DATE: LAB REPORT NO.: **2** SET:
TITLE OF THE PROGRAM: **RECURSIVE FUNCTION (RECURSION)**

OBJECTIVES

I. To understand and implement the recursion technique in user-defined functions.
II. To write programs for generating the Fibonacci series and calculating the factorial of a number using recursion.

REQUIREMENTS

1. C Compiler (e.g., GCC)
2. Computer System
3. IDE or Text Editor
4. OS compatible with the software

THEORY

Recursion is a programming technique where a function calls itself to solve a problem. It involves breaking down a complex problem into smaller, simpler instances of the same problem until a base case is reached. In the context of user-defined functions, recursion allows for the creation of functions that call themselves.

Any function which calls itself is called a **recursive function**, and such function calls are called recursive calls. Recursion involves several numbers of recursive calls. However, it is important to impose a termination condition of recursion.

Recursion cannot be applied to all problems, but it is more useful for tasks that can be defined in terms of similar subtasks. For Example, recursion may be applied to sorting, searching, and traversal problems. Generally, iterative solutions are more efficient than recursion since function call is always overhead.

PROCEDURE (Program Code, Comment, and Output)

1. **Fibonacci Series**

Algorithm:

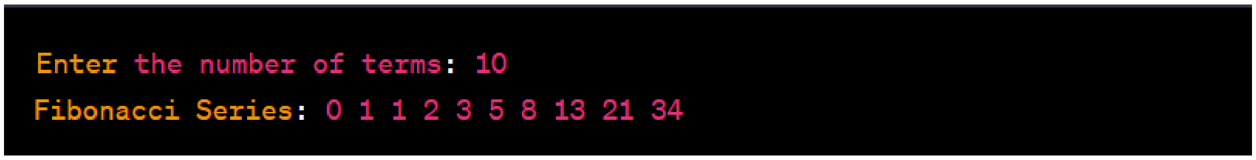
```
  - Start
  - Declare a function `fibonacci(n)` that takes an integer argument `n`.
  - If `n` is 0 or 1, return `n`.
  - Otherwise, return the sum of `fibonacci(n-1)` and `fibonacci(n-2)`.
  - End
```

Program Code:

```
#include<stdio.h>
```

```
int fibonacci(int);

int main()
{
    int n, i;

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

    printf("Fibonacci Series: ");
    for (i = 0; i < n; i++)
    {
        // Call the fibonacci function for each term and print the
result
        printf("%d ", fibonacci(i));
    }

    return 0;
}

// Recursive function to calculate the nth term of the Fibonacci
sequence
int fibonacci(int n)
{
    // Base case: if n is 0 or 1, return n
    if (n == 0 || n == 1)
        return n;
    else
        // Recursive case: calculate the nth term by adding the
previous two terms
        return fibonacci(n - 1) + fibonacci(n - 2);
}
```

Output:



```
Enter the number of terms: 10
Fibonacci Series: 0 1 1 2 3 5 8 13 21 34
```

Explanation:

- The `fibonacci()` function takes an integer `n` as an argument and returns the nth Fibonacci number.
- If `n` is 0 or 1 (base cases), the function returns `n`.
- Otherwise, it recursively calls itself with `n-1` and `n-2` as arguments and returns the sum of the two recursive calls.

- In the `main()` function, the user is prompted to enter the number of terms in the Fibonacci series.
- The `fibonacci()` function is called iteratively for each term and the series is printed.

2. **Factorial of a number**

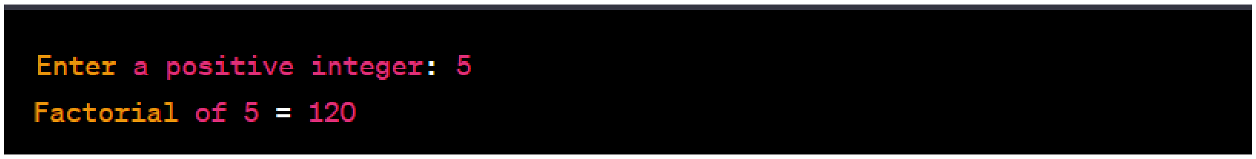
Algorithm:

```
  - Start
  - Declare a function `factorial(n)` that takes an integer argument `n`.
  - If `n` is 0, return 1.
  - Otherwise, return `n` multiplied by `factorial(n-1)`.
  - End
```

Program Code:

```c
#include<stdio.h>

int factorial(int);

int main()
{
    int n, result;

    printf("Enter a positive integer: ");
    scanf("%d", &n);

    // Call the factorial function and store the result
    result = factorial(n);

    printf("Factorial of %d = %d", n, result);

    return 0;
}

// Recursive function to calculate the factorial
int factorial(int n)
{
    // Base case: factorial of 0 is 1
    if (n == 0 || n == 1)
    {
        return 1;
    }
    else
    {
        // Recursive case: multiply the number with factorial of (n-1)
        return n * factorial(n - 1);
    }
}
```

Output:



```
Enter a positive integer: 5
Factorial of 5 = 120
```

Explanation:

- The \`factorial()\` function takes an integer \`n\` as an argument and returns the factorial of \`n\`.
- If \`n\` is 0 (base case), the function returns 1.
- Otherwise, it recursively calls itself with \`n-1\` as the argument and returns the product of \`n\` and the recursive call.
- In the \`main()\` function, the user is prompted to enter a positive integer.
- The \`factorial()\` function is called to calculate the factorial, and the result is printed.

CONCLUSION

Through successful implementation of the recursion technique in user-defined functions, we generated the Fibonacci series and calculated factorials of numbers, effectively solving complex problems by breaking them down into simpler instances and obtaining the expected outputs.