HW2

Due: Fri. May 29th by 11:55 pm

This is an individual assignment. Please review the rules on academic misconduct on the course D2L page. Here is what it boils down to:
DO NOT share your answers/code with others in any way (in person or electronically); DO NOT ask others to provide you with answers/code; write up the solutions/code yourself and clearly cite any external sources (books, websites, forums, discussions with peers etc.) that helped you arrive at your solutions.
If you have any doubts, please contact the instructor.

This homework assignment consists of two parts. The first part consists of a short coding exercise that will give you additional practice with Java generics, linked lists and sorting. The second part consists of problems that will give you more practice working with linear structures, recursion and recurrence relationships.

Both parts are to be submitted via gradescope. The coding part will be autograded whereas the written part will be graded manually.

Part I - Coding

(15 points)

In this exercise, you are asked to write code to sort a *singly linked list* using bubble sort. You will also get additional practice with generic classes and methods.

We went over bubble sort. In the version of bubble sort we discussed, we go over the array from right (higher indices) to left (lower indices) in each pass and swap adjacent elements that are out of order. During each pass the smallest element bubbles to the left to its appropriate spot. Think about the changes that are needed if, in each pass, we scan the array from left to right and the largest element bubbles to the right to its appropriate spot. Now think about what we would need to do if, instead of an array, we were using a singly linked list, and in each pass, we scanned the list from left to right and swapped adjacent elements that are out of order.

The following files are provided for this exercise:

1. `SLLNode.java` - Represents a node in a singly linked list.
2. `SLL.java` - A class that encapsulates a singly linked list. Helper methods are provided to add and remove elements, and to query the size.
3. `BubbleSortInterface.java` - An interface that specifies the signature of the bubble sort method that your class should implement.
4. `SLLBubbleSort.java` - This is the class that you have to implement. The provided file consists of the class skeleton that you need to fill in.

Your implementation should only rely on the provided classes. You are not allowed to use any classes from the Java collections framework. Test your program with lists consisting of different types of `Comparable` elements (Integers, Doubles, Strings etc.).

When you submit your class, the autograder will run some unit tests on it and grade it based on functionality. Please pay attention to the packaging structure of these classes and do not alter it or you will run into issues with the autograder on gradescope.

Submission and Grading

- Please submit via gradescope to the assignment entitled 'HW2_Coding'.
- *Please only submit the file* `SLLBubbleSort.java` *ensuring that the package declaration* `hw2.student` *is intact. Submit the file directly, do not compress it into an archive.*
- Once submitted, the autograder will compile and test your implementation providing you with output that shows the test results.
- You may submit multiple times. Only the most recent submission will be kept.
- You will be graded primarily based on functionality as tested by the autograder. The autograded score is subject to adjustments at the discretion of the marker.

Part II - Written

1. (5 points)
   Provide pseudocode or Java code for a recursive version of selection sort that operates on an array of n elements. Use your pseudocode/code to set up a recurrence relation for the total number of comparisons performed. Solve the recurrence relation to determine a Θ asymptotic characterization of selection sort.

2. (4 points)
   The Catalan numbers are defined recursively as:

$$C_0 = 1,$$
$$C_{n+1} = \sum_{i=0}^{n} C_i C_{n-i}, \quad n \geq 0.$$

   Use this definition to determine C_3. Illustrate your answer using a recursion call tree assuming a naïve recursive implementation.

3. (3 points)
   Use iterative substitution to determine a closed-form solution to the recurrence relation:
   $T(n) = 2T(n-1) + 1$, (for $n > 1$),
   $T(1) = a$, where $a > 0$.

4. (3 points)
   Use induction to show that the total number of additions performed when computing the nth Fibonacci number F_n recursively using a naïve implementation is $F_{n+1} - 1$.

Submission

Please handwrite or typeset your solutions to these problems. If hand-writing, please write neatly and make sure that your responses are clearly visible in your scanned/photographed images.

Please submit via gradescope to the assignment entitled 'HW2_Written'. Gradescope will accept both PDF and image file formats. Allow yourself plenty of time before the deadline to ensure a smooth submission process.