Experiment Name: Write a program to generate and display n terms of Fibonacci series using recursive function**.**

Objective:

- To understand how recursion works in C by generating the Fibonacci series up to n terms.

Problem analysis:

In this program, the objective is to generate and display the first n terms of the Fibonacci series using a recursive function. The user enters the number of terms, and the program uses a recursive function to calculate each term based on the relation:

F(n) = F(n-1) + F(n-2) with base conditions F(0) = 0 and F(1) = 1.

The recursive function is called repeatedly in a loop to print all terms of the series.

Input variable	Processing variable	Output variable	Header file
n → Total number of terms in the Fibonacci series	Recursive function fibonacci(int n) Loop counter (i)	First n terms of Fibonacci series	<stdio.h>

Algorithm:

Step1: Start

Step2: input number of terms n

Step3: Define recursive function:

If n == 0 → return 0

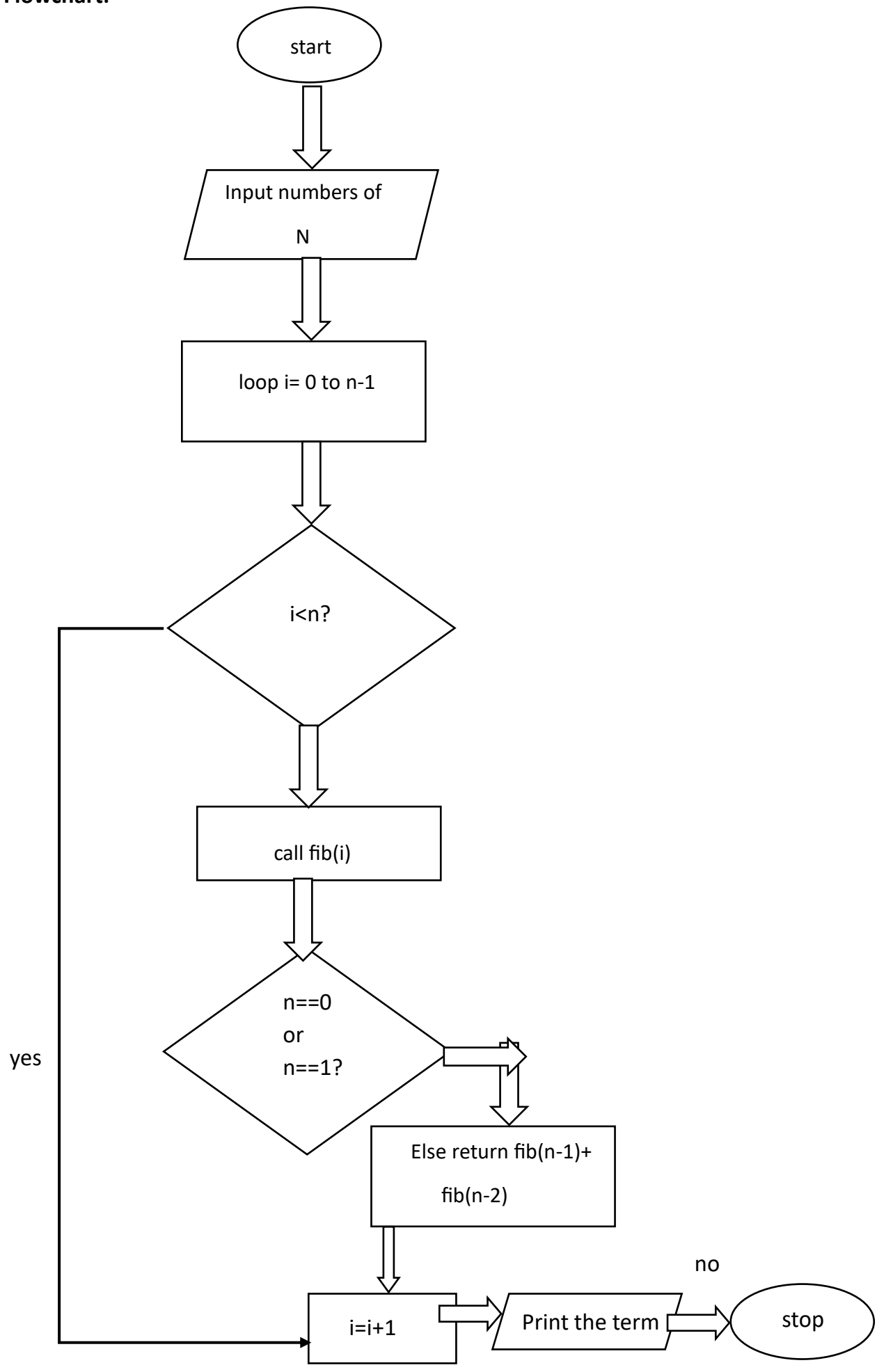
If n == 1 → return 1

Else → return fibonacci(n-1) + fibonacci(n-2)

Step4: Use loop from i = 0 to i < n Call and print fibonacci(i)

Step5: End

Flowchart:


start
Input numbers of
N
loop i= 0 to n-1
i<n?
call fib(i)
n==0
or
n==1?
yes
Else return fib(n-1)+
fib(n-2)
no
i=i+1
Print the term
stop

Source code:

lab6.c

```c
#include <stdio.h>
int fibonacci(int n)
{
if(n == 0)
    return 0;
else if(n == 1)
    return 1;
else
    return fibonacci(n - 1) + fibonacci(n - 2);
}

int main() {
int n, i;

printf("Enter number of terms: ");
scanf("%d", &n);

printf("Fibonacci series:\n");
for(i = 0; i < n; i++) {
    printf("%d ", fibonacci(i));
}

return 0;
}
```

```
D:\lab6\bin\Debug\lab6.exe
Enter number of terms: 15
Fibonacci series:
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377
Process returned 0 (0x0)   execution time : 6.852 s
Press any key to continue.
```

Discussion:

This program uses recursion to generate Fibonacci numbers based on its natural definition, which makes the logic simple. However, it's not efficient for large n due to repeated calculations. Still, it's a good way to understand recursion in C.