1. private void updateHeight(TreeNode<T> root);

- Parameters:
  - root: Receive a tree node as a parameter and update the height of the parameter node. Getting a height of a tree is max(left node height, right node height) + 1. This function literally does that. It also checks when the parameter is null value.
- Return value:
  - Void. There is no return value.

2. private int getBalance(TreeNode<T> root);

- Parameters:
  - root: Receive a tree node as a parameter and calculate a balance factor of the received node. Getting a balance factor is (left node height - right node height). It also checks when the parameter is null value.
- Return value:
  - Returns a balance factor of the input node.

3. private TreeNode<T> rebalance(TreeNode<T> root);

- Parameters:
  - root: Receive a tree node as a parameter and this method analyzes which rotation is needed for an input node. Before carry out with the rotation, this calls updateHeight() function to set the input node's height right. And then it judges which case (left, right-left, left-right, right rotation) is proper to the input node and calls the proper function to rebalnce.
- Return value:
  - Returns a rebalanced node.

4. private TreeNode<T> rightRotate(TreeNode<T> root)

- Parameters:
  - root: Receive a tree node as a parameter and this method carries out a right rotation of the node. After right rotation it re-calculates a height of the switched nodes.
- Return value:
  - Returns a right rotated node.

5. What is the formula for computing the height of a tree's root node, assuming the heights for its left and right subtrees are known?

- Height of a tree's root node = max(left node height, right node height) + 1

6. What should the height be for a newly allocated leaf node? For an empty subtree?

- A newly allocated leaf node: Height should be 0.
- An empty subtree: Height should be -1.

7. Why do the insert() and remove() methods declare recursive helper functions, rather than themselves being recursive?

- The Insert and remove function receive only a T value as a parameter. But helper functions take two input parameters as you can see in the source code. We can add up needed input parameters in the helper function so to make it easier to use the 'insert, remove' functions. Because users who want to call 'insert, remove' function only gives one parameter calling the functions and no need to worry about where is the root variable of the tree.

8. Here is one line from the insertHelper() method: root.setLeft(insertHelper(value, root.left));
Explain why the root's child pointer must be updated with the result of the call to insertHelper().

- It's impossible to connect the inserted node with the tree if there is no connection between the inserted node and a root's child pointer. The 'setLeft, setRight' functions carry out a connection to make a link between the inserted node and the root node. Moreover, the insertHelper also calls the insertHelper function recursively, the 'setLeft, setRight' function gives a mean to follow up to the root node.