Assignment 11b

Implement problem A, B, and C below in Java and submit by tomorrow:

Submit only the source program, no .class files or anything else; it can all be in one .java file or in three separate files. Along with the source code, a status report (how much was finished of A, B, C) should be submitted that includes answers to the questions (1, 2, and 3) below.

A. Write a Java program to implement a recursive version of the following function to compute an element of the Fibonacci sequence which is defined as follows (include a counter of the number of recursive calls made):
Fib(0) = 0
Fib(1) = 1
Fib(n) = Fib(n-1) + Fib(n-2)

B. Implement a memoized version of Fib. In the memoized version, you can store the elements of the sequence in any data structure with efficient access capability. Include a counter of the recursive calls made by your algorithm.

C. Implement Fib(n) that uses a two element array (or two variables) to store the previously computed values (if you have time).

1. How many recursive calls are made by the **non-memoized** brute force version for computing Fib(30)?

2. How many recursive calls are made by the **memoized** version for computing Fib(30)?

3. Compare the two versions (based on the resulting counts). Briefly explain why the brute force algorithm needs to use some form of dynamic programming?

B. Implement in Java an inefficient, brute force, recursive algorithm to compute the Longest Common Subsequence of two strings (LCS); base this on the recursive equations given in the lecture. Copy this implementation and memoize it. Run both implementations on small input sequences and compare the running times (include a counter to count basic operations and print it after computing both versions of LCS). Run your program on larger strings and note the number of recursive calls in both versions. Base your solution on the lecture notes, but make sure you understand the details and why and how the algorithms work for finding the LCS of two strings.