Homework 2
Due: Feb 18 (Thursday)

1. Show how to implement a queue using two stacks.
2. Suppose you are given two max-heaps, one of size m and one of size n. Design an efficient algorithm to merge them. (Hint, your target running time should be $O(m+n)$.)
3. Recall that in an AVL tree with n nodes, we require that for **every node**, the height difference between its left and right sub-trees is at most 1. What if we relax this requriement, and let the height difference be at most 2, is the height of this modified AVL tree still bounded by $O(\log n)$?
4. You are given an array of n real numbers and asked to sort the numbers. The twist here is that many elements have duplicates and there are only $k << n$ distinct numbers. Augue which sorting algorithm (heap sort, BST sort, insertion sort, merge sort, etc) is best suited for the problem, i.e., with the best running time.
5. We have discussed the divide-and-conquer polynomial multiplication algorithm, where each polynomial is partitioned into the high half and lower half. What if we partition the polynomials according to the index is even or odd. Will the divide-and-conquer appproach still work? Explain why or why not.
6. Given an array $A[1..n]$ of n integers, such that for all j, $1 \leq j < n$, $|A[j] - A[j+1]| \leq 1$. Let $A[1] = x$ and $A[n] = y$, such that $x < y$. Design an efficient search algorithm to find j such that $A[j] = z$ for a given value z, $x \leq z \leq y$.
7. Modify the radix sort algroithm to work for variable-length strings. In other words, you can no longer assume that all the numbers have exactly k digits. Some numbers maybe long and some maybe short. It is of course possible to pad all the numbers with "dummy" $0-$digits to make them all of the same length. Design an effcieint algorithm that avoids doing so and achieves a runing time linear in the total number of digits.
8. The input is a set of intervals on the $X-$axis, which are represented by their two endpoints. Design an algroithm to identify all intervals that are contained in another interval from the set. The algroithm should run in $O(n \log n)$ time.
9. Let $A = (a_1, a_2, ..., a_n)$ be a sequence of numbers. Another sequence $Z = (z_1, z_2, ..., z_m)$, $m \leq n$ is a subsequence of A if there exists a strictly increasing sequence $(i_1, i_2, ..., i_k)$ of indices of A such that for all $j = 1, 2, ..., k$, $z_j = A_{i_j}$. More intuitively, the sequence Z is obtained by deleting some numbers from A without changing the order of the remaining numbers. For example, if $A = (1, 3, 5, 7, 9, 11)$, then $Z = (3, 9, 11)$ is a subsequence of A.

   Design an $O(n^2)$ dynamic programming algorithm to find the longest mononotincally increasing subseqquence of a sequence of n numbers.
10. Consider the Knapsack problem as discussed in class. You are given a set of items with a positive integer sizes and positve value, and the goal is to find a sub-set of items to fill up a knapsack maximizing the total value. For this problem, you are given 2 knapsacks, each of a positive

integer size K. Design an algorithm to find a subset of items to fill up the two knapsacks, maximizing the total value.