CSCI 230 -- Project 2 (Sorting)

Due Wednesday, 04/15/2020 (MW Section)
Due Thursday, 04/16/2020 (TTh Section**)**

This project is designed to help you better understand sorting by implementing and comparing different sorting algorithms on different types of data. Download a small data file ***small1k.txt*** containing 1,000 values in random order. In addition, download the large data file ***large100k.txt*** containing 100,000 values in random order. Sort the small data file using Insertion-Sort, Merge-Sort, Quick-Sort (last element as pivot), and Quick-Sort3 ("median of three" as pivot). For each value in the file, you must create a <key, value> pair and then sort a list of <key, value> pairs. You will have to sort two different lists of <integer, string> pairs and <string, integer> pairs. For instance, you need to create <1234, "1234"> pair and <"1234", 1234> pair for an input value of 1234 from the file. For the larger file, you can skip the slower sorting algorithm, insertion sort. For each list, make sure to use the same list of entries for each sorting method (in another word, make sure not sort the sorted list, but sort the original data for each sorting algorithm). Output all relevant information below for each input data file to a text file (two output files).

1. Sorting Method
2. Input file name
3. Number of values sorted and key data type
4. Number of Key Compares and Data Moves
   (each SWAP of two values is counted as 3 data moves)
5. The time in milliseconds
6. The first 5 entries and last 5 entries

You should put all sorting methods in a class named **Sorting** and the sorting methods should be able to handle different data types (i.e. using a comparator). Try to reuse as many functions in the driver as possible so your program will not be unnecessary long. You could set up your program so it would run through all test cases in one go or you could use a menu to select input file, sorting algorithm, etc. It is very important that you summarize the results in one or more tables and discuss them in your write up including any expected and unexpected results. Make sure to compare the times collected for integer key versus string key to see any conclusions can be made.

Extra credit: You can earn up to 5 additional points if you can perform two of the following:

- Implement a Hybrid-Sort (use quick-sort3 or merge-sort and then insertion sort with 20 being the cutoff)
- Implement Heap-Sort
- Implement Shell-Sort
- Implement Radix-Sort (numbers of compares and data moves are not applicable)

Team Project: It is not required but it is recommended that you work in a team of two members. You must sign up by Wednesday, 03/25/20, for MW class or Thursday, 03/26/20, for TTh class if it is going to be a team project (email me). Make sure to split the responsibilities equally and work together. You do need to provide a short description on who did what and your experience as a team project. It is possible that all team members may not receive the same score. **If it is a team project, you need to implement two items of extra credit options as a required component and you can optionally implement two more items for extra credit**.

Alternate Project: Convert the sort animation Java applet from Bentley book as shown in class to a JavaFX application. You can add quick-sort3 (median of three) for extra credit. If you are doing a team project, then do both quick-sort3 and an additional sorting algorithm for extra credit.

Please provide documentation and applying good coding style because it is part of the grade. You must come up with a sufficient number of test cases since the test cases are also part of the grade. Please submit the following items (items 1 – 5 can be either be hardcopy or one pdf file on Canvas; if all items are submitted via Canvas, you must turn in a hardcopy of either title page or project sheet):

1. Title page with name, class, project number, and relevant information about your program (compiler and system used, file names).
2. Notes about your program (status of your program and lessons learned at the minimum).
3. A table with summarized results and discussion **(5 points as part of documentation)**.
4. Printouts of any input/output.
5. A printout of the source code.
6. A copy of your source code on Canvas (.h /.cpp or .java file).

Your program will be graded as follow:

- Correctness/Efficiency: 35 points
- Test Cases: 5 points
- Documentation/Coding Style: 10 points
