Consider a binary search tree. For a node $p$, denote by $p_{\text{pre}}$ and $p_{\text{post}}$ the positions of $p$ in the output sequences of Preorder-Tree-Walk and Postorder-Tree-Walk, respectively.\(^1\)

(a) (2 points) Show that if node $q$ is in node $p$’s subtree, then $p_{\text{pre}} < q_{\text{pre}}$ and $p_{\text{post}} > q_{\text{post}}$.

**Solution:** INSERT YOUR SOLUTION HERE

(b) (6 points) Is the converse (i.e., if $p_{\text{pre}} < q_{\text{pre}}$ and $p_{\text{post}} > q_{\text{post}}$, then $q$ is in $p$’s subtree) true as well? If so, prove it; otherwise, provide a counter example.

**Solution:** INSERT YOUR SOLUTION HERE

---

\(^1\)Recall that Preorder-Tree-Walk and Postorder-Tree-Walk print the root before resp. after the values in the subtrees.
Assume that you are given a 2-3 tree $T$ containing $n$ distinct elements.

(a) (4 points) Show how to find the successor of a given element $x \in T$ in time $O(\log n)$.

**Solution:** INSERT YOUR SOLUTION HERE

(b) (4 points) Show that if the input element $x$ is chosen *uniformly at random* from $T$, then your procedure from part (a) runs in *expected* time $O(1)$.

**Solution:** INSERT YOUR SOLUTION HERE

Assume that we wish to augment our 2-3 tree data structure so that each node $v$ maintains a pointer $v.succ$ to the successor of $v$, so that queries for the successor of an element can be answered in $O(1)$ time *worst-case*.

(c) (6 points) Show that the 2-3 trees can be augmented while maintaining $v.succ$, such that the *INSERT* and *DELETE* operations can still be performed in $O(\log n)$ time. (**Hint:** Think of a doubly-linked list.)

**Solution:** INSERT YOUR SOLUTION HERE
Consider a binary tree (BT) in which every node $v$ contains a unique key $v.key$ but in which the binary search tree property does not necessarily hold. In this task you will study the effects of using the usual search algorithm for BSTs in such a tree.

(a) (4 points) Consider the edge labeling that assigns 1 to an edge $(p, c)$, where $p$ is the parent of $c$, if “value in $c >$ value in $p$ and $c$ is the right child of $p$; or value in $c <$ value in $p$ and $c$ is the left child of $p”$ and 0 otherwise. Prove or disprove: $\text{Tree-Search}(v.key)$ finds $v$ if and only if all edges from the root to $v$ are labeled 1.

**Solution:** INSERT YOUR SOLUTION HERE

(b) (6 points) For every node $v$ in a BT with real-numbered keys, let $v.low, v.high \in \{-\infty, \infty\} \cup \mathbb{R}$ be such that $\text{Tree-Search}(x)$ traverses $v$ if and only if $x \in (v.low, v.high)$.\(^2\)

Describe in pseudo-code a linear-time algorithm based on pre-order tree traversal that assigns to each node $v$ values $v.low$ and $v.high$ such that the condition above is satisfied. Argue why your algorithm is correct.

**Solution:** INSERT YOUR SOLUTION HERE

(c) (4 points) Explain how to extend your algorithm in (a) such that it assigns to each node $v$ the value $v.found$ indicating whether $v$ is found if $v.key$ is searched for using $\text{Tree-Search}$.

**Solution:** INSERT YOUR SOLUTION HERE

\(^2\)Note that if $v.low \geq v.high$, then $(v.low, v.high) = \emptyset$. 

Assume we insert sequences of English letters into an empty 2-3-tree, following the standard alphabetical ordering when making comparisons: $A < B < C < \ldots < Z$.

(a) (3 points) Assume you insert the letters $R, E, L, A, T, I, O, N, S$ into a fresh 2-3-tree $T$ in that exact order. Draw the resulting tree. What is its height?

**Solution:** INSERT YOUR SOLUTION HERE

(b) (4 points) Let $h$ be your answer to part (a). Is it possible to get less than $h$ by inserting the letters from (a) in a different order? If not, prove it. If yes, give the best ordering you can. What is the depth $h_{\text{min}}$ you get?

**Solution:** INSERT YOUR SOLUTION HERE

(c) (3 points) Can you achieve $h_{\text{min}}$ with lexicographic order $A, E, I, L, N, O, R, S, T$? Draw the resulting tree.

**Solution:** INSERT YOUR SOLUTION HERE

(d) (4 points) **Extra credit:** Can you achieve $h_{\text{min}}$ for part (b) using an English word that is a permutation of $R, E, L, A, T, I, O, N, S$?

**Solution:** INSERT YOUR SOLUTION HERE

(e) (4 points) Give an example of a 2-3-tree $T$ with exactly 5 leaves and two distinct values $a, b$, such that one gets different final trees if one first inserts $a$ into $T$ and then deletes $b$, instead of first deleting $b$ from $T$ and then inserting $a$. Explain what happened.

**Solution:** INSERT YOUR SOLUTION HERE