CS 325 HW 1 – 30 points

Submit a copy of all your code files, data.txt and the script file in a ZIP file to TEACH. Submit the written report and pseudocode to Canvas.

1) (10 pts) **Merge Sort and Insertion Sort Programs**
Implement merge sort and insertion sort to sort an array/vector of integers. You may implement the algorithms in C, C++ or Python, name the programs "mergesort" and "insertsort". Your programs should be able to read inputs from a file called "data.txt" where the first value of each line is the number of integers that need to be sorted, followed by the integers.
Example values for data.txt:

4 19 2 5 11
8 1 2 3 4 5 6 1 2

The output will be displayed to the terminal.
For the above example the output would be:

2 5 11 19
1 1 2 2 3 4 5 6

Submit a copy of your pseudocode to Canvas and code to TEACH.

2) (10 pts) **Merge Sort vs Insertion Sort Running time analysis**
Modify code- Now that you have verified that your code runs correctly using the data.txt input file, you can modify the code to collect running time data. Instead of reading arrays from the file data.txt and sorting, you will now generate arrays of size n containing random integer values from 0 to 10,000 to sort. Use the system clock to record the running times of each algorithm for ten different values of n for example: n = 5000, 10000, 15000, 20,000, …, 50,000. You may need to modify the values of n if an algorithm runs too fast or too slow to collect the running time data (do not collect times over a minute). Output to the terminal the array size n and time to sort. Name these new programs insertTime and mergeTime.

In your write up submitted to Canvas explain how you modified your code to collect running times.

Submit the code to TEACH.

3) (10 pts) **Written report and data analysis**
a) Collect running times - Collect your timing data on the engineering flip server. You will need at least eight values of t (time) greater than 0. If there is variability in the times between runs of the same algorithm you may want to take the average time of several runs for each value of n. Create a table of running times for each algorithm. Is this best case, worst case or average case running time.

b) Plot data and fit a curve - For each algorithm plot the running time data you collected on an individual graph with n on the x-axis and time on the y-axis. You may use Desmos, Matlab, R or

any other software. What type of curve best fits each data set? Give the equation of the curve that best "fits" the data and draw that curve on the graphs. How does your experimental running times compare to the theoretical running times?

c) Combine - Plot the data from both algorithms together on a combined graph. If the scales are different you may want to use a log-log plot.

d) Prediction – For each algorithm use your best-fit equations to predict the running time for an array of size n = 500,000.