Assignment 5: Program – Sorting – 20 pts
This week, we implement several sorting algorithms and test them for efficiency.

Due Date
Monday **2020/03/23** at 4:00pm

Instructions
Questions below marked with an asterisk (*) are group problems, and are intended to be worked on as a group. You may speak to anyone in class about the problem, ask questions, or help your classmates. However, you must write up and submit your own solution to the problem. All other questions are intended as individual work. You may not discuss them with anyone other than the Instructor or the TA. This includes questioning what the problem means, or what is expected.

Problems

1. *(5 pts.) Design a test that will demonstrate the effective run-time of selection sort, insertion sort, shell sort, merge sort, and quick sort based on number of comparisons and number of swaps. Your test should create 5 different sized arrays of ints – 10k, 100k, 1m, 10m, and 100m. You should then pass the *same array* of each size to the five different sort methods and store the number of resulting swaps and compares for each.

2. (5 pts.) Implement JUnit tests for each of the sorts you will test. This is not part of your sort testing, but rather your way of verifying that your implementations are correct before you start run-time testing. You should have a separate test method for each of the sorting methods so that running your unit test will accurately tell you which implementations are correct, and which aren't. However, you should notice that verifying that a sort is correct is the same algorithm regardless of what method was used. You should implement a private `isSorted` method in your test class that each unit test can call to verify that sort worked.

3. *(5 pts.) Implement your sorts, running your unit tests frequently to confirm that your implementations are correct. You may use any external resources (e.g., Wikipedia) to get the algorithm or Java source code for each sort. However, you must use the following method headers for your sorts:

   ```
   public void selectionSort(int[] a)
   public void insertionSort(int[] a)
   public void shellSort(int[] a)
   public void mergeSort(int[] a)
   public void quickSort(int[] a)
   ```

   Notice that these headers imply that your sorting algorithms will sort values in-place, and therefore change the array after being called.

4. (5 pts.) Report your results in a nicely formatted “Test Report” document (e.g., MS Word, LibreOffice, Google Docs, etc.). You should include a brief description of your test design, the sorts you’ve tested, your results and discussion (along with any graphical representations of your results that will improve your discussion), and any conclusions that you have drawn.