1. (3 points)
The set of full binary trees is defined recursively:
Basis step: The tree consisting of a single vertex is a full binary
tree.

Recursive step: If $T_1$ and $T_2$ are disjoint full binary trees, there
is a full binary tree, denoted by $T_1 \cdot T_2$, consisting of a root $r$
together with edges connecting $r$ to each of the roots of the left
subtree $T_1$ and the right subtree $T_2$.

Use structural induction to show that $l(T)$, the number of leaves
of a full binary tree $T$, is 1 more than $i(T)$, the number of internal
vertices of $T$.

2. (4 points) Prove that no matter what node we start at in a height-$h$ binary search tree, $k$ successive calls to TREE-
SUCCESSOR take $O(k + h)$ time.

3. (3 points) Is the operation of deletion "commutative" in the sense that deleting $x$ and then $y$ from a binary search tree
leaves the same tree as deleting $y$ and then $x$? Argue why it is or give a counterexample.

4. (3 points) BT&T (Big Telephone & Telegraph) has 256 million customers. Their current telephone directory consists of
many heavy volumes typeset in small point size, which are expensive to print and inconvenient to use. To overcome the
above problems, BT&T has decided to set up an on-line computerized directory, and their software engineers are debating
what is the most efficient data structure for the purpose. Assume that the BT&T computer can compare two names in one
microsecond. You do not need to simplify your calculations.

4a) One of the engineers suggests implementing the on-line directory as an unsorted linked list. With this implementation,
give an estimate of the worst-case search and insertion times.

4b) Another engineer wants to implement the on-line directory as a red-black tree. With this implementation, give an
estimate of the worst-case search and insertion times.

4c) A third engineer proposes the use of a sorted array. With this implementation, give an estimate of the worst-case
search and insertion times.

5. (3 points) Describe a red-black tree on $n$ keys that realizes the largest possible ratio of red internal nodes to black
internal nodes. What is this ratio? What tree has the smallest possible ratio, and what is the ratio?

6. (4 points) What is the upper bound on the height of an AVL tree containing $n$ keys? Prove your result.

7. (2 points) Exercise 14.1-3 Write a nonrecursive version of OS-SELECT.

8. (3 points) Suppose that we would like to keep a set of keys in a binary search tree. New keys will be added at various
times, and searches may occur at any moment. Of course we’d like the tree to be balanced at all times, so that we can
search and insert in $O(\log n)$ time. One way to try to keep the tree balanced is the following randomized scheme: upon
insertion, assign each key a random real-valued score from $[0 \ldots 1]$. Keep the keys in a structure that is a binary search
tree on the keys and a heap on the scores. (I.e., if you only look at the keys of the resulting data structure, what you see is
a binary search tree. If you only look at the scores, what you see is a heap.) This structure is called a treap, because it is
both a tree and a heap. Here is an example with keys a, b, c, d and associated random scores .4, .8, .1, .7
8a) Prove (by using strong induction) that when the keys are distinct and each key has a fixed score, there is a unique structure that is both a binary search tree on the keys and a heap on the scores. In other words, prove that there is only one possible treap for the given assignment of scores to keys. You may also assume that the scores are distinct, because with probability 1 they are.

8b) Argue that regardless of the insertion order of the keys, a treap has expected height $O(\lg n)$. Hint: See section 12.4.