In order to ensure that the insert_element method works correctly, I created a series of test cases in which I inserted varying amounts of elements into trees. I inserted one element and five elements into empty trees, respectively. In the first test case, I tested whether the root of the tree, originally None, changes to the element being inserted. In the next test case, I inserted five into the tree in order to test whether the insert_element method was adding the element at the right location in the tree. These test cases were then duplicated and used for the pre-order and post-order traversals. Because all three traversals returned the expected outcome, I was able to conclude that the method functions correctly.

In order to ensure that the remove_element method works properly, I created three test cases in which I removed various elements from a tree. In the first test case, I removed the root of a tree with two elements in order to test whether the remaining element would become the new root. In the next test, I removed the root of a tree with two children to test whether the tree allocated the two surviving elements into the correct positions. In the next test case, I removed five elements from a tree with ten elements in order to test the ability of the tree to properly balance itself while removing a series of values. I then duplicated these test cases for the pre-order and post-order traversals. Because all three traversals returned the expected outcome, I was able to conclude that the method functions correctly.

In order to determine whether the get_height method functions correctly, I created four test cases. In the first test case, I tested whether the height of an empty tree is zero or not. I tested the get_height method in the next test case by inserting an element into an empty tree and determining whether insert_element updated the height of the tree. In the next test case, I tested the height of a tree with two children in the scenario where one child is greater than the root and the other child is smaller than the root. In the last test case, I tested the height of the tree after five insertions and three removals.

In order to ensure that the to_list method works correctly, I created two test cases in which I changed an empty tree and a tree with 5 elements into a list by calling the to_list method. In the first test case, I called to_list in order to change the empty tree into a list. I then tested whether to_list was adding the inserted elements at the correct locations in the list in the next test case. Because both tests returned the expected outcome, I was able to conclude that the to_list method works properly.

In order to test whether the less than, greater than, and equals methods work correctly, I created six test cases with three unique fractions. In the first two test cases, I test the less than method by comparing ½ to ⅓. I then test the greater than method in the next two cases, evaluating the fractions ½ and ⅓. Finally, I tested the equals method in the last two test cases by comparing the fractions ½, ¼, and 2/4. Because the method returns True when comparing ½ and 2/4 and false when comparing ½ and ⅓, I was confident that the method was working properly. Because all six test cases yielded the expected outcomes, I was able to conclude that all three methods function correctly.