CS 0445 Spring 2022
Recitation Exercise 4

Introduction:

Recently in lecture we have discussed the list interface and the author's LList<T> class. In this recitation you will add some useful functionality to the LList<T> class that is not part of the list interface. Specifically, you will add a copy constructor and an equals() method to the LList<T> class. You should be familiar with these concepts from Assignment 1. However, in Assignment 1 you implemented these in an array-based class. Now you will implement them in a linked-list based class.

Specifically, you will implement the following methods:

```
// Creates a new list that contains the same data in the
// same order that was contained in old.  The lists should
// be separated from each other (so you must make new
// Nodes in the copy) but the data values within the lists
// can be shared (so the copy is not totally deep).
public LList(LList<T> old)

// Return true if the current LList<T> and the argument
// LList<T> contain the same data (based on the equals() method
// for class T) in the same order; return false otherwise
public boolean equals(LList<T> rhs)
```

Once you have completed your modified LList<T> class, test it with the main program CS445Rec4.java. Your output should match that shown in file Rec4Out.txt. To run this program you will need the following files:

LList.java (author's original class)
ListInterface.java (even though we are not using ListInterface<T> in the main program, since LList<T> implements ListInterface<T>, we still need the interface file.
CS445Rec4.java (main program to test your class)
Rec4Out.txt (output to use for comparison)

Note: To get the author's files from the CS 0445 Web site you will need to accessing it using a Pitt IP address – either Wireless Pittnet or via VPN. If you prefer not to do this, you can also get them from the course Canvas site.

Hints:

For the copy constructor you will need to iterate through the nodes of your old LList<T>, making a new Node corresponding to each old Node (and containing the same data). This can be done with a loop, adding each new Node to the end of the list. Be careful to make sure that you are linking the Nodes correctly. Also be careful about "one off" errors – don't miss any Nodes (ex: first or last) and don't iterate too far (could give a NullPointerException).

For the equals() method you will need to iterate through both lists, keeping separate Node pointers as you proceed down each one. You should first test the lengths of the lists, since clearly if they are not the same the answer will be false – this will save you the trouble of having to deal with lists of different lengths in your loop.

Important Note:

It is possible to implement these methods without directly accessing the underlying lists, but these methods will be VERY inefficient and you should not implement them in this way. For example, you could iterate through the original LList<T> using the getEntry() method and you could put the data into the copy using the add() method. Both of these methods must iterate through the list **with each call** and thus are very inefficient in the overall copy process. Consider lecture discussions wrt algorithm efficiency.