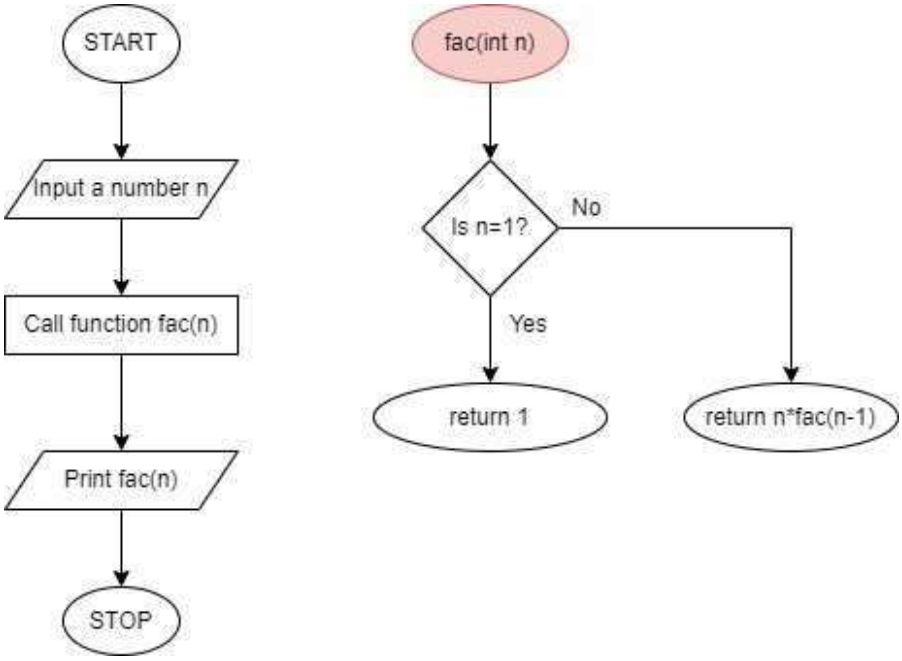
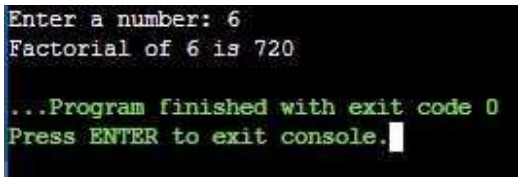
Name	Vineet Parmar
UID no.	2021300092
Experiment No.	4

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 fac(int n) with an integer number n 3. If n = 1 return 1 else return (n*fac(n-1)) 4. Define main() 5. Input a number n 6. Call fac(n) 7. Print value of function 8. STOP
FLOWCHART:	

PROGRAM:

```
#include<stdio.h>
int fac(int n)
{
   if (n==1)
   {
      return 1;
   }
   else
   {
      return n*fac(n-1);
   }
}
int main()
{
   int n;
   printf("Enter a number: ");
   scanf("%d",&n);
   printf("Factorial of %d is %d",n,fac(n));
   return 0;
}
```

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.

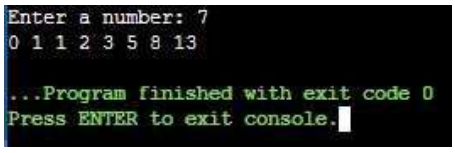
ALGORITHM:

1. START
2. Define function fib(int n) with an integer parameter n
3. If n <=1
   Return n
   Else
   Return fib(n-1) + fib(n-2)
4. Define function main()
5. Input a number n
6. i=1
7. Print fib(i)
8. i++
9. Repeat 7 and 8 till i=n
10. STOP

PROGRAM:

```
#include<stdio.h>
int fib(int n)
{
   if (n<=1)
   {
      return n;
   }
   else
   {
      return (fib(n-1)+fib(n-2));
   }
}
int main()
{
   int n;
   printf("Enter a number: ");
   scanf("%d",&n);
   for(int i=0;i<=n;i++)
   {
      printf("%d ",fib(i));
   }
   return 0;
}
```

RESULT:



Program 3

PROBLEM STATEMENT:

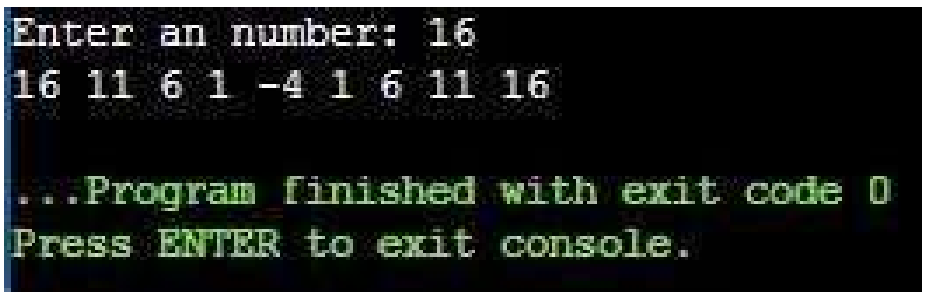
Given a number n, print following a pattern without using any loop.
Example:
Input: n = 16
Output: 16, 11, 6, 1, -4, 1, 6, 11, 16
Input: n = 10
Output: 10, 5, 0, 5, 10

ALGORITHM:

1. START
2. Define function series(int n) with an integer parameter n
3. Initalize num
4. If n<=0
   Print and return n
   Else
   {

	Print n num = n + series(n-5) Print and return num } 5. Define function main() 6. Input a number n 7. Call function series(n) 8. STOP

PROGRAM:	#include<stdio.h>
	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;
	}
	}
	int main()
	{
	int n;
	printf("Enter an number: ");
	scanf("%d",&n);
	series(n);
	return 0;
	}

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 ackerman(int m,int n) with two integer parameters m and n 3. If m=0 Return n+1 Else if (m>0 AND n=0) Return ackerman(m-1,1) Else Return ackerman(m-1,ackerman(m,n-1)) 4. Define function main() Print M <Tabspace> N <Tabspace> Ackerman Value 5. i=1 6. j=1 7. Print i <Tabspace> j <Tabspace> ackerman(i,j) 8. j++ 9. Repeat 7 and 8 till j=6 10. I++ 11. Repeat 6,7,8,9 and 10 till i=n 12. STOP

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

```
    }
}
int main()
{
    printf("m\tn\tAckerman Value");
    for(int i=1;i<=3;i++)
    {
        for(int j=1;j<=6;j++)
        {
            printf("\n%d\t%d\t%d",i,j,ackerman(i,j));
        }
    }
    return 0;
}
```

RESULT:

```
m       n       Ackerman Value
1       1       3
1       2       4
1       3       5
1       4       6
1       5       7
1       6       8
2       1       5
2       2       7
2       3       9
2       4       11
2       5       13
2       6       15
3       1       13
3       2       29
3       3       61
3       4       125
3       5       253
3       6       509

...Program finished with exit code 0
Press ENTER to exit console.
```

Program 5

PROBLEM STATEMENT:

There are at least two sequences attributed to B. Recamán. One is the sequence a_n formed by taking $a_1=1$ and letting $a_n=a_{n-1}-n$ if $a_{n-1}-n>0$ and is new $=a_{n-1}+n$ 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. Initialize array arr[101] and all elements to 0 3. Define integer function recaman(int n) with an integer parameter 4. If n=1 arr[1]=1 Return 1 Else if recaman(n-1)-n > 0 and arr[t]=0 arr[t]=1 Return recaman(n-1) – n Else arr[t+2*n] = 1 Return recaman(n-1)+n 5. Define void function reset() 6. I=1 7. Arr[I]=0 8. I++ 9. Repeat 7,8 till I<=100 10. Define function main() 11. I=1 12. T= series(I) 13. Print T 14. Call function Reset() 15. I++ 16. Repeat 12,13,14,15 till I<=N 17. STOP.

PROGRAM:

```
#include <stdio.h>
int arr[101]={0};

int series(int n)
{
   if(n==1)
   {
      arr[1]=1;
      return 1;
   }
   else
   {
      int t=series(n-1)-n;
      if(t>0 && arr[t]==0 )
      {
         arr[t]=1;
         return t;
      }
      else
      {
         arr[t+2*n]=1;
         return (t+2*n);
      }
   }

}
void reset()
{
   for(int i=1;i<=100;i++)
   {
      arr[i]=0;
   }
}
int main()
{
   int n;
   printf("Enter the number of terms of the series\n");
   scanf("%d",&n);
   for(int i=1;i<=n;i++)
   {
```

```
        int t=series(i);
        printf("%d ",t);
        reset();

    }
    return 0;
}
```

RESULT:

```
Enter the number of terms of the series
15
1 3 6 2 7 13 20 12 21 11 22 10 23 9 24

...Program finished with exit code 0
Press ENTER to exit console.
```

CONCLUSION:	We learnt how calling a function within the function itself helps in shortening the code and that is what we call recursion. Writing recursion programs also helps in finding out formulae's of many patterns and so.