Homework 3

Due: 4:00pm on November 14, 2024 (Thursday)

Objectives:

- Implement your own Singly Linked List from scratch using recursion.
- Solidify your understanding of recursion.
- Have a better insight about Comparable and compareTo in Java.
- Take an interface and implement it on your own.

In homework 1, you implemented the `MyArray` class to store words parsed from a text file.

In this homework, you will accomplish very similar results as you had from the homework 1 such as adding new words, getting the current size of the list, removing words and displaying the current values but you are to implement a different Data Structure class and use it.

```
public class SortedLinkedList implements MyListInterface
```

Your `SortedLinkedList` class must have the following properties.

- It must implement `MyListInterface` that is provided to you.
- It does not allow any duplicate at any point. It needs to check duplicates as a new word is being added.
- It is a linked structure of nodes (Singly) and each node's data type is String.
- It is **always** (or at any point) sorted (ordered) in an ascending order. (For example, when you add "d", "c", "a" and "b" sequentially into the `SortedLinkedList` and then call the display() method of the list, you should see **[a, b, c, d]** as a result. The structures of the list should be ordered! Also, please refer to `HW3Driver.java` file to check the correct output of display() method.)

In addition to the properties, methods in the `SortedLinkedList` must USE recursion. In other words, there should be NO loops in your `SortedLinkedList` implementation, not even in its constructors. **Once again, NOT a single loop is allowed**.

Also, you are not allowed to have any other fields in `SortedLinkedList` class except the head reference as follows. Do not use Generics since the type of data in node is String.

```
private Node head;
```

`MyListInterface` interface is where you can find necessary information about the methods to be implemented.

Along with a constructor with no parameters, the `SortedLinkedList` should have **another constructor that takes unsorted String array as a parameter and build a new SortedLinkedList** as you can see from the example of calling this constructor in the `HW3Driver.java.` ***Do not use any sorting algorithm on the input array!!*** The add method of the `SortedLinkedList` should take care of adding each string into the right position in the list so that the list stays ordered.

You should see the same output in the expected output comment in `HW3Driver.java` file when you run the `HW3Driver` with the childrensbible.txt file provided to you. When running the HW3Driver with the childrensbible.txt file, put the text file into the project folder, not in the src folder (if you use Eclipse). The display method should print all the values in one line and each word should be comma separated (After each comma, put a space too to make it easy to read. Please refer to example output in the driver file) except the last word. If the list is empty, print square brackets (opening and closing with no spaces in them). Pay attention to how you implement the display method. It should be efficient. Hint: think about lab 1.

You are NOT allowed to use anything from the Collections Framework. Also, there should be NO imports in your code.

Deliverables:

- Your source code file. (Make sure to include your Andrew ID and NAME in the header comments as an author tag.) Submit your source code file (`SortedLinkedList.java`) using Autolab (https://autolab.andrew.cmu.edu). Do not zip it! Do not use package.
    - Using either single-line or multi-line comments, write how your recursion works for any method where recursion is used. (For example, what the base case is and what the recursive case is.)

Grading:

Autolab will grade your assignment as follows.

- Working code: 90 points
- Coding conventions: 10 points
    - We'll deduct one point for each coding convention issue detected.

Autolab will show you the results and scores of its grading within approximately a few minutes of your submission. You may submit multiple times so as to correct any problems with your assignment. Autolab uses the last submission as your grade.

The Autolab score is not final. The TAs will look into your source code to check correctness and design and deduct points accordingly. The most important criterion is always correctness. Buggy code is useless (even if you may think a found bug is very minor). It is also critical that your code is efficient and follows the specifications properly. Additionally, it should be readable and well organized. This includes proper use of clear comments. Points will be deducted for poor design decisions and unreadable code, etc.

As mentioned in the syllabus, we will be using the Moss system to detect software plagiarism. Make sure to read the cheating policy and penalty in the syllabus. *Any cheating incident will be considered very seriously. The University also places a record of the incident in the student's permanent record.*

Late submissions will NOT be accepted and, if you have multiple versions of your code file, make sure you do **submit the correct version.**