A rooted tree $T$ is a set of nodes storing elements in a parent-child relationship with the following properties:

- If $T$ is nonempty, it has a designated root node, which has no parent.
- Every node $v$ other than the root has a unique parent
- Every node with parent $w$ is a child of $w$.

A binary search tree is a type of rooted tree. If the tree is non-empty, then the first node is a root of the tree (drawn at the top for some reason). You might think of a binary search tree like a linked list with two “next” pointers: one for the list that has only smaller values and one for only larger values. We call those “left” (smaller) and “right” (larger) pointers in this context.

**Question 1.** Starting with an initially empty binary search tree, insert the keys 50, 25, 75, 60, 55, 90, 65, 37

**Question 2.** From the following tree, what is the result if we delete node 62? What if we instead deleted 17? 78?

![Tree Diagram]

**Question 3.** Write a function that will output each key of a binary search tree from smallest to largest. You may assume you start with a pointer to the root.
**Question 4.** Write a function that will output each key of a binary search tree. Do so in such an order that if I insert each element of your output into an initially empty binary search tree, the result will be a tree that matches the tree we started with.

**Question 5.** Write code to evaluate a mathematical expression, represented as a *syntax tree*, such as the one below. A syntax tree will be a binary tree, but not a binary *search* tree. You may assume a function `eval(operator, left, right)` that evaluates any mathematical operator and `lookup(variable)` that looks up the value of a variable. You may also assume that if you call `lookup` with a constant as a parameter, it returns that constant.

```
+  /
/+-
/ / 3
x y 4
```

**Question 6.** I have a binary (not a binary search tree) with the following properties:

- Each non-null\(^1\) node of T contains a single character

*This is a reminder that this is not a binary search tree.*

Draw the tree. Briefly explain (1-2 sentences) how you decided that is the tree that meets this criteria.

**Reinforcement**

After the first lecture, you should be able to define and use the following terms with respect to a binary search tree: parent, child, sibling nodes, leaf node, internal node, in-order traversal, pre-order traversal, post-order traversal. You should be able to describe the procedure to insert a new key into a binary search tree, how to delete from a binary search tree, and how to search a binary search tree for a particular key. Given a list of keys, you should be able to insert them into an initially empty binary search tree.

\(^1\)Depending which textbook you are reading, this either means “all the internal nodes” or “all the internal nodes AND all the leaf nodes,” so I am trying to be less ambiguous by saying every non-null node.