CptS 233 Microassignment #5 - Instrumented Sorting

For this micro assignment, you must implement the following two sorting algorithms:

- **Mergesort**
- **Quicksort**

The sorts you implement each reside in their own files: QuickSort.java. and MergeSort.java. You'll note that the interface provided takes both the vector to be sorted and a pointer to a stats structure. Please look at the InsertionSort implementation carefully to see how the stats structure is used. The reason this is an instrumented sort assignment is the primary focus centers around counting how many times the different sorting algorithms compare values in the vectors and how many moves/swaps happen during the sort.

In the InsertionSort algorithm, I put a stats.comparisons++ before the actual comparisons in the if statements. This will count the number of times the algorithm does comparisons.

Then, since InsertionSort does moves, I count the number of moves. If you use a swap function, just count it once, but be consistent between your Quick and Merge sort implementations so the overall stats are comparable to each other.

The code has a Makefile to build, run tests, and cleanup if there's a problem:

You can run this with 'make' and 'make test' as per usual

You can also do larger tests: 'make test_thousands', 'make test_million', 'make test_billion'
Though, I've never waited through the million or billion run yet.

The make test and test_thousand stages are run on the CI server

Depending upon your implementations, the timing and comparison numbers might vary, so don't assume you'll get exactly the same values as someone else. The timing shall also vary wildly between runs just because the kernel won't always let you run the same amount of CPU time at any given moment, so the clock does lie here.

It currently builds and runs properly, but Merge, and Quick sort fail their tests because they're stubbed in code. For the recursive sorts (merge & quick), you will probably want to create more functions, much like we do for trees where the insert() calls an insertHelper(). Feel free to add more functions to the sorting files as needed, but don't change the interface on the existing function or the tests will fail.

Grading

Your submission will be graded based on the following:

1. [9] Your solution builds, does not cause any runtime issues, and passes all test cases
2. [1] Style and structure

BONUS:

If you decide to get extra points, then implement Radix sort as well. You'll have to create the class and add it to the tests yourself. I'll give you an extra 10 points if you get it implemented and added to my tests.

Final Submission

As per normal, the assignment is to be checked into your git repo. The only thing turned into Blackboard is a small file to let the TA know to grade your work for this assignment.