COMP 3160 – Fall 2014
Homework 1: Recursion

Number of People: Individual. Feel free to ask me for help, or visit the Computer Science Learning Center (www.cs.memphis.edu/cslc).

Due: Thurs., Sept. 11 by 1:00 pm

Submission: Zip all of your Java source files (you can zip the entire project folder if using an IDE) into a single file and upload it to the proper folder in the eCourseware dropbox at https://elearn.memphis.edu.

Coding Style: Use consistent indentation. Use standard Java naming conventions for **variableAndMethodNames**, **ClassNames**, **CONSTANT_NAMES**. Include a reasonable amount of comments.

Grader: TA, Kyle Cherry (kcherry2@memphis.edu). Questions about grading? Please contact him first!

1. **(5 pts)** Write an iterative (i.e., non-recursive) version of the Fibonacci method we discussed in lecture. Experiment with timing the iterative and recursive versions. How much more efficient is the iterative version?

2. **(5 pts)** Write a recursive method **compareStrings(String s1, String s2)**. This method should return 0 if the strings **s1** and **s2** contain exactly the same characters in the same order, 1 if **s1** is alphabetically (using ASCII values) "greater than" **s2**, or -1 if **s1** is alphabetically "less than" **s2**.

   Examples:

   compareStrings("Enterprise", "defiant") should return -1 (since ASCII values for capitals precede those for lower-case characters)
   compareStrings("enterprise", "enter") should return 1
   compareStrings("Excelsior", "") should return 1
   compareStrings("", "Excelsior") should return -1

3. **(15 pts)** In the **LinkedList<E>** class that we discussed in lecture, add recursive methods for each of the following wrappers. Your recursive methods should add at least one parameter to keep track of the current node in the list. You can use the posted code as a starting point.

   a. *(4 pts)* **contains(E someItem)** – returns **true** if **someItem** exists in the list, **false** if not.
   b. *(4 pts)* **search(E someItem)** – returns the index of the first occurrence of **someItem** if it exists in the list, or -1 if **someItem** is not in the list.
   c. *(7 pts, non-honors only)* **remove(int index)** – removes the node at the specified **index** in the list, or throws a **NoSuchElementException** if the index is invalid. (Hint: make sure it works for removing the head node too!)
   d. *(7 pts, honors only)* **reverse()** – reverses the calling list. (Hint: write a recursive method that reverses the list starting from a specific node, returning the head of the reversed list.)