



```
1 #include <vector>
2 using namespace std;
3
4 // Task 1: Sum of a List of Numbers
5
6 // Recursive version
7 int sumRecursive(const vector<int>& nums, int n) {
8     if (n == 0) return 0;
9     return nums[n - 1] + sumRecursive(nums, n - 1);
10 }
11
12 // Iterative version
13 int sumIterative(const vector<int>& nums) {
14     int total = 0;
15     for (int num : nums) {
16         total += num;
17     }
18     return total;
19 }
20
21 // Task 2: Fibonacci
22
23 // Recursive version
24 int fibonacciRecursive(int n) {
25     if (n <= 1) return n;
26     return fibonacciRecursive(n - 1) + fibonacciRecursive(n - 2);
27 }
28
29 // Iterative version
30 int fibonacciIterative(int n) {
31     if (n <= 1) return n;
32     int a = 0, b = 1, c;
33     for (int i = 2; i <= n; ++i) {
34         c = a + b;
35         a = b;
36         b = c;
37     }
38     return b;
39 }
40
41 // Main
42 int main() {
43     vector<int> nums = {1, 2, 3, 4, 5};
44     cout << "Sum Recursive: " << sumRecursive(nums, nums.size()) << endl;
45     cout << "Sum Iterative: " << sumIterative(nums) << endl;
46
47     int n = 10;
48     cout << "Fibonacci Recursive (" << n << ") = " << fibonacciRecursive(n) << endl;
49     cout << "Fibonacci Iterative (" << n << ") = " << fibonacciIterative(n) << endl;
50
51     return 0;
52 }
```

```
Sum Recursive: 15
Sum Iterative: 15
Fibonacci Recursive (10): 55
Fibonacci Iterative (10): 55

Process exited after 0.0926 seconds with return value 0
Press any key to continue . . .
```

```
Compiler: g++
Output: 0
Working: 0
Output File: C:\Users\joshua\Documents\09.exe
Output Size: 3.2294242237548 KB
Compilation Time: 0.41s
```

Explanation:

The Big-O-Notation shows how quick and how much memory an algorithm uses as the input gets larger, focusing on the worst-case-scenario.

Task 1: Summing a List of Numbers

- In the recursive version, the time complexity is $O(n)$ because the function runs once for each item in the list, But the space complexity is $O(1)$ because it uses only a fixed amount of memory.

Task 2: Fibonacci Sequence

- The recursive version, the time complexity is $O(2^n)$ because each number calls the function twice.
- The non recursive version or Iterative version since it loops through numbers once, and the space complexity is $O(1)$ as it only uses one or two variables.