**CS 2302 Lab 3 Report**

**Edgar Escobedo**

For this lab, I decided to go for option B, which would focus in storing words and finding the permutations of these words. In order to complete the lab, we were requested to read a file containing all the words of the English dictionary, we would have to separate the words per line and store these words inside of one of two trees. These trees would be an AVL tree or a red black tree, the option to choose one would be given to the user. Once we had the English words inside the trees, the user would choose a word and we would show him/her all the anagrams of that word. To find the anagrams, we would run the word with the show\_anagrams method, which would create all the permutations of the word, and by using either the AVL or red black tree we would check if that permutations gave an actual word. Lastly, we would read a new file and using a new method we would find which word in that file had the greatest number of anagrams, after this, we would inform the user.

To solve this lab, I first added the red and black tree and AVL tree from the zybooks implementation, from there I made a few changes in order to function in the way I wanted it to, this small changes were mostly in the search method, which in the previous case returned the a node with the word we were searching, but in my case I just needed to know if the word existed in the tree, so a Boolean variable was enough. Then I read the file containing all the words and then depending in what tree the user decided to use, would enter each word into the tree. For the AVL tree I would create a node with that word and enter it to the tree and for the red black tree I only need to enter the word into the tree. After loading all the words into the tree, in my interface I would ask the user what tree they wanted to use, and load it once, then using the show\_anagrams, which I changed in order to show a counter of the number of anagrams the word had, not actually printing them, I would search for as many words the user wanted to search for.

When the user decided to stop looking for words, he/she could simply type “3”, and the first part of the program, which was to look for the anagrams of a word, would terminate. To solve the second part, I read a new file, and created a new method which would store the word, and then use again that word in the show\_anagrams method. In this method I would get the number of anagrams this word had and store this information along with the word. In the end I would have a list with all the words along with their number of anagrams and just traverse once the list in order to find the word with the maximum number of anagrams. In each iteration I would check the number of anagrams of the word being checked now with a “max” variable that would store the number of maximum anagrams now, and the word that had that maximum, showing at the end the maximum number along with the word.

In order to test my program, I decided to create smaller test text files that would be introduced into the AVL and the red black tree. After inserting the words, I would traverse certain extends of the trees to see if the trees were loading in a proper manner. When the trees were checked, I would run the trees and search for the anagrams of word, having in mind that all the English vocabulary was not inside the trees, and check if the number of anagrams of that word matched what was expected. Finally, for the last part of the file, again I would run smaller test text files, but this time using a tree having all the English vocabulary, and only seeing if the word with the greatest number of anagrams was given as an answer. Next some graphs will be given for the test cases.

From the graphs, one using the AVL trees and the other using the red and black tree to look for 3, 6, 9 and 12 words we get those results. As the graphs show, the red and black tree is much faster to insert the words into the tree, in that way the whole execution of the search is reduced. In conclusion after implementing and seeing how both the AVL tree and the red and black tree works I found both very useful, but in this case the red and black tree being more efficient throughout the search. In the second part I learned more about the use of the method as the show\_anagrams and how it could be implemented to other uses if necessary, the overall idea of the lab being very insightful for my future projects.

**Appendix:**

**https://github.com/ejescobedo/Lab-3**

I, Edgar Escobedo, 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.