COMP 2150 – Spring 2014
Homework 7: Linked Lists

(Posted Apr. 7 – Due Apr. 14 by 12:40 pm)

Remember that as stated on the syllabus, this assignment should be an individual effort (contact me or visit the Computer Science Learning Center, www.cs.memphis.edu/cslc, if you need help).

Submission: Please zip your source code into a single file (you can zip the entire project folder if you're using BlueJ) and upload it to the proper folder in the eCourseware dropbox at https://elearn.memphis.edu. The dropbox will cut off all submissions after the indicated deadline, so don't wait until the very last minute to submit your work!

Grading: This assignment will be graded by the TA Rahul Vemuri (rvemuri@memphis.edu). If you have questions or concerns about your grade, please contact him first. I'll be happy to look over your assignment myself if he's not able to resolve the situation to your satisfaction. Also remember that as stated on the syllabus, all submissions MUST compile and run to receive credit. The TA does not have time to find and correct your syntax errors!

Coding Style: Be sure to follow the good coding practices that were discussed in COMP 1900:

- Use descriptive variable and method names. Avoid using single-character names unless it's very obvious what the variable's being used for (like a loop counter).
- Follow standard Java programming conventions for **variableAndMethodNames**, **ClassNames**, **CONSTANT_NAMES**.
- Consistently indent your code.
- Use comments judiciously. Every source code file should include a comment block at the top that lists your name and the assignment number. Every non-trivial method should have a comment preceding it that specifies what the method does, what its parameters are, and what it returns. For simple methods like accessors or mutators this isn't necessary, but it doesn't hurt to put in a simple comment like **`// accessor methods`**. I also expect to see comments throughout your code explaining what actions are being taken!

Add the following methods to the **LinkedList<E>** class that we wrote during lecture (code posted on Apr. 7). You may call the existing methods in **LinkedList<E>** if you want, but do not use anything from the built-in **java.util.LinkedList** class!

1. **(2 pts)** Write a new method named **clear()** that removes all elements from the list.

2. **(5 pts)** Write a new method named **remove(E item)** that removes and returns the first item in the list that is equivalent to the specified object. If the list does not contain such an item, the method should return **null**.

3. **(6 pts)** Write a new method named **reverse()** that reverses the order of the nodes in the calling list. Can you figure out a way of doing this non-recursively in $O(n)$ time?

4. **(6 pts)** Write a new method named **toArrayList()** that returns an **ArrayList<E>** object containing all elements in the calling list, in the same order (i.e., the head node's data should be stored in index 0 of the returned array list).

5. **(6 pts)** Write a new method named **slice(int beginIndex, int endIndex)** that returns a new **LinkedList<E>** object containing the elements of the calling list between **beginIndex** (inclusive) and **endIndex** (exclusive). The calling list should not be modified. This method should throw an **IndexOutOfBoundsException** if an invalid index is supplied, or if **beginIndex** is not at least 1 less than **endIndex**.

As usual, test your methods thoroughly once you've written them! Be especially careful about special cases such as empty lists or lists with only one node. Your code should not be throwing any **NullPointerExceptions**.