CS 0445 Fall 2021
Recitation Exercise 7

Introduction:

In Lecture 17 we discussed two simple sorting algorithms – InsertionSort and SelectionSort. Although these algorithms have similar asymptotic performance overall, their different approaches give them different performances under different circumstances. In this exercise you will look at these performance differences.

Details:

The two algorithms are implemented in file SimpleSorts.java. I have added a public static variable called `comps`. Look at the code to see where this variable is being used. Recall that a public static variable can be accessed via the class name, such as:

```
long totalcomps = SimpleSorts.comps;
```

Note that in each sort algorithm `comps` is first initialized to 0 and then it is incremented each time a comparison is done in the algorithm. Thus, the value of `comps` after a call to either sorting algorithm is the total number of comparisons done by that algorithm. Look again at the code and make sure that you understand why this is so.

In this exercise you will write a short main program that will do the following:

1) Ask the user to enter an array size and input it
2) Create two arrays of Integer equal in length to the value input by the user
3) Fill the arrays with random integers
4) Sort the arrays, one with each algorithm, and store the result of the static `comps` variable for each.
5) Sort the arrays again (note that they will already have been sorted), one with each algorithm, and store the result of the static `comps` variable for each.
6) Reverse the data in each array (so it will now be sorted high to low). There are various ways to do this.
7) Sort the reverse-sorted arrays, one with each algorithm, and store the result of the static `comps` variable for each.
8) Output the values stored to see the differences. Note that there will be three comparison values for each algorithm, one for random data, one for sorted data and one for reverse sorted data.

Think about our discussion of the algorithms in lecture and why the results are consistent with that discussion. Run the program several times, using different array sizes to see how the results change as the array size grows. Note also how the overall execution time increases as the array size gets larger.

Toward the end of the recitation period, share your impressions and conclusions with your classmates. Make sure everyone understands the differences in the algorithms and why they are so.