CS 455 Lab 8: Java LinkedList class

Spring 2019 [Bono]

Goals and Background

This lab consists of some problems for you to practice programming with the Java LinkedList class, and in particular get practice programming with iterators. *You can only get the lab points for your solution to an exercise if you write code so that it takes no more than O(n) to traverse a list.* As we discussed in class, using the `LinkedList` `get` method for traversal takes $O(n^2)$ to traverse a list.

You will probably have to refer to the Java API documentation for LinkedList and ListIterator to complete the problems. We have provided links directly to those pages below.

Advanced preparation

There is more code to write in this lab than most, so you may want to spend some time before the lab preparing handwritten solutions to the problems, especially the last problem.

Reading and reference material

- Java API `LinkedList` documentation
- Java API `ListIterator` documentation
- Horstmann, Section 15.2 Using Linked Lists
- 2/26 CSCI455 Lecture on Java LinkedList class

Exercise 1 (1 checkoff point)

Write a static method `printAlt` that prints every other element in a list, starting with the first one. For example

```
list          printAlt(list) output
2 4 7           2 7
2 4             2
2 4 7 9 3 2     2 7 3
2 4 7 9 3       2 7 3
```

We have provided a test driver for your method that tests it on several hard-coded cases. The test driver source file, `PrintAltTester.java`, contains the stub for the `printAlt` method.

Exercise 2 (1 checkoff point)

Write a static method `after7` that removes all the values after the last 7 in a list. Note: this problem is adapted from an array problem on the codingbat.com website. Here are some examples

```
list          list after call to after7(list)
2 4 7                 2 4 7
2 4                   2 4
2 4 7 9 3 2           2 4 7
2 4 7 9 7 3 7 1 2     2 4 7 9 7 3 7
7 5 4                 7
```

We have provided a test driver for your method that tests it on several hard-coded cases, and compares the actual results with the expected results. The test driver source file, `After7Tester.java`, contains the stub for the

`after7` method.

Hint: you can initialize an iterator positioned at the end of the list (i.e., after the last element) as follows:

```
ListIterator<Integer> iter = list.listIterator(list.size());
```

Exercise 3 (1 checkoff point)

Write a static method `hasPeak` that returns true if its list argument has a single peak, and false otherwise. This means that the sequence of values consists wholly of an increasing part followed by a decreasing part. You may assume no two consecutive values in the array are the same (i.e., no flat areas).

list	**hasPeak(list)**
2 4 7	false
2 7 4	true
7 4 2	false
2 4	false
2	false
2 4 7 9 3 2	true
2 4 7 9 7 3 7 1	false
7 5 4	false
2 4 2 3	false
5 4 3	false

We have provided a test driver for your method that tests it on several hard-coded cases, and compares the actual results with the expected results. The test driver source file, `PeakTester.java`, contains the stub for the `hasPeak` method.

Checkoff for DEN students

Make sure you put your name and NetID in all the files you submit. No README required for this lab.

When you click the `Submit` button, it will be looking for and compiling (for source code) the files `PrintAltTester.java`, `After7Tester.java`, and `PeakTester.java`.
