CS 124 Project 4

Due by 11:59pm on Wednesday, November 10th

For this project, you will sort the 1000 objects from your data set. You will modify each sorting algorithm to collect data. You will analyze the results from the different sorting algorithms.

Implement

- You should have your 1000+ objects stored in a vector, initially unsorted.
- Choose five sorting algorithms:
  1. Bubble Sort
  2. Selection Sort or Insertion Sort
  3. Merge Sort or Quick Sort
  4. Heap Sort
  5. Two-sort: sort by any algorithm (one of the above or a different one), then sort on a different field using a stable sorting algorithm (again, one of the above or a different one).
     - Hint for implementing two-sort: for the second stable algorithm, make a copy of the sorting function and take out the template part. That way you will be able to call a getter on your custom-type objects to compare a second field of your class.
- Modify each sorting algorithm to record the number of reads. This is the number of times you use a Comparable object. This could be using it to store somewhere else, using it to compare to another object, etc. Temporary Comparable objects count towards the reads.
  - Example code:

```
if (vec[i] > vec[i+1]) // This counts as two reads, which should
// be counted whether the if statements evaluates to true or false.
Comparable temp = vec[i]; // This is one read (for vec[i]).
smaller.push_back(vec[i]); // This is one read (for vec[i]).
```

- Modify each sorting algorithm to record the number of writes. This is the number of times you assign into a Comparable object. This could be to store a temporary Comparable, to overwrite an item in a Comparable vector, to push_back onto a Comparable vector, etc.
  - Example code:

```
Comparable temp = vec[i]; // This is one write (for temp).
smaller.push_back(vec[i]); // This is one write (for smaller).
vec[i] = vec[i+1]; // This is one write (for vec[i]) and one read
// (for vec[i+1]).
```

- Record the number of reads and writes needed to sort a vector of size 100, 200, 300, 400, 500, 600, 700, 800, 900, and 1000. Each of the five sorting algorithms should be given identical unsorted vectors to begin with.
- Analyze the data. Graph the number of reads and writes for each sorting algorithm and look at how the number of reads and writes grows when the size of the data set grows. Compare and contrast the different sorting algorithms and draw conclusions about which sorting algorithms are more efficient. Discuss complexities and their effects. All of this will go in your report.
- In your report, also include answers to the following questions: If you need to sort a contacts list on a mobile app, which sorting algorithm(s) would you use and why? What about if you need to sort a database of 20 million client files that are stored in a datacenter in the cloud?
- You must submit your source file(s), your data file(s), and your report. Please submit your report in PDF format.

Extra Credit

To earn up to 10 extra credit points (at the grader's discretion), you can get more thorough results. This can include:

- Setting timers to record how long it takes you to sort the objects with each algorithm.
- Performing the same experiment, except double the size of the data set each time (instead of having it grow linearly).
- Using more sorting algorithms.

If you implement any of these extra credit items, make sure to explicitly state it in your report with the results and analysis.

Grading

The project is out of 90 points.

5 pts	Program compiles and runs.
5 pts	Code style. Readable, naming style is consistent, comments where appropriate.
5 pts	Use five sorting algorithms according to the directions above.
15 pts	Sort the 100, 200, … 1000 objects according to the directions above.
40 pts	Record the correct number of reads and writes for each sort.
15 pts	Report: analysis of results.
5 pts	Report: professional, grammatically correct, cites sources.