Professional Integrity Report (CPS101)

Student Name: Alex McColm Assignment #2

It took me 8-14 hours to complete the assignment.

These parts of the program work well:
Tests that the algorithms sort properly, and tests of algorithm's runtime and C value over larger and larger datasets, terminating when C stabilizes; stability being defined as when the difference between 3 contiguous pairs of C values is no more than 10%.

These parts of the program don't work well (please identify the specific problem):
In a word, spaghetti! There is a, for me, intolerable amount of copy-pasted code in here, that I would like to break down into **fewer** and more **efficient** methods. The program relies on **floating-point comparisons** which can be shaky and are not best practice. The improved quicksort is slower than the regular quicksort for most array sizes, for some reason I was not able to ascertain!

I learnt the following in doing the assignment:
Maybe this one is obvious, but I learned how to conduct a thorough and empirical algorithmic efficiency analysis, which certainly has applications in the field. I pushed my skills further and learned more on how to work with floating point values, timing operations, planning out methods, wrote headers and drew up a UML diagram for a much more complex, yet still relatively simple, piece of software than I had written yet.

The difficulties I encountered were:

I had a hard time planning out methods and figuring out how to track the last 3 percentage changes in C. I ran into hiccups with comparing the floating points, and it is still not best practice the way it stands currently.

Here are some other comments or suggestions:

I ran the test on Bubble Sort just out of curiosity. By the time array sizes were over 260,000 elements or so, sorting was taking more than 15 minutes. Bubble Sort was the only algorithm which had a C value greater than 1.0 (around 3.0) on arrays larger than 100,000 elements, other algorithms had C values which were decimal values between 0 and 1. Further, it surprised

me how *much* slower Selection Sort was compared to Insertion, Quick, Merge, and Radix Sort algorithms.