CS 562
Qi Wang
ID: 932-439-151

Proposal For Testing Project

For this testing project, I will use the Template Scripting Testing Language(TSTL) to test a Python program called avltree.py. This program implements a self-balancing AVL tree. An AVL tree is an ordered node based tree key structure in which each node has at most two children, and the heights of each children of a node differ by at most one. The source of this program is in the following link: https://github.com/TylerSandman/py-bst/blob/master/build/lib/pybst/avltree.py.

There are multiple main functions in this program: insert, delete, get_balance, rotate_left, rotate_right. I will run general two steps to test this program. The first thing is that I will firstly test all main functions and ensure all main functions work correctly. Next step is to input different type values to the function to check if the program return error or compatible with other data type.

First of all, I will test the insert function. The insert function can insert a new node with key attribute and value attribute into the tree, and it is necessary to keep the tree balance. So if it success, it will return correct balanced tree. So in order to test insert function, I will input an invalid string to check if the program can return false. Then I will insert a valid type of value, if it returns the correct balanced AVL tree.

Then I will test the delete function. The delete function can delete the node with key attribute from the tree. It can delete a leaf or a parent of leaf. First of all, I will test this function and try to delete a leaf of the tree. If it runs successfully, the leaf will be deleted and then also it returns a balanced AVL tree. Then, I will also try to test if it works correctly to delete a node with one child. If it successes, it also returns a correct

balanced AVL tree. Finally, I will try to test deleting a node with two children node to check if it works correctly.

Next, I will test the get_balance function. Get_balance function can produce the balance of the tree. It defined as the height of the right subtree taken away from the height of the left subtree. In this function, I will test if the heights of each children of a node differ is at most one.

Then for the rotate_left function, it can perform a left tree rotation in the tree around the pivot node. For the rotate_right function, it can do a right tree rotation in the tree around the pivot node. In order to test these functions, I will delete a node to make it run the rotate_left function and rotate_right function. So I need to check if they work successfully and return a correct balanced AVL tree. I will also insert some nodes to the tree, to let it run the rotate_left and rotate_right functions, then check if the result of the tree is a correct balanced AVL tree.