Instructions: There are five problems in this exam. Some of them consist of several subproblems. Make sure you look through them carefully.

1. (2 pts.) Please order the following functions according to their growth rates, from slowest to fastest.

\[ \log^2 n, \log n, \sqrt{n}, 2^{100}, n \log n, \]
2. (2.5 pts.) In class, you learned how a min-heap can be used to store items so that the (i) insertion of an item and the (ii) deletion of an item with minimum key can be done efficiently. But a min-heap can easily be modified into a max-heap so that this time the (i) insertion of an item and the (ii) deletion of an item with maximum key can be supported efficiently.

a. Describe the two important properties a max-heap should have.

b. Starting with an empty max-heap, insert the following keys: 2, 1, 4, 5, 9, 3, 6. Then remove the maximum key. Please show the steps.
3. (2 pts.) Let $L_1, L_2, \ldots, L_k$ be $k$ LIST ADT’s whose elements are pairwise distinct (i.e. they do not have elements in common) and in sorted order. In one of the homeworks, you were asked to compare two different ways of combining the $k$ lists into one single sorted list. Here’s a third way that is a generalization of the UNION algorithm for two lists:

Start by initializing $L$ to an empty LIST ADT. Then let $f_i$ denote the first element stored in list $L_i$ for $i = 1, \ldots, k$. Find the minimum among these $k$ numbers, say $f_j$, and add it to the end of $L$. Replace $f_j$ with the next element in $L_j$. Repeat until all elements are added to $L$.

a. Write a pseudocode that describes the above procedure.

b. What is the running time of this algorithm? Keep in mind that there are $nk$ elements altogether.
4. (2 pts.) Suppose a company has been using a binary search tree $T$ to store its data. Somewhere along the way, however, the tree was corrupted, and the company is no longer certain that $T$ is correct. They know that all the data is still in $T$ but the ordering of the data may no longer be consistent with the rules of a binary search tree. Assuming $T$ has $n$ data items, describe an $O(n)$-time algorithm that outputs “yes” if $T$ is a proper binary search tree and “no” otherwise. Please briefly argue why your algorithm is correct and why its running time is $O(n)$. 
5. (2 pts.) This time around, let $T$ be an AVL tree. It contains $n$ items with distinct keys. Additionally, each of its internal node $v$ has an extra field that stores $v.size$, the number of items stored in its subtree. Given an integer $k$, where $1 \leq k \leq n$, describe an $O(\log n)$-time algorithm that outputs the item whose key is the $k$th smallest in the tree. Again, please briefly argue why your algorithm is correct and why its running time is $O(\log n)$. 
