C20 Lab Experience: Exploring Recursion Through the Fibonacci Sequence

Introduction:

Recursion is a powerful programming technique in which a function calls itself to solve smaller subproblems. Many problems can be elegantly solved using recursion, but it's essential to understand how it works to avoid issues like infinite loops and stack overflows.

The Fibonacci sequence is a classic example used to demonstrate recursion. The sequence starts with 1 and 1, and each subsequent number is the sum of the two preceding ones (e.g., 1, 1, 2, 3, 5, 8, ...). It can be defined mathematically as:

```
F(0) = 1

F(1) = 1

F(n) = F(n − 1) + F(n − 2), for n > 1
```

This definition naturally lends itself to a recursive implementation.

Step 1:

Write a recursive function that implements the definition above. The function should have the following signature:

```
int fib1(int n);
```

Step 2:

Write a function that uses fib1 to print the first 10 Fibonacci numbers.

Step 3:

Modify your function to print the first 50 Fibonacci numbers.

- What do you observe in terms of performance?
- Can you comment on the accuracy of the output?

Step 4:

Improve performance using **memoization**. Write a second version of the recursive Fibonacci function with the following signature:

```
int fib2(int n, map<int, int>& memo);
```

Step 5:

Use fib2 to generate the first 60 Fibonacci numbers.

- What changes do you notice in performance?
- What can you say about the accuracy of the results?

Step 6:

Download and install the **Boost** library from:
http://www.boost.org

Step 7:

Write a third version of the Fibonacci generator function using Boost's arbitrary-precision integer type (cpp_int). The function signature should be:

```
cpp_int fib3(cpp_int n, map<cpp_int, cpp_int>& memo);
```

- What can you say about the speed and accuracy of this version?