HW2

Due: Sun. Jun. 3rd by 11:55 pm

This is an individual assignment. Please review the rules on academic misconduct on the course D2L page.

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

Part I

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

2. (5 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. (5 points)
   Determine an asymptotic upper bound for the following recurrence relations. Assume that the cost of the base case is a constant.

   - $T(n) = 2T(n-1) + 1$
   - $T(n) = T(n-1) + T(n/2) + n$

Part II

1. (10 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 class. In the class version of bubble sort, 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.

   Two files are provided for this exercise:

`SLLNode.java` - Represents a node in a singly linked list.
`SLL.java` - A class that encapsulates a singly linked list. Helper methods are provided to add and remove elements, and to query the size.

Implement a generic static method called `BubbleSort` that takes a singly linked list as an input and sorts it *in place* using bubble sort. Your method should have the following signature:

```
public static <T extends Comparable<? super T>>
    void BubbleSort( SLL<T> list )
```

Please encapsulate your method in a class called `SLLBubbleSort`.

Your implementation should only rely on the classes `SLLNode.java` and `SLL.java`. 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.).

2. (5 points)
A prefix expression is one where the operator appears before the operands. Like the postfix notation, it is also parentheses-free. The order of evaluation is determined by the order in which the operators and the operands appear. For example:
`+ * 3 2 6` = `+ 6 6` = `12`
`+ - 2 * 3 4 5` = `+ - 2 12 5` = `+ -10 5` = `-5`

Write a recursive method in Java that evaluates a prefix expression. Your method should be encapsulated in a file called `Prefix.java` that has the following class structure:

```
public class Prefix {
  public static int evaluate( Scanner sc ) {
    // your recursive evaluation code
  }
}
```

Please note that you are required to use recursion for the evaluation. You can assume that the provided expression will be in valid prefix notation.

Submission Instructions

Please pay attention to the following when preparing your work for submission. The TAs will test your code during tutorials after the due date. Please ensure that your classes and methods exactly follow the naming specifications.

Part I

Please prepare a file called `part1.pdf` (scanned or typeset) that is packaged with the rest of the files for Part II.

Part II

Please package your Java class files in a compressed archive called `hw2.zip`. Please remove any driver programs (main methods) that you used to test your implementations. Please make sure that your name appears at the beginning of each file. Please do not place the files in different folders. Your archive should only contain the following Java files: `SLLNode.java`, `SLL.java`, `SLLBubbleSort.java` and `Prefix.java`. Please also include the file `part1.pdf` in your archive.