CPW 143 Fall 2017, Chapter 12 Extra Credit Programming Project 9

The game of "Jump It" consists of a board with *n* positive integers in a row except for the first and last columns which always contain zero. These numbers represent the cost to enter each column. Here is a sample board where *n* is 9:

Column	0	1	2	3	4	5	6	7	8
Value	0	28	57	60	36	39	35	91	0

The object of the game is to move from the first (0^{th}) column to the last (n^{th}) column in the lowest total cost. The number in each column represents the cost to enter each column. Always start the game in the first column and have two types of moves. You can either move to the adjacent column or jump over the adjacent column to land two columns over. The cost of the game is the sum of the costs of the visited columns.

In the board shown above, there are several ways to get to the end. Starting in the first column, our cost so far is 0. The greedy way to choose a path is to always choose whichever column has the lowest cost. For the above board, we would jump to 28, then jump to 57, then move to 36, then jump to 35, and finally to 0 for a total cost of 28 + 57 + 36 + 35 = 156. However, there is a way to get a lower cost: jump to 57, jump to 36, jump to 35, and finally to zero. The cost then would be 57 + 36 + 35 = 128.

Another example:

Column	0	1	2	3	4	5	6
Value	0	40	80	26	31	65	0

The lowest cost path is 0, 80, 26, 31, 0 for a total cost of 80 + 26 + 31 = 137.

The general concept for finding the lowest total cost is to start in the first column (with a cost of 0) and choose as your next step, either column 1 or column 2, which ever will give the lowest cost with whatever is possible from that position. In the above example, recursively, the lowest cost is either (40 + lowest cost when starting in column 1) or (80 + lowest cost when starting in column 2).

Your job is to complete the recursive method **`findLowestCost( int[] board, int index )`**. Submit to Canvas the entire class **`JumpItGame.java`**.

Here are some sample runs of the program:

```
For the board [0, 42, 49, 88, 67, 35, 2, 0]
Greedy algorithm says cost is: 160
and the actual lowest cost is: 118

For the board [0, 78, 47, 50, 72, 14, 6, 47, 60, 0]
Greedy algorithm says cost is: 164
and the actual lowest cost is: 158

For the board [0, 3, 8, 99, 0]
Greedy algorithm says cost is: 11
and the actual lowest cost is: 8

For the board [0, 58, 91, 94, 28, 5, 75, 100, 25, 0]
Greedy algorithm says cost is: 282
and the actual lowest cost is: 219
```

```java
import java.util.*;

public class JumpItGame {
    public static void main( String[] args ) {
        Random rand = new Random();

        int[] board = new int[ 5 + rand.nextInt( 6 ) ];
        board[ 0 ] = 0;
        board[ board.length - 1 ] = 0;
        for ( int i = 1; i < board.length - 1; i++ ) {
            board[ i ] = 1 + rand.nextInt( 100 );
        }

        System.out.println( "For the board " + Arrays.toString( board
) );
        System.out.println( "Greedy algorithm says cost is: "
                            + greedy( board ) );
        System.out.print( "and the actual lowest cost is: "
                            + findLowestCost( board ) );
    }

    public static int greedy( int[] board ) {
        int cost = 0;
        int position = 0;
        while ( position < board.length - 2 ) {
            if ( board[ position + 1 ] < board[ position + 2 ] ) {
                cost += board[ position + 1 ];
                position += 1;
            }
            else { // move two columns
                cost += board[ position + 2 ];
                position += 2;
            }
        }
        return cost;
    }

    public static int findLowestCost( int[] board ) {
        return findLowestCost( board, 0 );
    }
    private static int findLowestCost( int[] board, int index ) {
        // TODO: YOUR CODE GOES HERE.

        // TODO: the following line is here only to make the
        // starter code compile. You need to replace it.
        return 0;
    }
}
```