Recursion in C++

# Definition:

Recursion is a programming technique where a function calls itself in order to solve a problem. The recursive function generally has two parts:  
1. Base Case: The condition under which the function stops calling itself.  
2. Recursive Case: The part where the function calls itself with a different argument.

# Example of a Recursive Function:

#include <iostream>  
using namespace std;  
  
int factorial(int n) {  
 if (n <= 1) return 1; // Base case  
 else return n \* factorial(n - 1); // Recursive case  
}  
  
int main() {  
 int num = 5;  
 cout << "Factorial of " << num << " is " << factorial(num) << endl;  
 return 0;  
}

# Advantages of Recursion:

- Simplifies code for problems that can be broken into similar sub-problems.  
- Easier to write and understand for problems like tree traversals, sorting algorithms, etc.

# Disadvantages of Recursion:

- Can lead to excessive memory use and stack overflow if not implemented correctly.  
- Generally slower than iterative solutions due to function call overhead.

# Questions

## Easy

1. Factorial Calculation  
- Write a recursive function to calculate the factorial of a given number.

2. Fibonacci Series  
- Write a recursive function to generate the nth Fibonacci number.

## Medium

3. Sum of Digits  
- Write a recursive function to find the sum of digits of a number.

4. Reverse a String  
- Write a recursive function to reverse a given string.

## Hard

5. Towers of Hanoi  
- Write a recursive function to solve the Towers of Hanoi problem.

6. Permutations of a String  
- Write a recursive function to generate all permutations of a given string.