Assignment No. 1.1	
Using C++ for Recursion	
Course Code: CPE010	**Program:** Computer Engineering
Course Title: Data Structures and Algorithms	**Date Performed:8/12/25**
Section:CPE21S4	**Date Submitted:8/12/25**
Name(s):Crishen Luper S. Pulgado	**Instructor: Jimlord Quejado**
Output	

Recursive:

main.cpp — Share — Run — Output

```cpp
#include <iostream>

int sumNumbers(int arr[], int size){
    if (size == 1) return arr[0];
    return arr[size - 1] + sumNumbers(arr, size - 1);
}

int main() {
   int listNumbers[] = {1,2,3,4,5,6};
   int size = sizeof(listNumbers) / sizeof(listNumbers[0]);
   std::cout<< "The sum is: " << sumNumbers(listNumbers, size) << "\n";

    return 0;
}
```

```
The sum is: 21


=== Code Execution
```

- The “if (if (size == 1) return arr[0];” runs in O(1) time. One element is processed at a time, followed by calling itself with size – 1. Then, we have n calls each doing O(1) work and O(n) total. For the space complexity, every recursive call is on the call stack until it finishes O(n) extra memory.

Non-recursive

main.cpp — Share — Run — Output

```cpp
#include <iostream>

int sumNumbers(int arr[], int size){
    int sumNumbers = 0;
    for (int i = 0; i < size; i++) {
        sumNumbers += arr[i];
    }
    return sumNumbers;
}

int main() {
   int listNumbers[] = {1,2,3,4,5,6};
   int size = sizeof(listNumbers) / sizeof(listNumbers[0]);
   std::cout<< "The sum is: " << sumNumbers(listNumbers, size) << "\n";

    return 0;
}
```

```
The sum is: 21


=== Code Execution S
```

- The time complexity is O(n) since loop runs n times. The space complexity is O(1) since no recursion.

Recursive:

main.cpp

```cpp
#include <iostream>

int fibonacci(int n) {
    if (n == 0) return 0;
    if (n == 1) return 1;

    return fibonacci(n - 1) + fibonacci(n - 2);
}

int main() {
    int n = 10;
    std::cout << "Recursive Fibonacci (" << n << ") = "
              << fibonacci(n) << "\n";
}
```

Output

```
Recursive Fibonacci (10) = 55


=== Code Execution Successful =
```

- Since every call makes a 2 or more calls, then the time complexity is O(2^n). The space complexity is O(n) since it is a recursion

Non-recursive:

main.cpp

```cpp
#include <iostream>

int fibonacci(int n) {
    if (n == 0) return 0;
    if (n == 1) return 1;

    int a = 0, b = 1, next;
    for (int i = 2; i <= n; i++) {
        next = a + b;
        a = b;
        b = next;
    }
    return b;
}

int main() {
    int n = 10;
    std::cout << "Iterative Fibonacci (" << n << ") = "
              << fibonacci(n) << "\n";
```

Output

```
Iterative Fibonacci (10) = 55


=== Code Execution Successful ==
```

- It runs n times in the loop, which is the time complexity is O(n). The space complexity is O(1) since it uses only few variables.