National University
of computer and emerging sciences

Lab 4: Recursion

Items	Description
Course Title	Object Oriented Programming
Lab Title	Recursion
Duration	3 Hours
Operating System/Tool/ Language	Ubuntu/ Eclipse, Gtest / C++
Objective	To get familiar with recursion in C++

Introduction:

Recursion is a powerful programming technique where a function calls itself directly or indirectly to solve a problem. In this lab, we will explore recursion in functions. Recursion provides an elegant solution to many problems, especially those that can be broken down into smaller, similar subproblems.

Recursive Function Structure

A recursive function typically has two main parts:

Base Case: This is the condition that stops the recursion. Without a base case, the function would continue calling itself indefinitely, leading to a stack overflow.
Recursive Case: This is where the function calls itself with a modified argument, moving towards the base case.

How Recursion Works

Initial Call: The function is initially called with a specific set of arguments.
Recursive Calls: The function makes a call to itself with new arguments that bring it closer to the base case.
Base Case: Once the base case is reached, the function returns a result without making further recursive calls.
Returning Results: Each function call returns a result to its caller, eventually culminating in the final result of the initial call.

National University
of computer and emerging sciences

Iterative VS Recursive:

Aspect	Iterative Approach	Recursive Approach
Control Flow	Uses loops (for, while) to repeatedly execute instructions.	Involves function calling itself with smaller input until a base case is reached.
Memory Usage	Generally uses less memory as there's no function call stack.	May use more memory due to function call stack, potentially leading to stack overflow.
Performance	Often more efficient in terms of time and space complexity.	Can be less efficient due to function call overhead and redundant calculations.
Readability	Can be more verbose but follows straightforward logic.	Can be more concise and elegant, especially for problems with a natural recursive structure.
Suitability	Suitable for problems with straightforward iterations.	Suitable for problems with a recursive structure, involving smaller similar subproblems.

Choosing Between Iterative and Recursive Approaches:

Problem Complexity: For simple problems with a linear sequence of steps, an iterative approach may be more suitable. For complex problems with a recursive structure, a recursive approach may be more appropriate.

Performance Requirements: If performance is critical and memory usage needs to be minimized, an iterative solution may be preferred. However, if code elegance and clarity are more important, a recursive solution may be chosen despite potential performance drawbacks.

National University
of computer and emerging sciences

Factorial Calculation:

ITERATIVE:

```
unsigned long long factorialIterative(int n) {
    unsigned long long result = 1;
    for (int i = 1; i <= n; ++i) {
        result *= i;
    }
    return result;
}

int main() {
    int n = 10;
    factorialIterative(n);
    return 0;
}
```

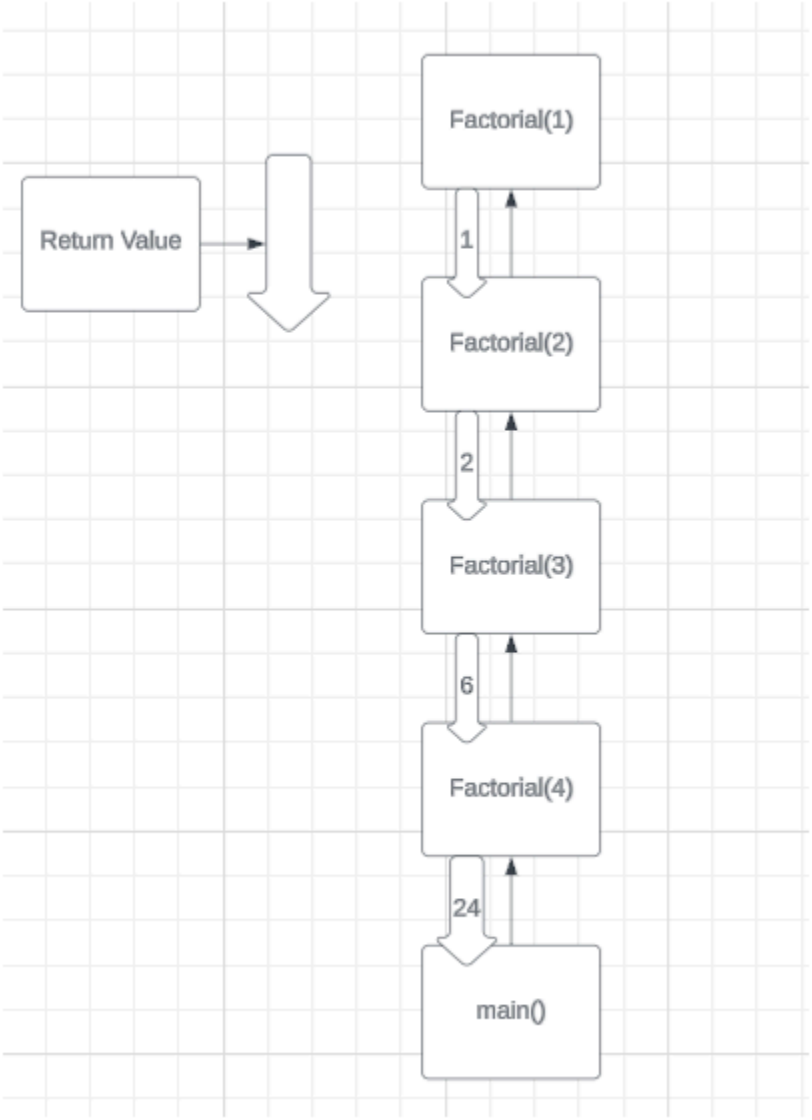
RECURSIVE:

```
unsigned long long factorialRecursive(int n) {
    if (n == 0 || n == 1) {
        return 1;
    } else {
        return n * factorialRecursive(n - 1);
    }
}

int main() {
    int n = 10;
    factorialRecursive (n);
    return 0;
}
```

National University
of computer and emerging sciences

Function Call Stack:



National University
of computer and emerging sciences

Lab Tasks

Task 1

Write a recursive function to calculate the power of the given number.

- Input: 5,3
- Output: 125

int pow(int b,int e)

Task 2

Write a recursive function to produce multiplication of two numbers without using *

int product(int a, int b)

Task 3

Write a recursive function to calculate the sum of the digits of a positive integer n.
int SumOfDigits(int n);

`SumOfDigits(1234)` should return `10` because 1 + 2 + 3 + 4 = 10.

Task 4

Write a recursive function to print the following pattern:

Task 5

Write a recursive function to print the following sequence:
Input: N=16
Output: 16 11 6 1 -4 1 6 11 16