Assignment 1
Linked Lists and Arrays

Release Time	Due Date
September 14, 2016	**September 27, 2016**

Objectives

- Creating linked lists
- To practice add, delete, search, change and sort operations using linked lists and arrays
- Practice developing high-performance solutions
- Compare theoretical vs empirical complexities

Problem Specification

In reality, we always use various data structures to store data, and different data structures have different advantages and dis-advantages. For instance, some data structures are good at adding/deleting new data, but they may have deficiency on searching. From this assignment, let's learn and compare the two basic data structures, namely linked lists and arrays, After the assignment is complete, you will know more about what are the real differences between these two data structures. Here we go.

Write a JAVA application to solve the following problem:

1) Generate n random integers from 1 to m i.e., in the range [1, m]

   (Hint: http://stackoverflow.com/questions/20389890/generating-a-random-number-between-1-and-10-java);

2) Store the generated numbers to a linked list (doubly-linked list should suffice; design your own for this homework exercise; do not use Java's built-in LinkedList class – of course, you may draw inspiration from it to design your own data members and methods);

3) Count how many numbers in the linked list are larger than 50, i.e, in the range (50, m] ;

   - If there are more than 5 integers that are larger than 50, sort the data in the linked list in a non-decreasing order, then delete the fifth element;
   - Otherwise, sort the linked list in a non-increasing order and delete the second element;

4) Insert the number 10 to linked list in its correct place (don't destroy the sort-order);

5) Then, output the data in your final linked list;

6) Redo steps 2) to 5) with array instead of linked list;

7) Perform a theoretical complexity analysis of your design (i.e., count number of operations/instructions and space usage) and then express that using asymptotic notation as a function of the input size (Pause: what is the input size of your problem?)

8) Empirically measure the time and space complexity of your code.

9) Compare the complexities from steps 7) and 8). (Long pause: in order to compare, what all you will need to do? We will discuss some alternatives in class, so stay tuned and this writeup will be modified accordingly at a later date.) *Sept 20 update*: input m and r from the console (m is the upper bound on the range of numbers to be generated, and r is the number of times to repeat the computation of steps 2 – 4 for each value of n for better measurement of average-time); use the following values of n=10, 50, 100, 500, 1000, 5000, 10000, 50000, 100000 etc – push the limit of your machine to maximum n that your machine can handle for your implementation; no need to output the final linked list or array when n > 100.

10) Write a SLC report including at most 1-page writeup on your observations of comparison from step 9, include any graphs/charts that may help express your observation precisely and concisely). This report must be produced using MS-Word or equivalent word-processing applications and must be of professional quality. A link to sample report using SLC has been included in the TopicsCovered page of our class' web presence.

Here is an example of what your data might look like. This is only an example. You will have your own randomly generated values.

- The n=10 input random numbers in the range [1,100=m]: 3, 5, 23, 56, 21, 7, 3, 67, 87, 1;
- Store them to a linked list: 3->5->23->56->21->7->3->67->87->1;
- The number of values larger than 50 is 3 which is less than 5, so the linked list should be sorted in a non-increasing order: 87-> 67 -> 56 -> 23 -> 21 -> 7 -> 5 -> 3 -> 3 -> 1;
- Delete the second element: 87-> 56 -> 23 -> 21 -> 7 -> 5 -> 3 -> 3 -> 1;
- Then insert the number 10 to the linked list: 87-> 56 -> 23 -> 21 -> 10 -> 7 -> 5 -> 3 -> 3 -> 1;
- Output the final linked list: 87-> 56 -> 23 -> 21 -> 10 -> 7 -> 5 -> 3 -> 3 -> 1;
- Then redo it with array instead of linked list;

Design Requirements

Code Documentation

For this assignment, you must include documentation for your code as generated by JavaDoc. You should have JavaDoc comments for every class, constructor, and method. By default, JavaDoc should output html documentation to a subfolder within your project (/dist/javadoc). Make sure this folder is

included when you zip your files for submission. You do not need to submit a hard copy of this documentation.

Hint: http://stackoverflow.com/questions/4468669/how-to-generate-javadoc-html-in-eclipse

Coding Conventions and Programming Standards

You must adhere to all conventions in the CS 3310 Java coding standard. This includes the use of white spaces for readability and the use of comments to explain the meaning of various methods and attributes. Be sure to follow the conventions for naming classes, variables, method parameters and methods. Read the material linked from our class web-pages (in case you can't recall programming styles and conventions from your CS1 and CS2 courses).

Testing

Make sure you test your application with several different values capturing different cases, to make sure it works.

Assignment Submission

- Generate a .zip file that contains all your files, including:
  - Source code files
  - Including any input or output files
- Javadocs
- A single-pdf and source files of your SLC report (step 10)
- Don't forget to follow the naming convention specified for submitting assignments