CSCI 3104-Spring 2016: Assignment #4.

Assigned date: Tuesday, 2/16/2016,
Due date: Tuesday, 2/23/2016, before class
Maximum Points: 50 points (includes 5 points for legibility).

Note: This assignment *must be turned in on paper, before class.* Please do not email: it is very hard for us to keep track of email submissions. Further instructions are on the moodle page.

P1 (40 points) Provide efficient algorithms for each of the problems below. You should describe the main idea in a few sentences and provide pseudo code. Your pseudocode can use any previously taught algorithm as a function call. Just specify what the call does and its time complexity.

Also, write down the worst case complexities for your algorithms. Assume all arrays below are integer arrays.

(A, 5 points) Given *sorted arrays* `a` and `b`, each of size n, do they have an element in common?

(B, 5 points) Given a *sorted array* `a` of size m and an unsorted array `b` of size n, do they have an element in common? Your solution must not sort the array `b`.

(C, 10 points) Given *unsorted arrays* `a` and `b` of size n, do they have an element in common? Your solution must avoid sorting either array.

(D, 10 points) The rank of a given number e in an array a is the number of elements in a that are strictly less than e. Also, given array a, let $\mathsf{minimum}(a)$ represent the minimum element of a.

Given *unsorted arrays* `a` and `b` of size n, find the rank of $\mathsf{minimum}(a)$ in the array `b`. Your solution must run in time $\Theta(n)$.

(E, 5 points) Given *min heaps* `a` of size m and `b` of size n, merge them into a single *min heap* of size $m + n$. Assume that $m \leq \sqrt{n}$.

(F, 5 points) Given two unsorted arrays `a` and `b`, each of size n, output "YES" if every element of `a` is strictly less than every element of `b`, and "NO" otherwise.

P2 (10 points) A company constantly receives (integer) price quotes for its product over the internet, and at any point, we are asked to store the k smallest price quotes, where k is fixed by the manager in advance.

We consider three possible data structures for the task of maintaining the k smallest quotes so far.

1. A min heap data structure with k elements in it.
2. A balanced binary search tree data structure.
3. A sorted array of k elements.

For each of the structures, write down how you would update the data structure when a new quote comes in and the worst case running time.