Assignment 1

Sorting the Sorts

Out: February 4, 2015.
Due: February 24, 2015.

Consider the following ubiquitous problem:

SORTING
Input: An array A of integers of size N (where N is a positive integer).
Output: The array A sorted in ascending order.

Assume that the cost model charges an algorithm one unit for every comparison that involves an element of the input array. This should holds even if the array element is transfer out of the array to a different variable.

Provide a C++ implementation of the following sorting algorithms: bubble sort, insertion sort, selection sort, mergesort, and quicksort (with a random pivot and with a median pivot).

The sorting algorithms should have the following prototypes:

```
        void bsort(int array[], int size); // bubble sort
        void isort(int array[], int size); // insertion sort
        void ssort(int array[], int size); // selection sort
        void msort(int array[], int size); // mergesort
        void my_qsort(int array[], int size, int (*choose_pivot)(int [], int)); // quicksort
```

Also, implement the *linear-time* algorithm for finding medians and use this to implement a quicksort that uses a median pivot.

Prepare a single plot that compares the costs of the above sorting algorithms as a function of the array size.

Sort your sorting algorithms based on the empirical cost analysis

---

What to Hand-in

Source:
Submit via **moodle** a single C++ source file called "sorts.cpp" (containing all the sorting functions and any auxiliary functions):

- Your program should compile (with g++) and run (under Linux) in the ITL lab or on the AFS system. Test your program using a test driver (called "main.cpp") compiled using a makefile provided through our moodle course page (corresponding to this assignment).
- Your source programs should be properly documented (include your name in the documentation).

Analysis
Submit a **hardcopy** of a theoretical analysis (in the form of a solved recurrence or summation) for each of the sorting algorithms as well as the median finding algorithm.

Plot:
Submit **hardcopies** of:

- a single plot which shows the costs of all sorting algorithms as a function of varying input sizes (from 1 to at least 128). Provide a ranking of your sorting algorithms from best to worst (that is, sort them). Confirm if this agrees with the theoretical predictions.
- a single plot that shows the cost of your median finding algorithm versus the function 10N (where N is

the array size).