Homework 2
Due: End of Day Friday October 24th, 2025

1. Show how to implement a stack using a binary search tree.

   (Hint, the problem may seems odd, but its goal is to ensure you understand stack, queue, and BST.)

2. Suppose you are given two min-heaps of size m and n. Develop an algorithm that can combine these two min-heaps into one min-heap in $O(m+n)$ time.

3. Suppose you are given two BSTs of size m and n. Develop an algorithm that can combine these two BSTs into one BST in $O(m+n)$ time.

4. As promised in class, please investigate whether partitioning into groups of 6 will lead to a linear time selection algorithm.

5. 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 sub-sequence of A if there exists a strictly increasing sequence $(i_1, i_2, ..., i_m)$ of indices of A such that for all $j = 1, 2, ..., m$, $z_j = A_{i_j}$. 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 sub-sequence of A.

   Design an $O(n^2)$ dynamic programming algorithm to find the longest monotonically increasing sub-sequence of a sequence of n numbers.

   Hint: Can you solve the problem by converting it to a shortest path problem on a directed acyclic graph?

6. The input is a permutation of a set of n numbers in an array $A[1..n]$ with the permutation σ specified. Design an $\Theta(n \log n)$ algorithm to order the numbers in A according to the permutation $(1, 2, 3, ..., n)$ without using more than constant additional space.

   For example, if $A = \{10.5, 9.3, 2.7, 13.6\}$ and $\sigma = \{4, 2, 3, 1\}$, then the output should be $A = \{13.6, 9.3, 2.7, 10.5\}$.

7. Consider the Knapsack problem as discussed in CS361. You are given a set of items with a positive integer size and a positive value, and the goal is to find a subset 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.

8. The input is a set of intervals on the $X-$axis, which are represented by their two endpoints. Design an algorithm to identify all intervals that are contained in another interval from the set. The algorithm should run in $O(n \log n)$ time.