Recursion

A **recursion function** in C++ is a function that **calls itself** directly or indirectly in order to solve a problem. It typically solves a smaller version of the problem at each step, and must have a **base case** to stop the recursion and prevent infinite loops.

Why Use Recursion?

Easier to solve problems that have recursive structure (like tree traversal, factorial, Fibonacci). But recursion can be **inefficient** if not handled carefully (like in Fibonacci), and can cause **stack overflow** if too deep.

Basic Structure of a Recursive Function:

```
returnType functionName(parameters) {
    if (base_case_condition) {
        // stop recursion
        return base_case_value;
    } else {
        // recursive call
        return functionName(smaller_problem);
    }
}
```

Example 1: Factorial of a Number

```
#include <iostream>
using namespace std;

int factorial(int n) {
```


```
    if (n == 0)  // base case
        return 1;
    else
        return n * factorial(n - 1);  // recursive call
}

int main() {
    int num = 5;
    cout << "Factorial of " << num << " is " << factorial(num);
    return 0;
}
```

Example 2: Fibonacci Series

```
#include <iostream>
using namespace std;

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 = 6;
    cout << "Fibonacci of " << n << " is " << fibonacci(n);
    return 0;
}
```

Example 3: Sum of Natural Numbers

```
#include <iostream>
using namespace std;

int sum(int n) {
    if (n == 0)
        return 0;
    else
        return n + sum(n - 1);
}

int main() {
    int n = 10;
    cout << "Sum of numbers from 1 to " << n << " is " << sum(n);
    return 0;
}
```

10 + sum(9)
10 + (9 + sum(8))
10 + (9 + (8 + sum(7)))
...
10 + 9 + 8 + 7 + 6 + 5 + 4 + 3 + 2 + 1 + sum(0)
10 + 9 + 8 + 7 + 6 + 5 + 4 + 3 + 2 + 1 + 0