Recursion in C++

1. Definition

Recursion is a programming technique where a function calls itself to solve a smaller instance of the same problem.
Key components:

- **Base case**: the condition under which the recursion stops.
- **Recursive case**: the part where the function calls itself with a simpler or smaller argument.

2. How It Works

Each recursive call is pushed onto the call stack. When a base case is reached, the stack unwinds, returning results back up the chain.

3. Example 1: Factorial

Compute $n!$ defined by

$$n! = \begin{cases} 1, & n \leq 1, \\ n \times (n-1)!, & n > 1. \end{cases}$$

```cpp
// factorial.cpp
#include <iostream>

unsigned long factorial(unsigned int n) {
    if (n <= 1)                // base case
        return 1;
    return n * factorial(n-1); // recursive case
}

int main() {
    std::cout << "5! = " << factorial(5) << "\n";
    return 0;
}
```

4. Example 2: Fibonacci (Naïve)

Compute the nth Fibonacci number:

$$F(n) = \begin{cases} 0, & n = 0, \\ 1, & n = 1, \\ F(n-1) + F(n-2), & n > 1. \end{cases}$$

```
// fibonacci.cpp
#include <iostream>

unsigned int fib(unsigned int n) {
    if (n < 2)                // base case
        return n;
    return fib(n-1) + fib(n-2); // recursive case
}

int main() {
    std::cout << "F(6) = " << fib(6) << "\n";
    return 0;
}
```

5. Notes and Pitfalls

- **Infinite recursion**: missing or incorrect base case leads to stack overflow.
- **Performance**: naïve recursion (e.g., Fibonacci) may have exponential time complexity.