Name:

Login:

Signature:

ECE 368 Spring 2016

Homework 4

1. Heap Sort (30pts)

a) The time complexity for operations on binary min-heaps is the following:

	front	arbitrary	back
insert	$O(\ln(n))^*$	$O(1)$	$O(1)$
access	$O(1)$	$O(n)$	$O(n)$
delete	$O(\ln(n))$	$O(n)$	$O(n)$

Justify the time complexity for each entry. **(10pts)**

b) There are the following two strategies for pop using heaps:

1. Remove the top and heapify

2. Replace the top with the last element in an array implementation and then percolate it down.

Assume we start with a binary min-heap that is also a complete tree. Which of these two strategies does not maintain the completeness of the tree? Give a counter example if the strategy that does not maintain completeness and provide an explanatory proof if it does. **(10pts)**

c) Sort the following 12 entries using heap sort. Show each step of heap sort using either tree or array. **(10pts)**

34	15	65	59	79	42	40	80	50	61	23	46

2. Insertion Sort vs Merge Sort (10pts)

Consider a category of unsorted arrays where the number of inversions grows linearly with the size of the array. In other words, the number of inversions is O(n). Which one is better, insertion sort or merge sort? Explain the reason.

3. Programming Questions (60 Pts):

a) Given the "hw4.c" file, write the following functions:

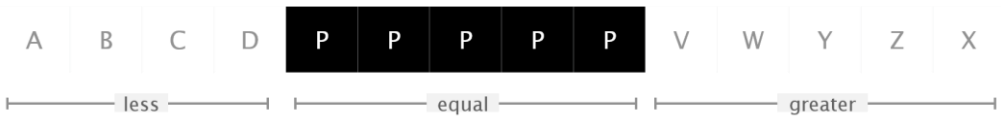
InsertionSort (10 Pts)

QuickSort1 (10 Pts)

QuickSort2 (10 Pts)

QuickSort3P (20 Pts)

- Notes:
  - For QuickSort1, make pivot the first element of subarray (Array[first]).
  - For QuickSort2, make pivot the median between the first, middle, and last element from the array (Array[first], Array[floor((first+last)/2)], Array[last].
  - For QuickSort3P, implement 3 way partitioning quicksort which partitions the array into 3 parts (less than, equal to and greater than the pivot):



  - Your files will be graded by inputting an Array stored in file named "Array.txt" or "Array2.txt" and checking the results in a file named "Sorted.txt". **Changing this names will jeopardize your grading**.

  - You need to submit only one file named "hw4.c".

b) Algorithm Complexity **(10 Pts)**

For the input file "Array2.txt", what are the total numbers of comparisons and moves for:

Algorithm	Comparisons	Moves
InsertionSort		
QuickSort1		
QuickSort2		
QuickSort3P		

- Notes:
  - Whenever you use the "=" operator, add +1 move.
  - Whenever you use the ">, <, ==, <=, >=" operators or loop through a **while/for** loop, add +1 comparison.
  - Answers may vary depending on your implementation.

c) Extra Credit **(5 Pts)**

For the input file "Array.txt", what are the total numbers of comparisons and moves for:

Algorithm	Comparisons	Moves
InsertionSort		
QuickSort1		
QuickSort2		
QuickSort3P		