# Testing

to_list testing:

The to_list method was tested with an empty tree, multiple trees with heights up to 4, and after different insertions and deletions. I am sure it is fully tested because it is supposed to sort the values in increasing order, with essentially the same order as the in order traversal. The tests proved that in every one of these scenarios, the to_list method was functioning appropriately.

Insert_element testing:

The insert_element method was tested from an empty tree to a tree with a height of 4. With multiple tests with different trees, it inserted the node to the correct place, which was double checked with me building a tree on paper to get the correct output from all 3 traversal methods. If the insertion was incorrect, or any of its components was incorrect, when building a tree up to a height of 4, the traversals output would be incorrect at some point. With the way I was inserting numbers, the method required multiple rotations including both left and right, so after checking the traversals at multiple levels with tree variations, I can confidently say that the insert_element correctly inserts the input every time and is fully tested. I also tested adding a duplicate number and a ValueError was raised appropriately.

Remove_element testing:

The remove_element method was tested with removing a nonexistant value from the tree, and it was tested with removing values from multiple heights of the tree, with random orders, such as removing a root with subtrees on both sides to ensure that the tree had to undergo rotations. With multiple tests with different configurations for the AVL tree, I can say that the deletions were correct every time and appropriate rotations took place. I also made sure to remove elements in an order that would require rotations in both left and right. Through this testing, I was able to confirm (assertEqual) my handwritten tree with all 3 traversal methods. If the remove_element was wrong in any form, the resulting traversals would be incorrect, which did not happen.

Fraction testing:

For testing fraction.py, I did informal tests in the main section of the code. I followed the instructions regarding making an array with fraction values and printed the unsorted and sorted list. And they were functioning correctly.

Then I did tests for all three functions, greater than, less than, and equal to. I made sure to include cases where the output should be false or true, depending on the function. For example, for the less than method, I made sure that ½ < 1/3 was false, and that ½<13/2 was true. I did this with more values for less than. And with the greater than function, I did the same thing, except in the opposite way. ½ > 1/3 was true, and ½ > 13/2 was false. I also made sure that the equal function worked where if the numbers were not equal it would be false, and if the numbers were equal it would be true. I also tested adding a fractional value that was equal to a value in the tree, but with different numerator and denominator and it raised the appropriate ValueError. All these tests made me 100% confident that this works perfectly.