CS 253: Homework 1 (total 8 points: 6 points for Problem 1, and 2 points for problem 2)

**Problem 1.** Design a program that allows you to experiment with different sorting algorithms. This program should allow you to easily plug-in new sorting algorithms to empirically compare their run time performance as explained below. For this homework, you will work with insertion, selection and bubble sorts (more sorts will be added later).

**GRADING NOTES:**

* **1 point will be taken out automatically if this design specification is not satisfied**, i.e. algorithms are implemented separately, not as part of a unified framework. This penalty will also be applied for further extensions of the sorting framework (in further homeworks).
* **Missing either one of the two sets of experiments explained below will result in a penalty of 2.5 points.**

Assume that input data is stored in text files (presumably, automatically generated). There must be files of ordered data (prototype and large set), files of data in reverse order (prototype and large set), and files of random data (eventually more than one for prototype, and more than one for large data sets). You will experiment only with integers. MUST WORK WITH THE SAME DATA SETS in all experiments – keep all these text files for further experiments.

You will experiment with your program(s) in two steps:

**Step 1:** Experimenting with a prototype data (integers from 1 to 16) to ensure that your implementation works correctly and the results match expectations. The results must be reported in a table format (not generated by the program, but collected manually from multiple program runs) in the a Word document as follows:

ordered dataset reverse order random order

comparisons exchanges comparisons exchanges comparisons exchanges

bubble sort ... ... ... ... ... ...

selection sort ... ... ... ... ... ...

insertion sort ... ... ... ... ... ...

**Step 2:** Experimenting with large data sets of 2000 elements (1 – 2000). The results must be reported in a separate table with the same format.

In addition, in the report, explain the empirical results generated by your program comparing them to the known theoretical results (you must figure these numbers out for prototype data to validate your results) paying special attention to any discrepancies between empirical and expected results. The total length of the report must NOT EXCEED 1.5 pages, including tables. The working code is a DEFAULT MUST (submitting a code that does not work will results in 0 points for this project), but 90% of the grade will be assigned to the report itself.

**Must submit for grading:** the java code ONLY (all .java files, do not submit .txt files -- DO NOT ZIP your files) and the report (Word document) containing your empirical results summarized in two separate tables as explained above plus brief explanation (max 1 page) which discusses YOUR OWN numbers/results.

**Problem 2.** Study, program, and compare the efficiencies of the following recursive algorithms:

- Recursive binary search

- Computing the factorial of N

- Computing the N-th Fibonacci number.

Must submit for grading the tree of recursive calls for each program, the Big-O function with an explanation for each program, PLUS the java code and example runs to demonstrate the correctness of the implementation.

**Remember that homework projects are individual assignments.** Shared work will be graded by the rule "Share the work, share the grade". However, you can get some help from the code on my web site or occasionally additional help posted on BB. Also, please remember that this is NOT a programming course -- you will be graded on your understanding of CONCEPTS and THEORY that each project covers. Please DO NOT assume that I will be addressing java related questions – writing java code is covered in CS 151 and CS 152, and is assumed skill for this course.