CS146: Quiz 9
Due Tuesday, April 11, at 7:00AM
10 points

You will solve two dynamic programming problems each in two ways (using the top-down strategy (memoization) and the bottom up strategy) **To get started, import the starter file, Fibonacci.java and MinSumPath into the dynamic package you create in a new Java Project.** Please do not change any of the method signatures in the class. Implement the methods described below. You are free to test your code however you prefer. These methods should be completed individually using any IDE you are comfortable with. You are free to use the textbook, slides, class notes, and the Java API Documentation, but **DO NOT** consult any other resources.

Calculating the Fibonacci Numbers

Below is the formula to compute Fibonacci Numbers. Note that both methods should work correctly for any integer n such that $0 \leq n \leq 92$

$Fib_0 = 0$
$Fib_1 = 1$
$Fib_n = Fib_{n-1} + Fib_{n-2}\ for\ n \geq 2$

`public static long fibMemo(int n)`
This method will calculate the nth Fibonacci number using the top down strategy. Note this method MUST BE recursive and you will need to create a recursive helper method.

`public static long fibBottomUp(int n)`
This method will calculate the nth Fibonacci number using the bottom up strategy. Note this method CANNOT be recursive and you should not create any additional helper functions.

CONTINUES ON NEXT PAGE

Calculating the Minimum Sum Path in a Triangle (LeetCode Problem)

Given a `triangle` array, return the minimum path sum from top to bottom. For each step, you may move to an adjacent number of the row below. More formally, if you are on `index i` on the current row, you may move to either `index i` or `index i + 1` on the next row.

Example 1

```
Input: triangle = [[2],[3,4],[6,5,7],[4,1,8,3]]
Output: 11
Explanation: The triangle looks like:
   2
  3 4
 6 5 7
4 1 8 3
The minimum path sum from top to bottom is 2 + 3 + 5 + 1 = 11 (underlined above)
```

Example 2:

```
Input: triangle = [[-10]]
Output: -10
```

Note that `triangle[0]` has length 1 and `triangle[i].length = triangle[i-1].length+1`. Your methods should work correctly for any input where the triangle length is between 1 and 200 (inclusively). The triangle can contain numbers between -10000 and 10000.

`public static int minSumPathMemo(int[][] triangle)`
This method will calculate the minimum sum path in the triangle using the top down strategy. Note this method MUST BE recursive and you will need to create a recursive helper method.

`public static int minSumPathBottomUp(int[][] triangle)`
This method will calculate the minimum sum path in the triangle using the bottom up strategy. Note this method CANNOT be recursive and you should not create any additional helper functions.

Extra Challenge: Could you do this using only `O(n)` extra space where n is the total number of rows in the triangle?

Submission

Please create a jar or zip file of your project. It should include the java files (Fibonacci.java and MinSumPath.java) and submit it on Canvas. If your file includes class files or other extraneous files outside of the package folder and java files, you will receive an automatic one point deduction. It is important that you learn how to correctly submit your assignment.