CPSC 223 - HW2 - Sorting 1

Goals:
The goal of this assignment is to empirically test what we have learned in theory–what is the time cost of various sorting algorithms under various input conditions?

You are given a framework that will help you answer this question. We are testing the following sorts:

Bubble Sort (*provided)*
Bubble Sort - Optimized
Insertion Sort
Selection Sort
The built-in std::sort() in CPP (*provided)*

You are tasked with implementing all of the above sorts, except for the generic bubble sort and std::sort() which is already provided.

To test each sort create sort *objects* and add them to the sortVector vector in main. Follow the example given for Bubble Sort.

Learning outcomes in this course include algorithm analysis and some more advanced C++ topics you may not have seen in previous courses. This assignment aims to introduce you to these ideas and grow your programming ability. I would suggest taking some time trying to understand the code before diving in. It's important to see how everything connects to each other and the included bubble sort implementation is a good reference.

Data collecting:
Each sort is timed over 5 runs, and the average time taken is printed to the console for each case. We are interested in collecting explicit time costs for each algorithm.

Note: you may decide it's easier to analyze the data by altering the console output and then redirecting to a file using a linux command. I would recommend this approach, however please keep the original console output for your final submission.

Comment and uncomment lines in the makefile (or make more cases) to time your sorts.
Your code should run using the commands: **make build** and **make test**

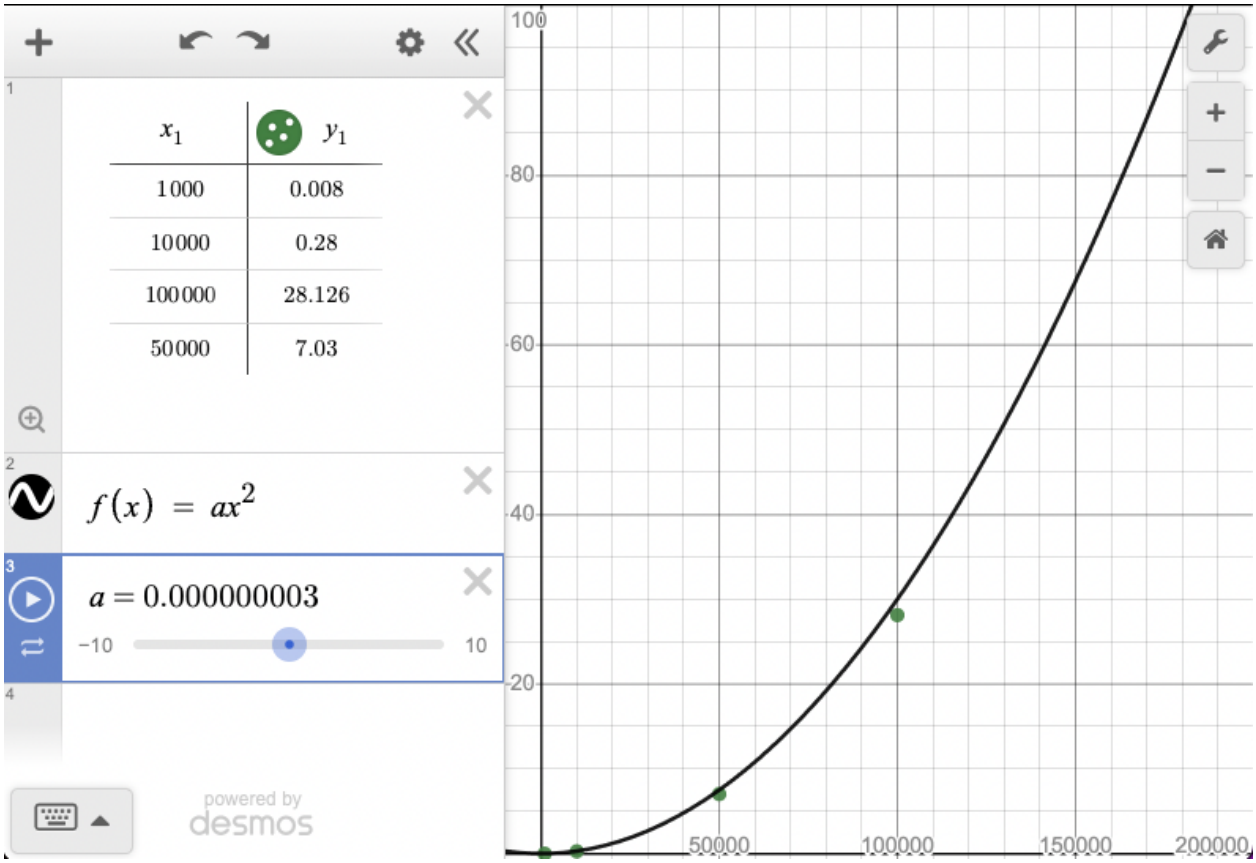
Depending on the speed of your system you may choose to use any large-size data set you wish, but you should use a range that gives a good representation of the growth rate of each algorithm. You must test *at least* n=100, n=1000, n=10000, n=20000. Include additional sizes for n as needed to extract a reasonable conclusion about the end behavior of the algorithm.

TODO:
Complete the code. All algorithms should function properly (there is an assertion that checks if the data is sorted) and all TODO's should be completed.

Create a report pdf in the "documents" directory that contains the following:

1. Copy-paste the output of your completed program into the pdf document
2. Pick one algorithm that you implemented (you cannot choose include bubblesort or std::sort) and graph the results (you may use any graphing tool you wish, i would suggest desmo). Plot all the points you collected for this algorithm for each case, ascending, descending, and random. Then plot two lines, some form of x^2 and some form of x. Meaning, your empirical data may not scale well with $f(x) = x^2$, but remember $0.000000003x^2$ is still a function of x^2.

   For example, some data for the generic bubble sort using ascending data is as follows:



   Notice, in order to "fit" your empirical data to an x^2 function you may need to find the right coefficient, as well as window scaling.

   Plot each category of points with a different color (e.g. green = randomized, red = descending, blue = ascending, or any colors you choose). Also, plot a quadratic line and

a linear line. Use enough data points to ensure a clear indication of growth rate (linear or quadratic), i would suggest five from each (random, ascending, descending).

Include a screenshot of your graph in your report and write at least one paragraph with your conclusions about this chosen algorithm's empirical growth rates compared to the theoretical growth rates.

3. Answer the following questions below and include them in your report:

Questions:

A. Based on the results, do you see any correlation to the number of inversions present in the initial dataset and the time to execute each sort? For each sort, explain.

B. Does the time cost of the sorts tend to follow expected growth for their input? Are they better or worse than expected? Which sorts acted as expected and which sorts surprised you? Include the vector size, n, and the data category for each example you discuss. Give three examples.

C. Based on the output times for the built-in cpp sort, what kind of sorting algorithm do you think std::sort() is using "under the hood"? It might be a good idea to test additional n sizes to investigate this question properly.

D. Reflection: having completed the assignment, what are some challenges you faced and how would you change your programming strategy if you were to do it again? Was there any particular area that you struggled with, or found easy? Feel free to journal about your programming and write about any experiences you had while completing this assignment.

Rubric:

Programming: (10 points) Program compiles (0 points in this category for non-compiling code) (5 points) Output is as expected: each sort runs successfully for each case and a time is produced. Format should match starter code output. (5 points) Commenting and style: each file should have comments at the top with your name, the file name, date modified, etc. Code you write should follow good conventions with variable naming and include comments as necessary to explain your code. (20 points) Selection Sort (20 points) Insertion Sort, (10 points) Optimized Bubble Sort	70 points
Report: (5 points) prompt #1 (13 points) prompt #2 - Graph of data (12 points) Each question at end is answered, 3 points each	30 Points