CPSC 319
Assignment 1

Implementation and Comparison of Sorting Algorithms

Computers are frequently used to sort lists of items, especially when these lists are long. Many sorting techniques are available, some more efficient than others. In this assignment, you will implement four different sorting algorithms in a Java language program, and compare their performance. The sorting algorithms you will implement are:

1. Selection Sort
2. Insertion Sort
3. Merge Sort
4. Quick Sort

You can base your own code for these algorithms on the code given in the textbook or discussed in lectures, or on code found from other sources. Be sure to cite the source for any code you borrow or adapt, putting the citation in comments in your program.

Your program will create an array of integers to be sorted. The user will be able to specify whether the array is filled with integers in random order, ascending order, or descending order. The length of the array is arbitrary and will be specified at run time. The array may contain duplicate numbers. Your program will take this array and sort it into ascending order, outputting the sorted list to a text file, one item per line. The sorting algorithm to use and the name of the output file will also be specified at run time using command-line arguments.

Your program will time how long it takes to sort the array. Be sure to time only the sorting function itself, and not the time taken to fill the array with numbers or do input or output. Your program will print the time in seconds to the screen (standard output). You will use this data to help compare the efficiency of the four algorithms.

Write a program in the Java programming language to implement the above requirements. Your program will be invoked from the command line as follows:

```
java Assign1 order size algorithm outputfile
```

where *Assign1* is the name of the file containing executable bytecode for your program, *order* is the order of the integers in the input array (use one of the following strings: *ascending*, *descending*, *random*), *size* is the number of items in the integer array to be sorted, *algorithm* specifies the sorting technique to use (use one of the following strings: *selection*, *insertion*, *merge*, *quick*), and *outputfile* is the name of the output file where the sorted list will be written to. Be sure your program detects any command-line input errors (for example, a negative number for *size*), printing out an error message, and aborting the program.

Experiments and Data Collection

You will use your program to perform a series of experiments that help you to compare the efficiency of the four sorting algorithms in the best case (input array in ascending order), worst case (input array in descending order), and average case (input array in random order). At the very least, use the following sizes for the array length: 10, 100, 1000, 10,000, 100,000, and 1,000,000. Determine the timings for all four algorithms using these inputs.

Complexity Analysis

Using the techniques shown in class, do a complexity analysis of each of the four algorithms. Be sure you show all the steps for this, from counting operations to the final big-O characterization of the algorithm.

Report

Create a formal written report that describes at least the following:

1. The experimental method used.
2. The data collected (use tables and graphs to help illustrate this).
3. Data analysis. What does the data tell us about the algorithms?
4. The complexity analysis (show all steps for this).
5. Interpretation. What does the empirical data and complexity analysis tell us about the algorithms? How do the algorithms compare with each other? Does the order of input matter? How do algorithms within the same big-O classification compare to each other?
6. Conclusions. What algorithms are appropriate for practical use when sorting either small or large amounts of data? Are there any limitations with any of the algorithms? How does your analysis support these conclusions?

You can add more to this, if necessary. Be sure the report is complete, organized, and well formatted.

Submit electronically using the D2L dropbox:

1. Your source code files. Your TA will run your program to verify that it works correctly. Make sure your Java program compiles and runs on the Computer Science Linux servers.
2. Your report in Word or PDF format.

Be sure both components are labeled with your name, Unix login name, course number, tutorial number, TA's name, and assignment number.

Collaboration

The assignment must be done individually, so everything that you hand in must be your own original work, except for the code adapted from the text, lectures, or other sources to implement your sorting algorithms. When someone else's code is used like this, you must acknowledge the source explicitly in your program. Copying another student's work, in whole, or in part, will be deemed academic misconduct. Contact your TA if you have problems getting your code to execute properly.

CPSC 319
Assignment 1 Grading

Student:____

Program			
	Command-line arguments	3	______
	Error checking of arguments	3	______
	Creation of input integer array	4	______
	Selection sort implementation	4	______
	Insertion sort implementation	4	______
	Merge sort implementation	6	______
	Quick sort implementation	6	______
	Timing code	2	______
	Output to file	2	______
	Code structure (design, documentation, formatting, etc.)	4	______
Report			
	Description of experimental method	2	______
	Description of collected data	4	______
	Analysis of collected data	4	______
	Complexity Analysis	8	______
	Interpretation	4	______
	Conclusions	4	______
	Complete, well organized and formatted?	2	______
Total		**66**	______ _____**%**