NAME: Yash Sarang
ROLL NO.: 47
CLASS: D1AD

CP LAB ASSIGNMENT 3

SECTION 1: FUNCTIONS

AIM: To print n terms of fibonacci series.
THEORY:

A function is a group of statements that together perform a task. Every C program has at least one function, which is main(), and all the most trivial programs can define additional functions.

You can divide up your code into separate functions. How you divide up your code among different functions is up to you, but logically the division is such that each function performs a specific task.

A function declaration tells the compiler about a function's name, return type, and parameters. A function definition provides the actual body of the function.

The C standard library provides numerous built-in functions that your program can call. For example, strcat() to concatenate two strings, memcpy() to copy one memory location to another location, and many more functions.

A function can also be referred to as a method or a subroutine or a procedure, etc.

DEFINING A FUNCTION:

The general form of a function definition in C programming language is as follows −

```
return_type function_name( parameter list ) {
  body of the function
}
```

A function definition in C programming consists of a function header and a function body. Here are all the parts of a function −

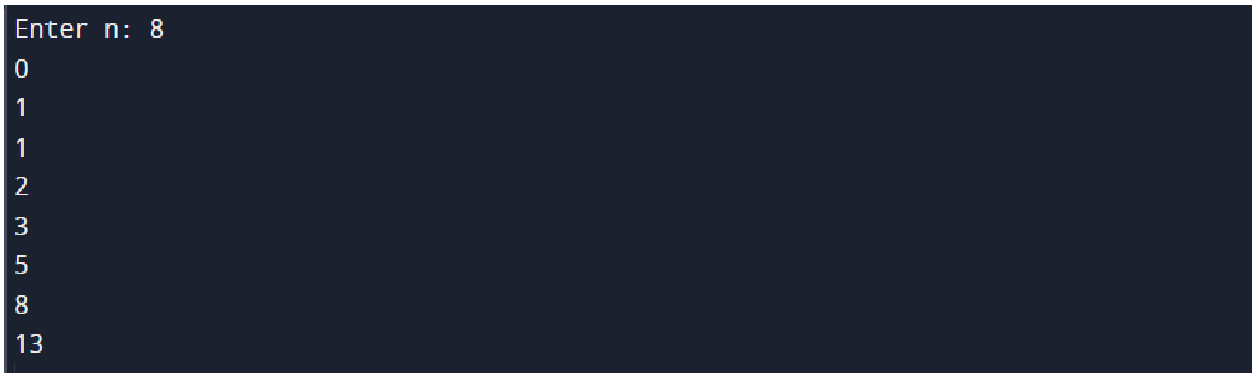
- RETURN TYPE − A function may return a value. The return_type is the data type of the value the function returns. Some functions perform the desired operations without returning a value. In this case, the return_type is the keyword void.
- FUNCTION NAME − This is the actual name of the function. The function name and the parameter list together constitute the function signature.

- PARAMETERS − A parameter is like a placeholder. When a function is invoked, you pass a value to the parameter. This value is referred to as an actual parameter or argument. The parameter list refers to the type, order, and number of the parameters of a function. Parameters are optional; that is, a function may contain no parameters.
- FUNCTIONS − The function body contains a collection of statements that define what the function does.

PROGRAM C CODE:

```
#include <stdio.h>
void fibonacci();
int main() {
   int n;
   printf("Enter n: ");
   scanf("%d", &n);
   fibonacci(n);
   return 0;
}
void fibonacci(int n)
{
   int t1 = 0, t2 = 1;
   for(int i = 1; i <= n; i++)
   {
      printf("%d\n", t1);
      int nt = t1 + t2;
      t1 = t2;
      t2 = nt;
   }
}
```

OUTPUT:



```
Enter n: 8
0
1
1
2
3
5
8
13
```

SECTION 2: Recursion

AIM:

THEORY:

Recursion is the process of repeating items in a self-similar way. In programming languages, if a program allows you to call a function inside the same function, then it is called a recursive call of the function.

```
void recursion() {
  recursion(); /* function calls itself */
}

int main() {
  recursion();
}
```

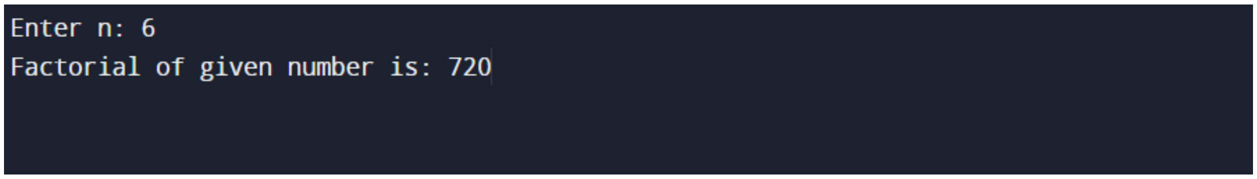
The C programming language supports recursion, i.e., a function to call itself. But while using recursion, programmers need to be careful to define an exit condition from the function, otherwise it will go into an infinite loop.

Recursive functions are very useful to solve many mathematical problems, such as calculating the factorial of a number, generating Fibonacci series, etc.

PROGRAM C CODE:

```
#include <stdio.h>
long factorial(int n)
{
   if (n>0)
      return n*factorial(n-1);
   else
      return 1;
}
int main() {
   int n;
   printf("Enter n: ");
   scanf("%d", &n);
   long result = factorial (n);
   printf("\nFactorial of given number is: %ld", result);
   return 0;
}
```

OUTPUT:



```
Enter n: 6
Factorial of given number is: 720
```