Running Head: AVL Tree Fixes Henry 1

**Introduction**

The problems that this program is trying to solve are an extension of our AVL lectures. In this program, there are two functional requirements. The first is to fix a bug wherein only the first node of a tree printed, while the subsequent nodes are blank. The second is to print the balance factor of the nodes. These two functional requirements will allow the programmer to be able to better understand the bugs that are currently present within the currently implemented AVL tree.

(blue = filled field by author, yellow = pre-filled)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** |  | **Functional Requirement** |  | **Value** | **Stakeholder** |
| FID001 | **I want to** | Fix an issue where only the first node is printed and the rest are blank | **so that** | the student developer is able to identify and correct bugs to make the program run as intended | Student |
| FID002 | **I want to** | Print the balance factor of each node of the tree | **so that** | The person running the program may see that the tree is being correctly balanced. | Student/instructor |

**Analysis**

The first functional requirement make is a simple mistake. Within the setBalance() function there needs to be an if statement nested within the for each loop. This is because the nodes that are created after the root of the entire tree have a value of null. The tree ends up being balanced in this case, but it does not fulfill the requirements of bulk inserting data and having the tree automatically be balanced. The second functional requirement was implemented by using a new function printBalancePreOrder() that calls a private function printNodeBalancePreOrder() with root as the argument. This function uses the number curated from the setBalance function, which will subtract the height of the left from the height of the right. If a node gets inserted or deleted into the AVL tree, then the program will make use of the rebalance algorithm to automatically make sure none of the tree’s balances are less than -1 or greater than 1.

**Conclusion**

With these functional problems solved, a use can expect an array of data to be inserted, and correctly inserted into a tree while displaying the balance factor of each node. This is the correct way the program was meant to run. This assignment was a bit and a half for me. I was able to find the error in the code, but I’m not exactly sure why the fix works the way it does. Other than that hiccup, I really feel like I understand AVL trees pretty well, and being able to have something be automated like this is a really neat feeling. Changing the code in the AVLTree function to be a series of parents, oldRT, newRT, left and right really made the entire thing much easier to grasp than going back and forth between the idea of a, b, and c. This kind of tree makes me curious as to what else I can automate to this extent.