Lab 4 Report: AVL Trees

Introduction

This lab was so far the most intensive in terms of actual learning as I had never experienced AVL trees before. The material from lecture definitely helped with the lab, but I will admit I had to read over the AVL tree section in the textbook several times in order to fully grasp the concept of this data structure. In a general discussion, as with all recursion, it does a simple task on many levels, and this is hard to conceptualize.

Program Structure

For this program, I implemented an AVLTree class, and modified my Node and Queue classes from previous labs in order to accommodate the new data structure. More specifically, the Nodes were given a height variable which was used to calculate tree height differences.

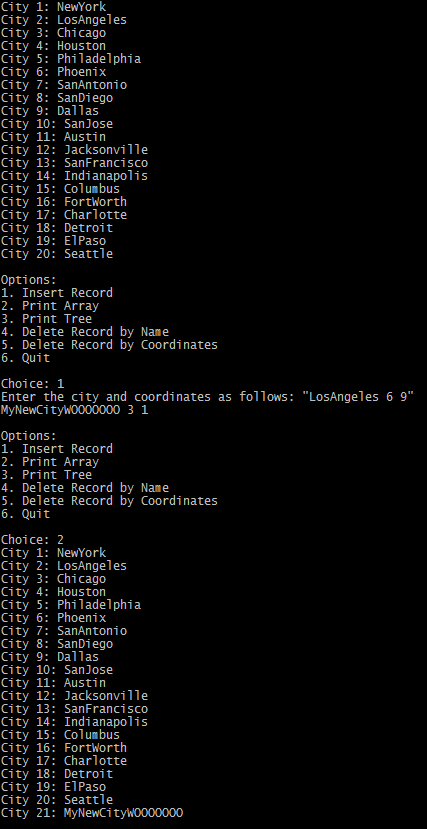
I had to make the left, right, and parent pointer nodes public in order to effectively pass them by reference, which was integral to my insertion/deletion algorithms. This was used together with dereferencing pointers in order to modify the actual pointer object, which was vital in my insertion and deletion methods.

In the main.cpp file of the program, I first load my data file into the unordered array as well as the AVLTree, I then print out both data structures to show proper functionality. Next, there’s a while loop which handles user input so the user can input their own city values (along with coordinates).

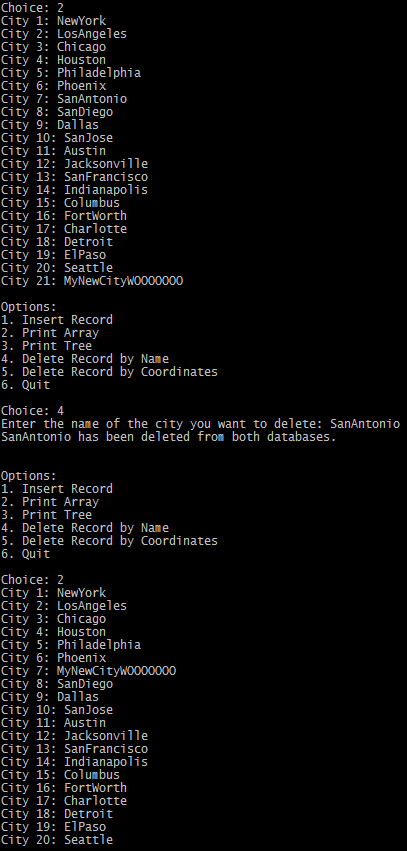
Testing

Testing the unordered array can be done through the console. Since no complex modification is required, we can easily see that inserting to, and deleting from an unordered array works (deletion by name or by coordinates):

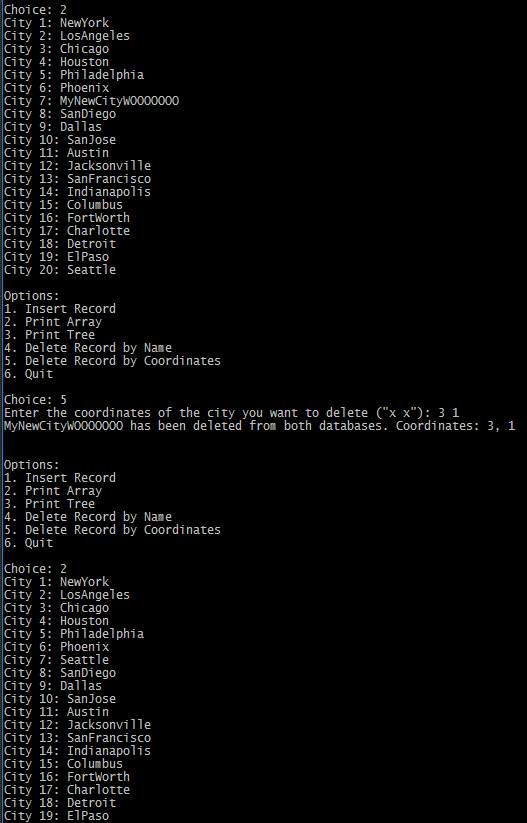
Inserting (unordered array)



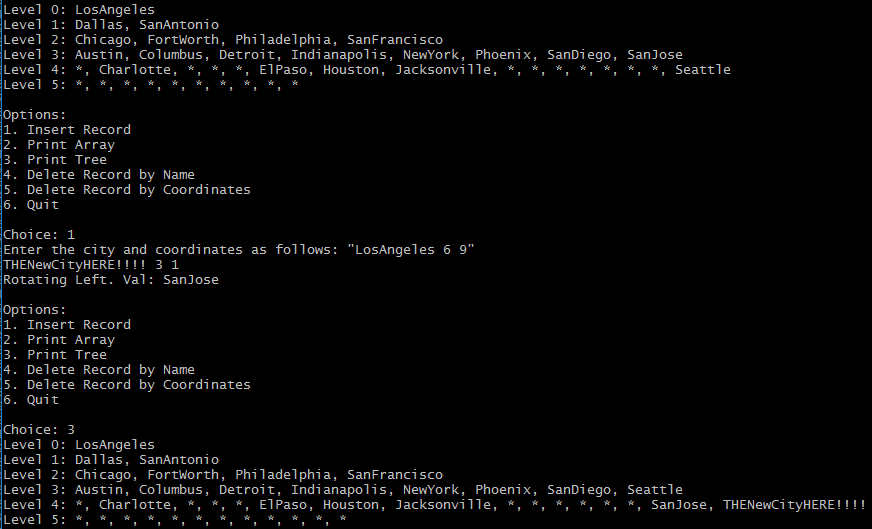
Deleting by name (unordered array):



Deleting by coordinates (unordered array):

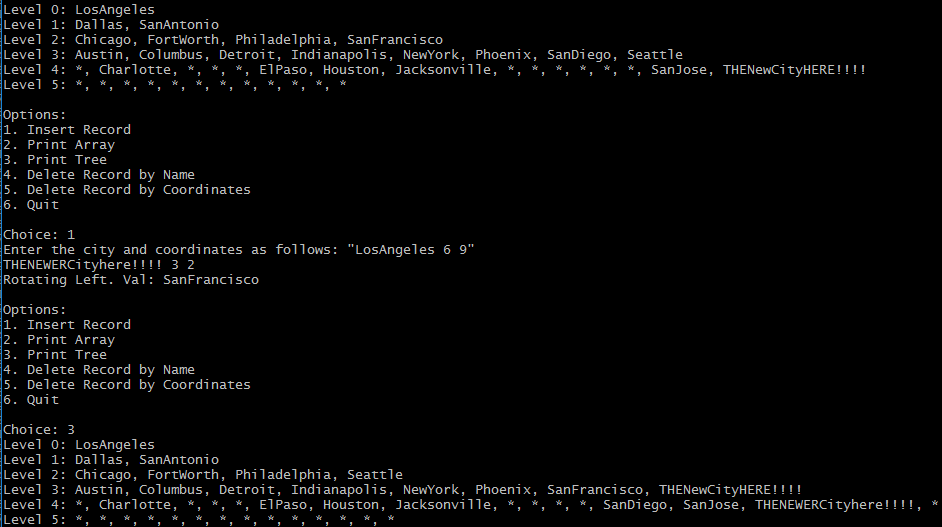


Inserting (AVL Tree)



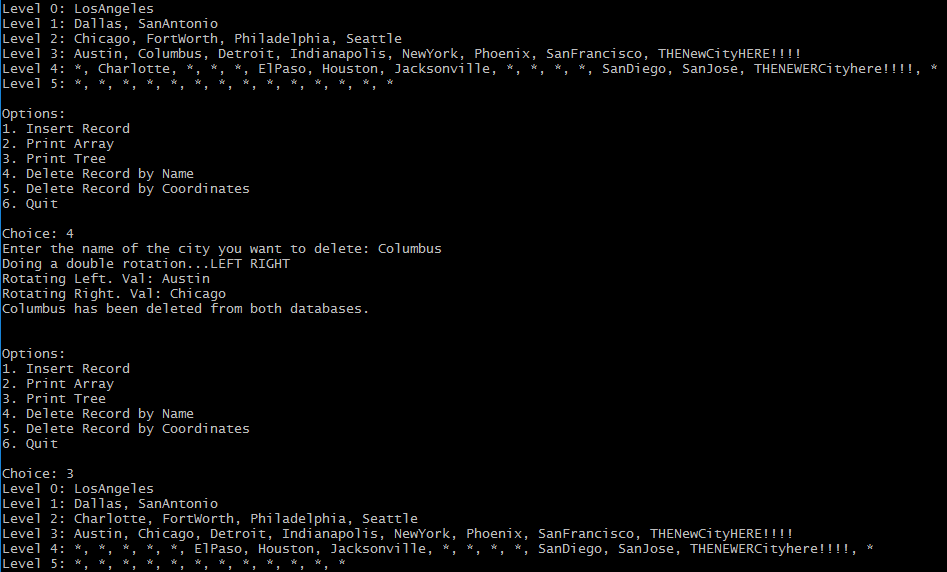
Looking at this insertion, we can see that the balance mechanism works as it should. I inserted “THENewCityHERE!!!” Which would place itself to the right of Seattle. Upon examining the tree, we can see that the tree starting at “SanJose” became unbalanced as a result of the insertion. Since it’s a RR child, a left rotation must be performed. After the rotation has occurred, SanJose has shifted down to its left child, Seattle has shifted up, and THENewCityHERE!!! has taken its correct spot.

Inserting (AVL Tree) – Continued



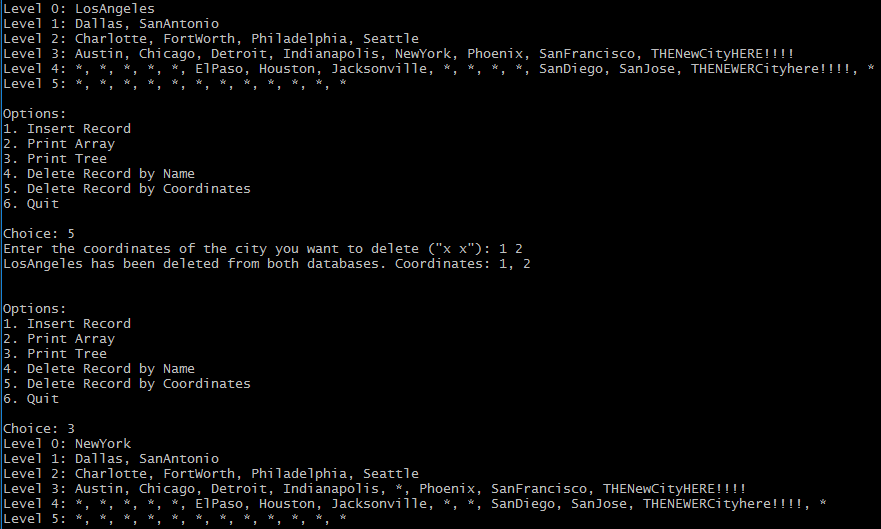
Shown here, I have added yet another city that causes the subtree with root “SanFrancisco” to become unbalanced. This insertion causes Seattle to shift up, SanFrancisco to shift to its left child, and this shoves SanDiego down to be siblings with SanJose. Lastly, we can see the newly inserted city has found its home on the very right side of the tree (since it’s the ‘largest’ value).

Deleting by name (AVL Tree)



I chose to delete Columbus to show a double rotate. We can see from looking at the first, unaltered tree, removing Columbus would create a left right imbalance, which requires two rotations, first shift Austin Left, then shift Chicago right. This is clearly demonstrated in this insertion as the new node structure has Charlotte as the root, Austin as the left child, and Chicago as the right child.

Deleting by coordinates (AVL Tree)



This deletion goes through the same process as the above deletion, as it simply searches the tree for the coordinates that were input, finds the node, and then deletes the node using the name stored in the found node. In this case, the coordinates 1, 2 belonged to the city of LosAngeles, so the search function was performed on these coordinates and returned the Node containing LosAngeles. Then, the same delete function was carried out on that Node, which keeps the program consistent.

We can see that by deleting the root its in order successor had to replace it, which was NewYork. Since NewYork was a leaf, no rotations had to be carried out.

Conclusion

To conclude, I believe I have adequately demonstrated my program’s functionality. I’m also confident that the unordered array functions correctly as well. As always, if there are any questions or concerns, please email me at jayofferdahl@ku.edu.