To test that all my methods worked properly in all cases, I went and tried example code to see whether the AVL tree would break or not show the right value in any case. I first checked that the errors for adding duplicate elements or removing elements not in the tree would return when running, which happened during my unit testing. I also checked for inserting and removing elements that would cause changes in the balance and force the tree to rebalance itself. I checked cases that had double right rotations, double left rotations, right rotations, and left rotations, and multiple rotations operating together to make sure that the rotations weren't coming out in the wrong way. To make sure that the rotations were correct, I did the test cases by hand first, and checked the preorder and postorder strings printed with my own to make sure that they were accurate. Since unique nodes can't have preorder, inorder, and postorder sequences all be the same, this let me check my work on whether my AVL tree functioned properly.

To show that the fraction class worked, I made a list of unordered fraction values, then put all the values into a BST. Since the inorder representation of the tree of fraction was correct, and adding duplicate elements failed, as well as removing elements not in the tree, this meant that the comparison methods I had written worked. Since the inorder representation was correct, that that the greater than and less than functions worked since the elements were ordered properly. Since I couldn't add duplicate elements, this let me know my method for checking equality was correct. Finally getting an error for removing an element not in the tree let me know that the comparison functions fully worked, even for values not in the tree, meaning that my fraction class worked completely correctly.