Instructions: There are five problems in this exam. Some of them consist of several subproblems. Make sure you answer every part.

1. (3 pts.) For each subproblem below, indicate whether \( f(n) = O(g(n)) \), \( f(n) = \Omega(g(n)) \) or \( f(n) = \Theta(g(n)) \) and provide a brief explanation to your answer.
   a. \( f(n) = 100n + \log n \) and \( g(n) = \frac{n}{500} + \log^5 n \)
   b. \( f(n) = \sqrt{n} \) and \( g(n) = n \log n \)
   c. \( f(n) = 4^n \) and \( g(n) = 3^n \).


2. (4 pts.) Suppose a binary tree $T$ has 10 nodes which are labeled $a, b, c, \ldots, j$. We ran the preorder and inorder traversals on the tree and the nodes were processed in the following order:

**preorder traversal:** $j, d, a, c, b, i, g, e, f, h$

**inorder traversal:** $a, d, b, c, j, e, g, f, i, h$

a. Draw $T$ with its nodes labeled. Briefly explain how you arrived at the answer.

b. You can think of the two sequences above as the *preorder signature* and the *inorder signature* of $T$ respectively. Thus, what you’ve done in part (a) is you’ve reconstructed $T$ based on its preorder and inorder signatures. It turns out that this is always possible. That is, given a binary tree's preorder and inorder signatures, we can reconstruct the tree. However, using one signature is not enough. Come up with two binary trees that have exactly the same preorder signature.
3. (5 pts.) *Short questions.*

(i) Let $A[1 \cdots 20]$ store the keys of a min-heap with 20 items. Assume that all the keys are distinct. Answer the following questions:

a. What are the possible indices $i$ so that $A[i]$ contains the second smallest key? Why?

b. What are the possible indices $i$ so that $A[i]$ contains the third smallest key? Why?

c. What are the possible indices $i$ so that $A[i]$ contains the largest key? Why?

(ii) Starting with an empty AVL tree, insert the following keys: 35, 19, 20, 31, 29, 32. Show each step of the insertion.
More space ...
4. (3 pts.) Let $T$ be a binary search tree where each node $v$ has a field $v.size$ that indicates the number of items stored in the subtree $T_v$. Design an $O(h)$-time algorithm for the method $\text{countAllKeysLessThan}(k)$ where $h$ is the height of $T$. That is, given an integer $k$, the method should output the number of keys in $T$ that are strictly less than $k$. 
5. (3 pts.) Let $A$ be an $n \times n$ array. Each row contains only 1’s and 0’s and the 1’s appear before any of the 0’s. Assume $A$ is already in memory (i.e. you can immediately access its elements). Your goal is to design an algorithm that counts the number of 1’s in $A$.

a. Suppose you do the brute force method: check all the entries of $A$ to count all the 1’s. How much time will this take?

b. Now improve on it! Design an algorithm that counts the number of 1’s in $A$ in $O(n \log n)$ time.
Extra space...