**I honor Parkland's core values by affirming that I have followed all academic integrity guidelines for this work.**

**Evelyn Hosana**

It is most unfortunate that the AVL class provided along with all other classes (in other .py files) have provided far too many errors in which my initiator would not accept my word storage into dictionary, thus providing difficulties completing my coding assignment including the requirements. Hence, to at least get something working and turned in, I'm hoping for a few points, even if it’s for this summary as to how my code works or bits and pieces of my code. Respectfully, by receiving the same error past multiple days and hours of a completed code just not working for me, I decided this was a better route to take and further manipulate existing code for use.

The following is a detailed written explanation of my program and its components' explanations.

The website I've developed allows users to input text into a textarea and checks for misspelled words. It leverages an AVL tree data structure to efficiently store and search for words in the dictionary that Ken Urban provided (located at /home/staff/kurban/public/lists/web2.txt). The AVL tree is a self-balancing binary search tree that maintains a balanced structure by automatically adjusting the positions of nodes during insertions and deletions.

In the code, the AVLTreeMap class represents the AVL tree. It consists of AVLNode objects, which store the keys (words) in the tree. Each AVLNode has references to its left and right child nodes, as well as its height in the tree.

When the website loads, the dictionary file is read, and each word is inserted into the AVL tree using the insert method. The insert method maintains the balance of the tree by performing rotation operations if necessary. This ensures that the tree remains balanced, and the lookup operations are efficient.

When a user submits the form with their text, the program splits the text into words and checks each word against the AVL tree using the in operator. If a word is not found in the AVL tree (indicating it is misspelled), it is added to the misspelled\_words list. Finally, the program outputs the list of misspelled words, or a message indicating that no misspelled words were found.

The AVL tree provides an efficient way to store and search for words in the dictionary due to its self-balancing property. The balancing operations guarantee a maximum height of O(log n), where n is the number of words in the dictionary. This results in faster lookup times compared to unbalanced binary search trees.

By leveraging the AVL tree data structure, the website can quickly identify misspelled words from the user's input, helping to improve the accuracy of their writing.