M9: Post Lab

After you have successfully completed your AVL tree code, answer the following questions.
First, for each of the methods you implemented, describe briefly (in three sentences or less) how it works. Did it use any instance variables of the class, and if so, which ones? Are any helper methods called by the method (either of your own creation or in the original codebase), and if so, which ones and what do they do?

The methods that you should describe are:

1. `private void updateHeight(TreeNode<T> root);`
2. `private int getBalance(TreeNode<T> root);`
3. `private TreeNode<T> rebalance(TreeNode<T> root);`
4. `private TreeNode<T> rightRotate(TreeNode<T> root);`

(Again, I won't ask you to repeat yourself by giving the same info for `leftRotate()`.)

5. Briefly describe the modifications you made to the insertion and removal functions to maintain height and keep the tree balanced. Where did you initialize the height of a new node?

For questions 6-8, consider the following proposed extension to our `AVLTree` class. We'd like to add a method `T firstAfter(T v)` that, given a value v, returns the least element of the set that is $\geq v$. If no such element exists, the method should return a special value "notFound". v itself may or may not be in the set.

6. Suppose we implement `firstAfter()` as a top-down tree walk, similar to `find()`. What should we do if the root of the tree has a key $< v$? Justify why the behavior you specify is correct.
7. If the root has a key $k \geq v$, what should we do to determine whether k is the *least* key $\geq v$? Justify why the behavior you specify is correct.
8. Based on your answers above, give pseudocode for an implementation of `firstAfter()` that runs on a BST in time proportional to its height.