
        printf("%d ",n);
        num = 5 + series(n-5);
        printf("%d ",num);
        return num;
    }
}
```

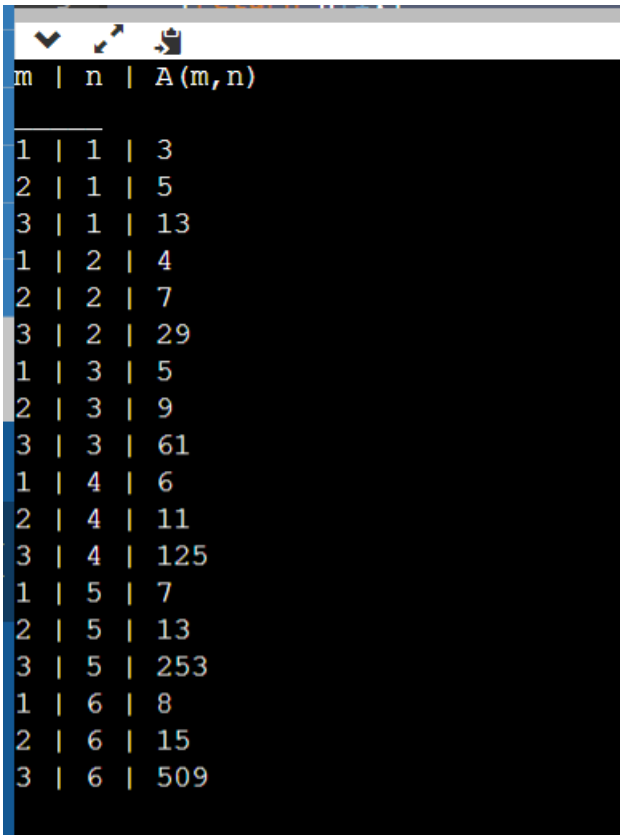
RESULT:



Program 4	
PROBLEM STATEMENT:	Ackerman's function is defined by: A(m,n)=n+1 if m=0 =A(m-1,1) if m≠0 and n=0 =A(m-1, A(m,n-1)) if m≠0 and n≠0 Write a function which given m and n returns A(m,n). Tabulate the values of A(m,n) for all m in the range 1 to 3 and all n in the range 1 to 6.
ALGORITHM:	1. START 2.Define function int ack(m,n) with integer parameter m and n 3. if(m==0),return n+1 else if(m!=0 && n==0), return ack(m-1,1);} else if(m!=0 && n!=0) return ack(m-1,ack(m,n-1)) 4.Define function void table 5.Initialize m and n to 1 6. print m | n | A(m,n) 7. print _____ 8. while(n<=6), while(m<=3) Print m,n,ack(m,n) 9.m++,n++ 10.Define main function,in it call function table 11.STOP
FLOWCHART:	
PROGRAM:	#include<stdio.h> int ack(int m,int n) {

```
   if(m==0)
   {return n+1;}
   else if(m!=0 && n==0)
   {return ack(m-1,1);}
   else if(m!=0 && n!=0)
   {
     return ack(m-1,ack(m,n-1));
   }


}

void table()
{
   int m=1;
   int n=1;
   printf("m | n | A(m,n)\n");
   printf("_____\n");
   while(n<=6)
   {
      while(m<=3)
      {
         printf("%d | %d | %d\n",m,n,ack(m,n));

         m++;
      }
      m=1;
      n++;
   }
}

int main()
{  table();
   return 0;
}
```

RESULT:



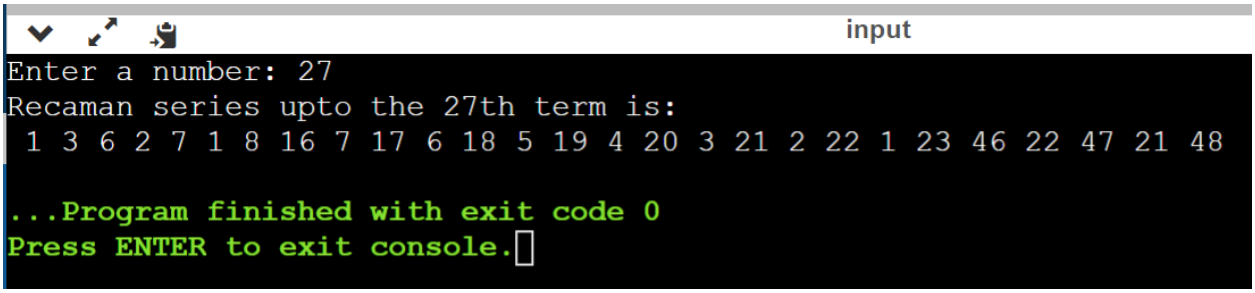
Program 5	
PROBLEM STATEMENT:	There are at least two sequences attributed to B. Recamán. One is the sequence an formed by taking a1=1 and letting an =an-1 -n if an-1 -n>0 and is new =an-1 +n otherwise which can be succinctly defined as "subtract if you can, otherwise add." The first few terms are 1, 3, 6, 2, 7, 13, 20, 12, 21, 11, ..so on.
ALGORITHM:	1.START 2.Define function rec(n) with integer parameter n 3.if(n==1),return 1 else if((rec(n-1)-n)>0)

	return rec(n-1)-n else, return rec(n-1)+n 4.In main function,input a number n 5.Recaman series upto the nth element is rec(i) 6.STOP
FLOWCHART:	

PROGRAM:	#include<stdio.h>
	int rec(int n)
	{
	if(n==1)
	{
	return 1;
	}
	else if((rec(n-1)-n)>0)
	{
	return rec(n-1)-n;
	}
	else
	{
	return rec(n-1)+n;
	}
	}
	int main()

```
{
   int n;
   printf("Enter a number: ");
   scanf("%d",&n);
   printf("Recaman series upto the %dth term is:\n ",n);
   for(int i=1;i<=n;i++)
   {
      printf("%d ",rec(i));
   }
   return 0;
}
```

RESULT:



CONCLUSION:	We learnt about Recursions in the above experiment, which actually means it calls itself directly or indirectly which helps in making the code shorter and easier.