CS 201: Data Structures
Homework 10

Layla Oesper
Fall 2017

This assignment is due by **10pm on Friday October 27** and is worth 14 points.

1 Goals

The goal of this assignment is to get more practice with recursion.

2 Setup and Requirements

This is a solo assignment. You may certainly discuss the assignment and how to get to your final solutions with your classmates, but you must write your own solution. Also, make sure to cite any sources used or classmates consulted!

You may need to spend more time thinking about this problem and how to approach it than writing code. I strongly encourage you to work with paper and pencil first. The total amount of code you will need to write, is relatively small.

3 Background Information: The Comparable Interface

The Comparable Interface is an interface in Java that allows for there to be a total ordering of the objects in each class that implements it. In particular, any class that implements the `Comparable` interface, must implement a `compareTo` method. Here is a description of how the `compareTo` method should work.

Method	**return type**	**description**
`compareTo(E obj)`	int	This method will compare the object calling the method to the object passed as a parameter (this object should be of the same type as the object calling the method). Returns a negative integer, zero, or a positive integer if this object is less than, equal to, or greater than the specified object.

A few examples of classes we have seen that implement the comparable interface are: String, Integer, Double and Calendar. Consider the following snippet of code:

```java
Integer one = new Integer(1);
Integer two = new Integer(2);
int comp = one.compareTo(two);
System.out.println(comp);
```

This would print a negative number (-1) since 1 is less than 2. Similarly, consider the following snippet of code:

```java
String one = "one";
String two = "two"
```

```
int comp = two.compareTo(one);
System.out.println(comp);
```

This will print a positive number (5) since "two" comes later alphabetically (hence is greater) than "one".

4 Code Provided to You

As part of this assignment I'll provide to you a class called `RecursiveLinkedList.java` that implements the List ADT (also provided in the form of an interface `List.java`) using a LinkedList structure. I've also modified this code to only work with items that implement that Comparable interface. You should spend some time just looking through this code. In addition to the public methods outline in the interface `List.java`, I've included a number of private helper methods. Take note of how these helper methods can make some of my code for other methods (e.g. `public E remove(int index)`) much cleaner and easier to understand what the different cases are. All of this code is largely taken straight from you textbook, but it's nice to see it all in once place.

5 Your Task

Your task with this assignment will be to write the code to add the following three methods to this class. To receive full credit, **you must implement each of these methods using recursion** and your implementation of each must run in $O(n)$ where n is the number of elements in the list.

```
/**
 * Return the maximum element in the list using compareTo() method
 * of Comparable
 *
 * @return maximum element of the list
 **/
public E max(){
    //TODO: complete this method recursively
    // I strongly suggest you use a helper function
    return null;
}

/**
 * Remove all elements that match e using the equal() operator
 * to determine a match
 *
 * @param e The element that should be removed
 **/
public void remove(E element){
    //TODO: complete this method recursively
    // I strongly suggest you use a helper function
    return;
}
```

```
/**
 * Duplicate each element of the list
 *
 * For example, the list [ 0 1 2 ] duplicated becomes [ 0 0 1 1 2 2 ]
 **/
public void duplicate(){
    //TODO: complete this method recursively
    //I strongly suggest you use a helper function
    return;
}
```

6 Hints

- I suggest you take my suggestion about using a helper method seriously. The key thing here will be to think about how to define the signature of that method.
- You will likely want to define helper methods that either take a `Node` object as a parameter or return a `Node` object. If you do this, you should make sure that your method declaration (the first line that says what you return and the parameter types) always uses `Node<E>`. For example, you might define the following method that returns a Node and takes in a Node as a parameter:

  ```
  public Node<E> helperFunction(Node<E> aNode)
  ```

- It's totally fine if all the recursion is done in your helper methods, rather than the specific methods I am asking you to implement.
- You will not be evaluated on it, but I suggest adding code to the main method that tests out your methods to ensure that they are implemented correctly.

7 Submission and Grading

Please submit your modified version of `RecursiveLinkedList.java`. This assignment is worth 14 points. Each of the three functions you will implement is worth 4 points. To get the full points, it needs to be implemented recursively (no loops are needed) and it needs to run in $O(n)$. The more elegant your code, the better. Your code will also be evaluated for general style guideline (are you using good variable names, using proper indentation, etc). This will be worth 2 points.

Start early, ask lots of questions, and have fun! Layla, the lab assistants, and the prefect are all here to help you succeed - don't hesitate to ask for help if you're struggling!