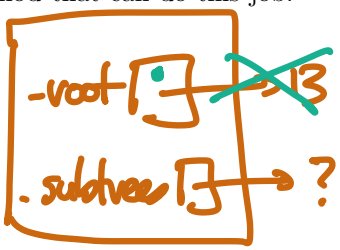
CSC148 - Tree Deletion Algorithms

We've seen that when deleting an item from a tree, the bulk of the work comes when you've already found the item, that is, you are "at" a subtree where the item is in the root, and you need to delete it. Let's write a method that can do this job. Here's its API:

```
def delete_root(self) -> None:
    """Remove the root of this tree. Precondition: this tree is not empty."""
```



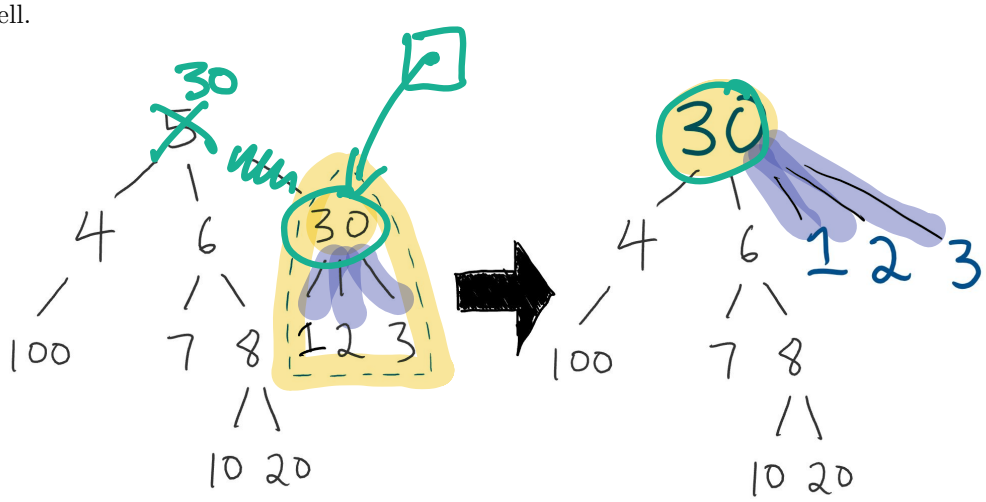
1. We can't just set the `_root` attribute to `None`. Why not?

   no, would violate RIs.

   Instead, we will give `_root` a new value from somewhere else in the tree. Let's look at two different ways we can do this.

2. **One Strategy: "Promoting" a subtree**

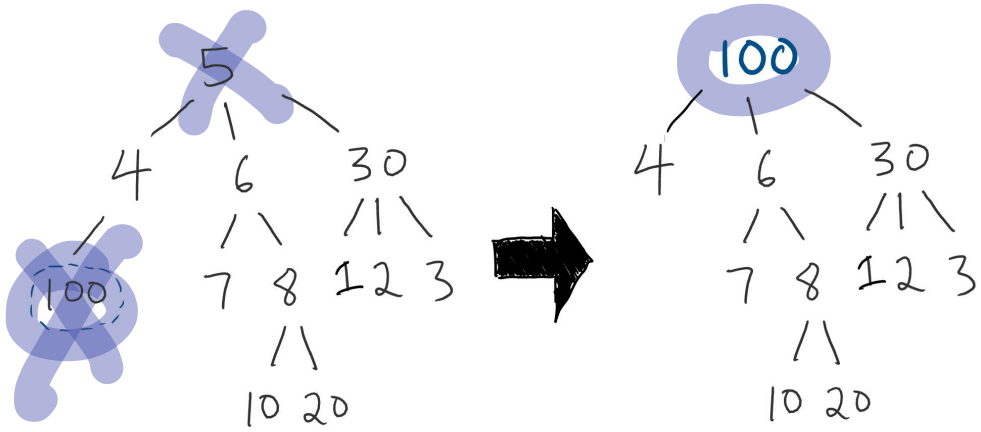
   Idea: to delete the root, take the rightmost subtree t_1, and make the root of t_1 the new root of the full tree, and make the subtrees of t_1 become subtrees of the full tree. Note that we could have promoted the leftmost subtree, or any other subtree, just as well.



   Implement the method this way.

3. **Another Strategy: Replacing the root with a leaf**

Idea: to delete the root, find the leftmost *leaf* of the tree, and make the leaf value the new root value. (No other values in the tree should move.) Note that we could have replaced the root with the value in rightmost leaf, or any other leaf, just as well.



Now implement the method this way.

This is left as an exercise, because it will help you with A2.

Note: Recursion is not needed, just some careful updates.

I will post the code after you've had a chance to try it.