Homework 4

Out: 10.12.16
Due: 10.19.16

1. [Sorting, 20 points]
   a. Give an example of a worst-case sequence with 10 elements for insertion sort, and show that insertion-sort runs in $\Omega(n^2)$ time on the sequence.
   b. What is the worst-case running time for bucket sort, and for which input characteristics will it be exhibited? Propose a modification to the algorithm that will preserve its linear average-case running time and reduce the worst case running time to $\Theta(n \log n)$.

2. [Sorting, 20 points]
   We modify the deterministic version of QuickSort such that instead of selecting the last element in the sequence as the pivot, we choose the element at index $\left\lfloor \frac{n}{2} \right\rfloor$.
   a. What is the running time of this version of QuickSort on a sequence that is already sorted? Explain.
   b. Provide an input sequence that would cause this version of QuickSort to run in $\Omega(n^2)$ time.

3. [Stable Sort, 10 points]
   Which of the following sorting algorithms are stable: insertion sort, merge sort, heapsort, and quicksort? Explain. Provide a scheme to convert any sorting algorithm to stable. How much additional time and space does your modification require?

4. [Remove Duplicates, 10 points]
   Describe and analyze an efficient algorithm for removing all duplicates from a collection of n elements. You may assume that the ordering of the elements does not matter.

5. [Heaps, 10 Points]
   You are given the array representation of a binary max-heap containing n elements, and a real number x. Design an O(k) time algorithm to determine if the k^{th} largest element in the heap is less than or equal to x. (Hint: Notice that you are not asked to find the k^{th} largest element itself.)

6. [Collinear Points in a Plane, 30 points]
   Implement a program that reads N points in a plane and outputs any group of four or more collinear points (i.e., points on the same line).

   Your program should read the input from the provided *points.txt* file, which contains the number of input points on the first line, followed by two floating-point numbers representing the x-axis and y-axis for a point on each of the following lines. You may assume that all input points are unique. Your program should then output to the screen the collinear points in the format specified in the sample output below. If there is more than a single group of collinear points, you should separate the groups with an empty line. The ordering of the points within each group does not matter.

Sample *points.txt* file (provided), which describes the set of points (1, 4.5), (3,4), (4,4), (2,2) (1,1) (0,0) (-1,-1):

```
7
1 4.5
3 4
4 4
2 2
1 1
0 0
-1 -1
```

Sample output for the above input:

```
(-1,-1)
(0,0)
(1,1)
(2,2)
(4,4)
```

For this exercise, you are required to use STL data structures. You may find it useful to define lines with slopes and y-intercepts. Try to make your implementation as efficient as you can.

Submit your solution, including all associated files, along with a modified *makefile* that you used to compile it on the lab computers. Make sure to write your name in a comment at the top of the program, along with an explanation of your algorithm and its runtime.