Lab 3 was the lab where I had the most amount of files connected together. The main method was on its own, the AVLTree object was put together with its Node, and the RBTree object had its own separate Node. Aside from that I had the word file provided, an empty .txt file that the code would write on, and a shorter word file for the purpose of testing.

The AVLTree and RBTree objects were taken out of xzybooks, as indicated, and modified according to what the main method required. Such as, changing the type of node data from int to str (for the fist key) and float (for the rest of the keys). I also added a height method to the AVLTree object that was previously missing.

In the main method, the code indicates a .txt to read lines from, each line with a word followed by a series of float numbers from 1 to -1. If a line contained a “word” that was a symbol, it would not be included in the following list created from the file. There was then a new file created from only the accepted words, without the float numbers. The user was then asked which tree they wanted to implement with the list of words. If the user chose RBTree, then a separate tree for each word would be created, with the nodes being the float numbers provided, in ascending order. If the user chose AVLTree, then a tree with all the words would be created. Each node would contain a list with the word and its float numbers. Both methods calculated the height of the trees as well as the number of nodes.

The thing that I’m missing from this code is the method that calculates the relationship between two words. I was ready to use the search method in the AVLTree in order to find the word and its values. Then use the comparison equation to compare the two. My problem was not understanding the comparison equation or how to implement it. But I was ready.

I certify that this project is entirely my own work. I wrote, debugged, and tested the code being presented, performed the experiments, and wrote the report. I also certify that I did not share my code or report or provided inappropriate assistance to any student in the class.

*from* RedBlackTree *import* RedBlackTree  
*from* AVLTree *import* AVLTree, Node  
*import* re  
  
  
*############# RED-BLACK TREE ###############  
def* RBT\_method(keys):  
  
 *# Create an RedBlackTree object and add the values* tree = RedBlackTree()  
  
 *for* i *in* range(len(keys)):  
 key = keys[i]  
 *for* j *in* range(1,len(key)):  
 value = key[j]  
 tree.insert(float(value))  
  
  
 num\_nodes = len(tree)//len(keys)  
  
 *# Print number of nodes  
 print*("The number of nodes on this RB tree is: ", str(num\_nodes))  
 *# Print height  
 print*("The height of this RB tree is: ", str(tree.get\_height()))  
  
*############# AVL TREE ###############  
  
def* AVLT\_method(keys):  
 *# Create an empty AVLTree object.* tree = AVLTree()  
 simple\_tree = AVLTree()  
  
 *# Insert keys into the tree.  
 for* key *in* keys:  
 *#print(key)* node = Node(key)  
 tree.insert(node)  
  
 *# Simpler tree to understand  
 for* key *in* keys:  
 *#print(key[0])* node = Node(key[0])  
 simple\_tree.insert(node)  
  
 *# Print the tree after all inserts are complete.  
 print*("Tree after initial insertions:")  
 *print*(simple\_tree)  
  
 *# Print number of nodes  
 print*("The number of nodes on this AVL tree is: ", str(len(keys)))  
 *# Print height  
 print*("The height of this AVL tree is: ", str(tree.get\_height()))  
  
 k = input(' What depth would you like to search? \n')  
 *print*()  
 simple\_tree.get\_depth(k)  
  
*############## MAIN #################  
  
# Read file and create list sans symbols  
#with open('glove.test.txt', 'r') as textFile:  
with* open('glove.6B.50d.txt', 'r') *as* textFile:  
 lines= textFile.readlines()  
 new\_lines = []  
  
 regex = re.compile('[a-zA-Z]')  
  
 *for* i *in* range(len(lines)):  
 line = lines[i].split(' ')  
 line[len(line)-1] = line[len(line)-1].rstrip("\n")  
 *if* re.match(regex, line[0]):  
 new\_lines.append(line)  
  
*with* open("newFile.txt", "a") *as* new\_file:  
 *for* i *in* range(len(new\_lines)):  
 *#print(new\_lines[i])* new\_file.write(new\_lines[i][0])  
 new\_file.write(" \n")  
  
  
*# Ask user for RBTree or AVLTree*user\_choice = input(' 1 FOR RED-BLACK TREE\n 2 FOR AVL TREE\n')  
*print*()  
  
*if* user\_choice == "1":  
 RBT\_method(new\_lines)  
*elif* user\_choice == "2":  
 AVLT\_method(new\_lines)  
*else*:  
 *print*('INVALID')  
  
user\_choice = input(' What depth would you like to search? \n')  
*print*()