ESO207A: Data Structures and Algorithms	
Homework *2: Heaps, QuickSort*	HW Due Date: Sep 3, 2018

Instructions.

1. Solutions have to be submitted on Gradescope. For this, each problem should start on a new sheet. If this is not followed then it may not be possible to grade your answers.
2. For each problem, write your name, Roll No., problem number, the date and the names of any students with whom you collaborated. Remember that you must write the answer and the algorithm in your own words.
3. For questions in which algorithms are asked for, your answer should provide the following: (a) A clear description of the algorithm in English and/or pseudo-code, where, helpful, (b) A proof/argument of the correctness of the algorithm, and, (c) an analysis of the running time of the algorithm.

   Full marks will be given only to correct solutions which are described clearly. Convoluted and unclear descriptions will receive *low marks*.

Problem 1. *Heap.* The following is an alternative routine to build a max-heap by permuting the elements of an array. It uses the subroutine *Max-Heap-Insert*(A, v). Here A is a max-heap where the elements are stored in the array $A[1 \ldots N]$, where, N is the maximum capacity of the array, and (ii) the field $A.heapsize$ that counts the number of items in the heap A. This routine takes the heap A and inserts v into the max-heap. It also increments $A.heapsize$ by 1.

1. Consider the following procedure for building a heap.

   procedure Build-Max-Heap(A, n) // Build Max Heap for the elements in $A[1, \ldots, n]$
   1. $A.heapsize = 1$
   2. **for** $i = 2$ **to** n
   3. *Max-Heap-Insert*$(A, A[i])$

   Show that this procedure takes $O(n \log n)$ time. (10)
2. Show a worst case input and analyze it to show that on that input the procedure takes $\Omega(n \log n)$ time. Hence this algorithm takes $\Theta(n \log n)$ time worst-case. (15)

Problem 2. *Young tableau.* $(12.5 \times 4 = 50)$

An $m \times n$ Young tableau is an $m \times n$ matrix such that the entries of each row are in sorted order from left to right and the entries of each column are in sorted order from top to bottom. Some of the entries may be ∞ that are treated as non-existent items. Thus a Young tableau can be used to hold $r \leq mn$ finite numbers. Note that an $m \times n$ tableau Y is empty if $Y[1, 1] = \infty$ and is full if $Y[m, n] < \infty$. An example Young tableau is $\begin{bmatrix} 2 & 4 & 8 & 12 \\ 3 & 5 & 9 & \infty \\ 12 & 14 & 16 & \infty \end{bmatrix}$

1. Give an algorithm for EXTRACT-MIN on a non-empty $m \times n$ Young tableau that runs in time $O(m+n)$. (*Hint*: Follow Min-Heapify.)

2. Show how to insert a new element into a non-full $m \times n$ Young tableau in $O(m+n)$ time.

3. Using no other sorting method as a subroutine, show how to use an $n \times n$ Young tableau to sort n^2 numbers in $O(n^3)$ time.

4. Given an $O(m+n)$-time algorithm to determine whether a given number is stored in a given $m \times n$ tableau.

Problem 3. *Selection.* Consider a generalized version of the selection problem where, given an array $A[1 \ldots n]$ of numbers and a number k, we wish to return the first to the kth smallest elements in *some* order, not necessarily in increasing order. That is, we want to return the smallest element, the second smallest element, ..., the kth smallest element, and these elements only, in some order. Give an efficient algorithm with expected time $O(n)$ for this problem, that is, without fully sorting the array. Give an analysis of your algorithm. (25 points)