Assignment 9
(Full Score: 80 points)

(**Due in class on 3/11/Friday**)

Your name:	Score:

Java Collections API

Create a Java class/project that tests the performance of some of the Java Collections Framework classes and interfaces.

Requirements:

1. Your code must create an instance of the *java.util.LinkedList<Integer>* class.

2. Populate this class's instance with random integers. Make sure that these integers have both positive and negative values and are uniformly distributed.

3. Using the Java Collections Framework *java.util.Collections.min(Collection<? extends T> coll)* static method, find the **smallest element** in your list and record the **execution time** for this method.

4. Using the Java Collections Framework *java.util.Collections.sort(List<T> list)* static method, sort your list and record the **execution time** for this method.

5. Repeat steps 2 through 4 using different list sizes - suggested values are 10, 100, 500, 1000, 5000, 10000, 50000, 100000, and 1000000. You may modify these values or add new ones based on the actual running times of the program. The number and range of values should be selected so that you could perform the analysis of the experimental results and derive reliable Big-O estimates of the algorithms used to implement the *min()* and *sort()* methods.

6. Write an **assignment/project report (a WORD or PDF file)** to report the execution results in two formats::
   a. <observed running time> vs. <number of list elements>;
   b. <observed running time> divided by <running time from Big-O analysis of the method> vs. <number of list elements>.

7. Your report should include the Big-O estimates of the implementations of the *java.util.Collections.min(Collection<? extends T> coll)* and *java.util.Collections.sort(List<T> list)* methods as well as the justification of your estimates.

Continued on ***Back➔OVER***

8. Based on these results provide in your assignment report recommendations on the usage of the search for the minimum list element vs. sorting the list and returning the minimum element as the first element in the sorted list.

9. Create a javadoc file (HTML) for the class(es) in this assignments.

Hints:

1. Reuse the code from the performance testing framework created the previous assignments.

2. Use Java coding guidelines when naming your identifiers and creating class fields. The fields should be *private* or *protected* and appropriate accessor and mutator methods should be provided.

Submissions:

- Compress the **COMPLETE project folder** including the **Repor**t and **javadoc file** into a single compressed ZIP file and submit it at Moodle by due time.