LAB PROJECT

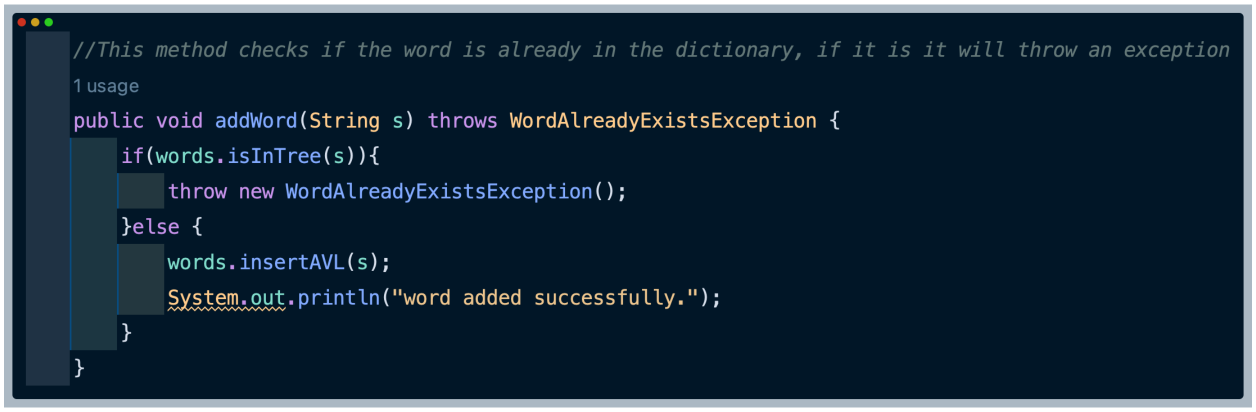
ABDULRAHMAN ALABDULKARIM
202164170

Introduction

This project involves creating a dictionary data structure enabling users to insert, search, and delete, find similar words. To achieve optimal efficiency for these operations i used an AVL tree, as their complexity for these tasks is O(log n) which is most optimal with the data structures we can use.

Methods

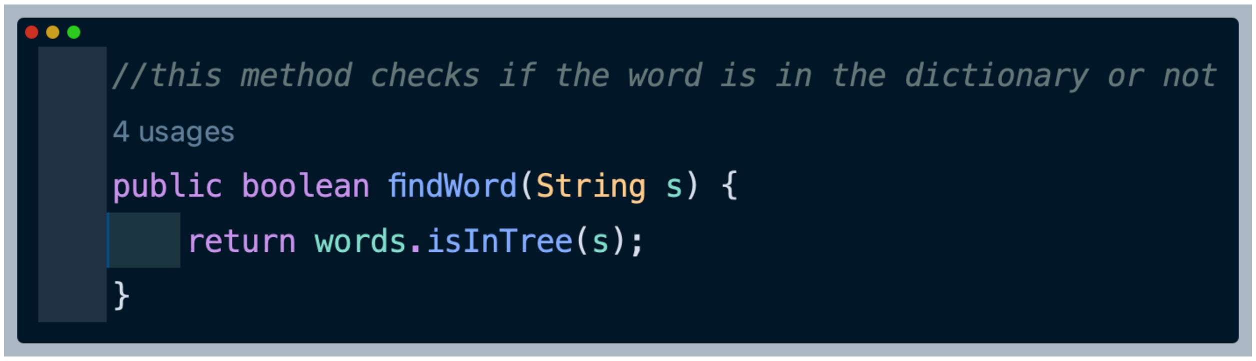
Addword



```java
//This method checks if the word is already in the dictionary, if it is it will throw an exception
1 usage
public void addWord(String s) throws WordAlreadyExistsException {
    if(words.isInTree(s)){
        throw new WordAlreadyExistsException();
    }else {
        words.insertAVL(s);
        System.out.println("word added successfully.");
    }
}
```

For this method, I decided to use the premade insertion method of the avl tree data structure to add the word. This method is the most optimal for adding words.

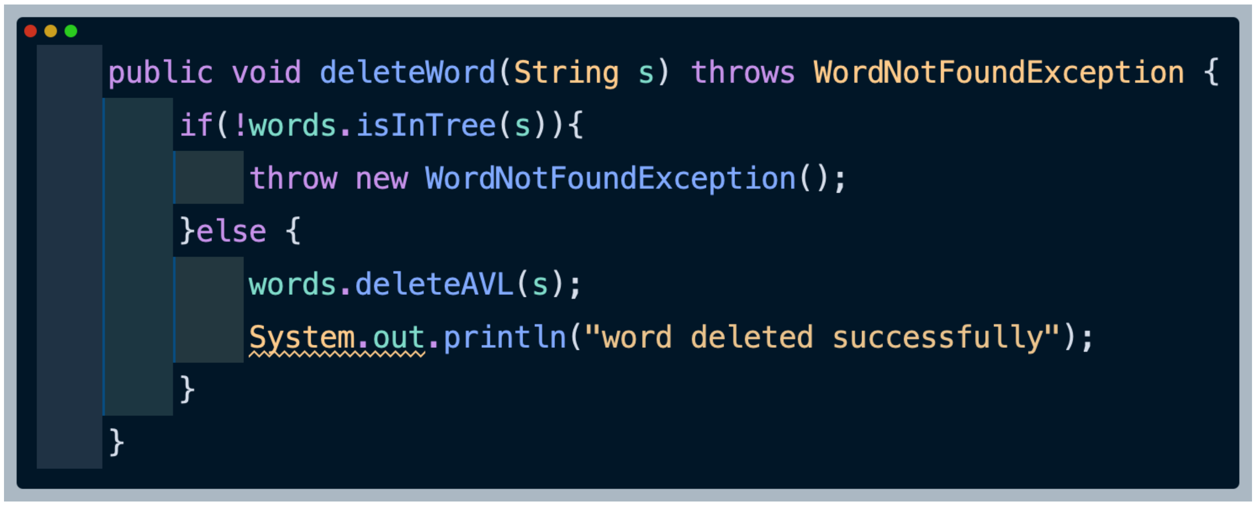
Find word



```java
//this method checks if the word is in the dictionary or not
4 usages
public boolean findWord(String s) {
    return words.isInTree(s);
}
```

For this method I also utilized the pre existing AVL tree method of isInTree. Which will check if the word is in the tree and will return a Boolean which is what gets returned to findWord

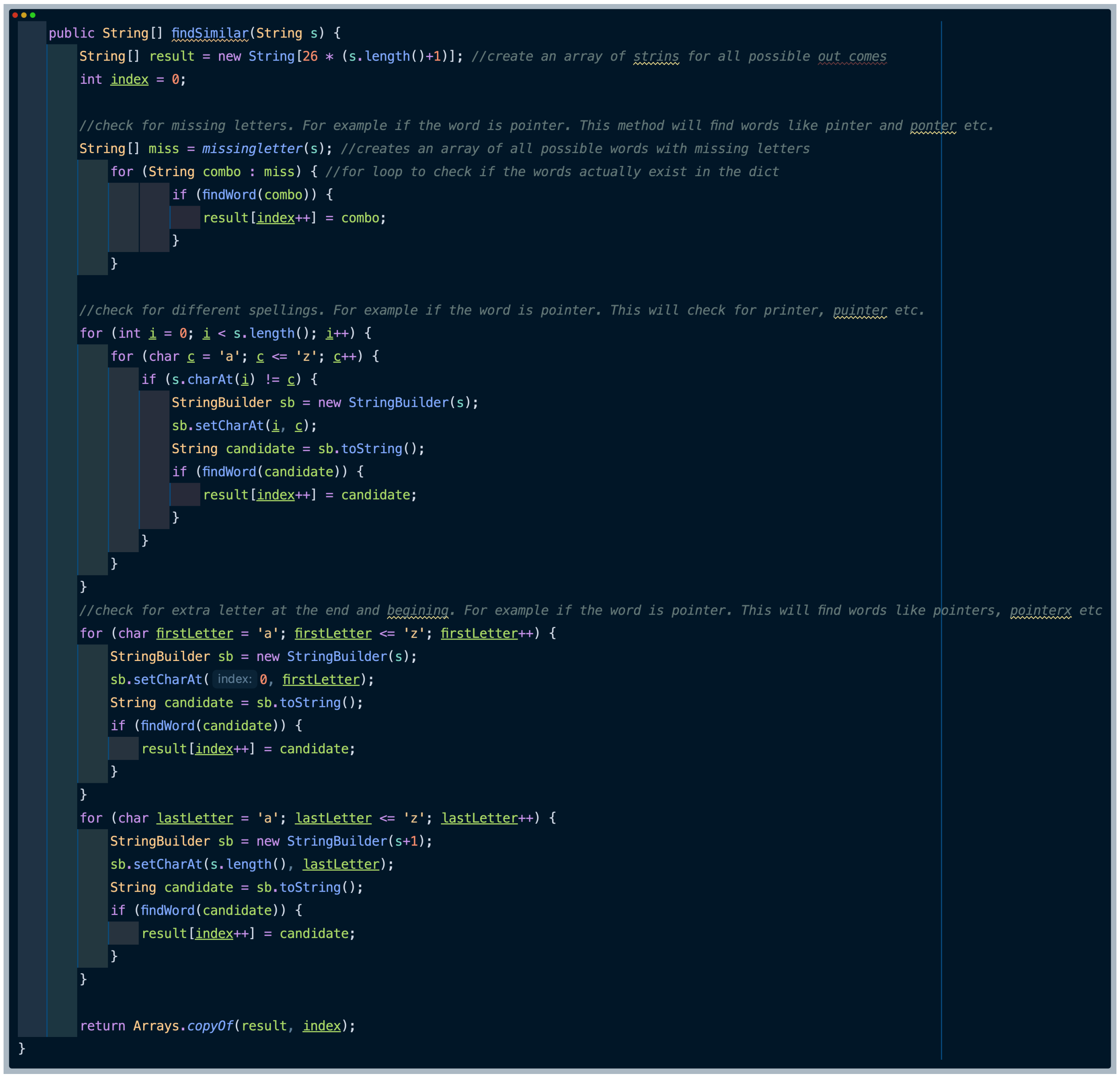
Delete word



```java
public void deleteWord(String s) throws WordNotFoundException {
    if(!words.isInTree(s)){
        throw new WordNotFoundException();
    }else {
        words.deleteAVL(s);
        System.out.println("word deleted successfully");
    }
}
```

For this method I utilized a similar approach to the add word method, but instead of .insertavl. I used .delete avl

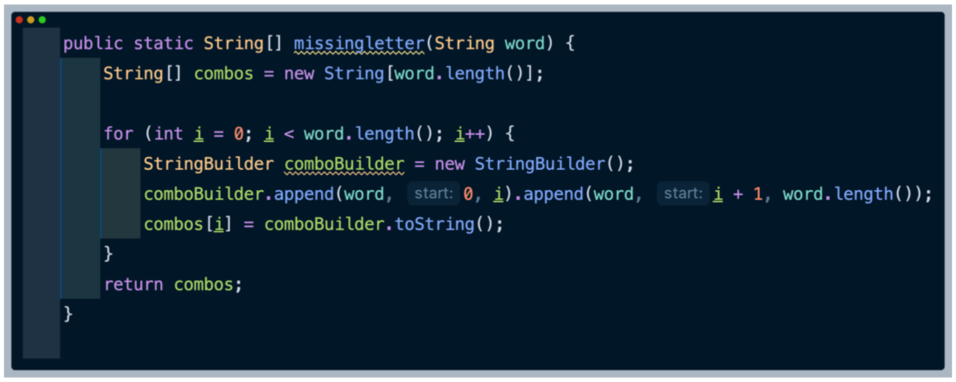
Find similar method



```java
public String[] findSimilar(String s) {
    String[] result = new String[26 * (s.length()+1)]; //create an array of strins for all possible out comes
    int index = 0;

    //check for missing letters. For example if the word is pointer. This method will find words like pinter and ponter etc.
    String[] miss = missingletter(s); //creates an array of all possible words with missing letters
    for (String combo : miss) { //for loop to check if the words actually exist in the dict
        if (findWord(combo)) {
            result[index++] = combo;
        }
    }

    //check for different spellings. For example if the word is pointer. This will check for printer, puinter etc.
    for (int i = 0; i < s.length(); i++) {
        for (char c = 'a'; c <= 'z'; c++) {
            if (s.charAt(i) != c) {
                StringBuilder sb = new StringBuilder(s);
                sb.setCharAt(i, c);
                String candidate = sb.toString();
                if (findWord(candidate)) {
                    result[index++] = candidate;
                }
            }
        }
    }
    //check for extra letter at the end and begining. For example if the word is pointer. This will find words like pointers, pointerx etc
    for (char firstLetter = 'a'; firstLetter <= 'z'; firstLetter++) {
        StringBuilder sb = new StringBuilder(s);
        sb.setCharAt( index: 0, firstLetter);
        String candidate = sb.toString();
        if (findWord(candidate)) {
            result[index++] = candidate;
        }
    }
    for (char lastLetter = 'a'; lastLetter <= 'z'; lastLetter++) {
        StringBuilder sb = new StringBuilder(s+1);
        sb.setCharAt(s.length(), lastLetter);
        String candidate = sb.toString();
        if (findWord(candidate)) {
            result[index++] = candidate;
        }
    }

    return Arrays.copyOf(result, index);
}
```

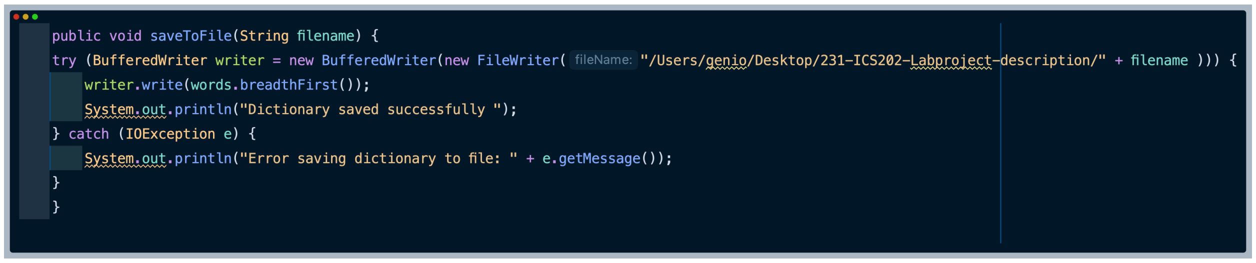
This method took me the most time. I decide to solve it step by step. First I went ahead with using brute force to find every possible word out code. By creating an array of alphabet then changing each char of the string and then searching if the new string is actually a word. After finishing the main algorithm. I made some more adjustments to find special cases like, words where there is a missing letter. Where I decided to solve this by creating a helper method that removes each letter of the string and then adding that word to an array of words which I then search for in the dictionary. I did the same for the case where it adds an extra letter and beginning and end

Helper method used



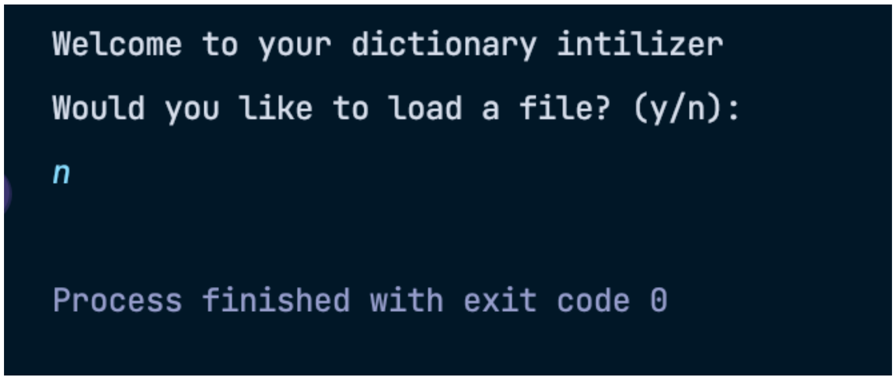
```java
public static String[] missingletter(String word) {
    String[] combos = new String[word.length()];

    for (int i = 0; i < word.length(); i++) {
        StringBuilder comboBuilder = new StringBuilder();
        comboBuilder.append(word, start: 0, i).append(word, start: i + 1, word.length());
        combos[i] = comboBuilder.toString();
    }
    return combos;
}
```

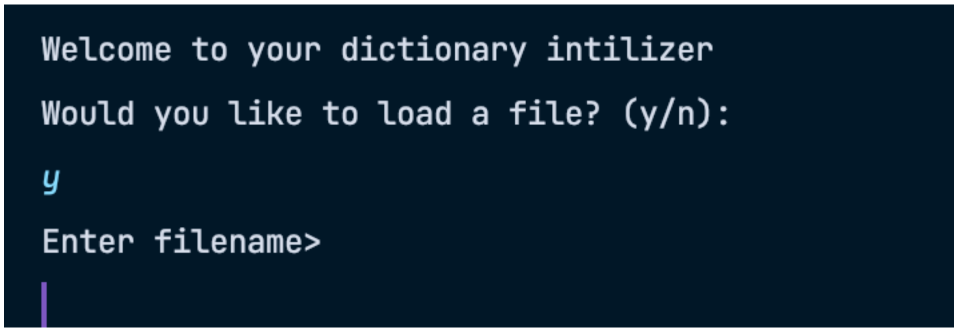
Save file method

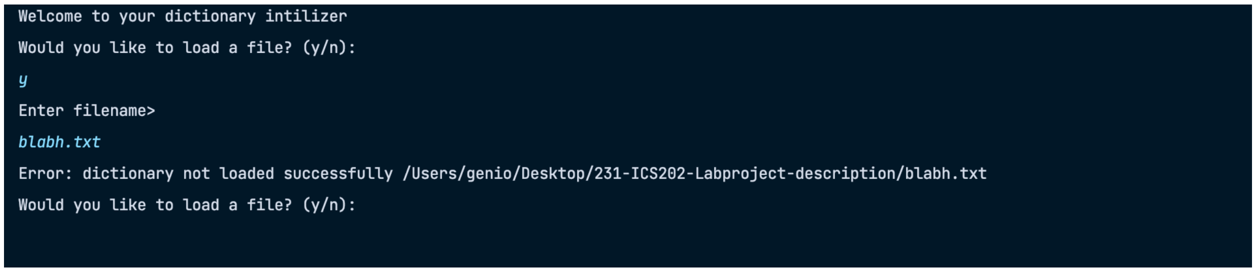

```
public void saveToFile(String filename) {
try (BufferedWriter writer = new BufferedWriter(new FileWriter( fileName: "/Users/genio/Desktop/231-ICS202-Labproject-description/" + filename ))) {
    writer.write(words.breadthFirst());
    System.out.println("Dictionary saved successfully ");
} catch (IOException e) {
    System.out.println("Error saving dictionary to file: " + e.getMessage());
}
}
```

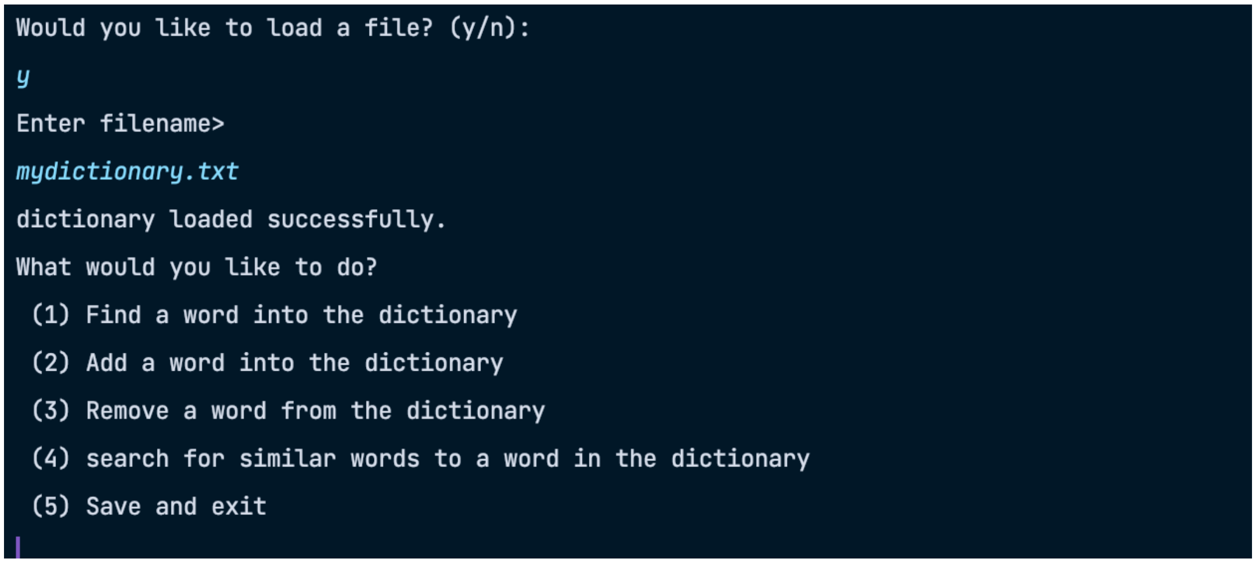
This method uses breadthfirst to traverse the tree and adds each word to the new dictionary place

Tests

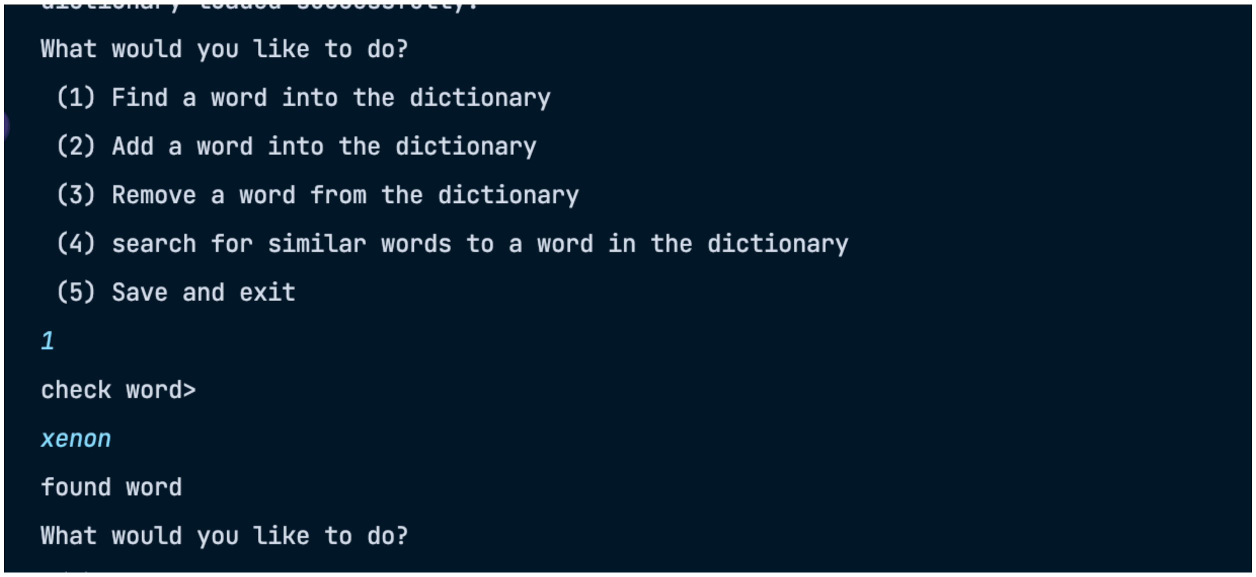

```
Welcome to your dictionary intilizer
Would you like to load a file? (y/n):
n


Process finished with exit code 0
```


```
Welcome to your dictionary intilizer
Would you like to load a file? (y/n):
y
Enter filename>
```

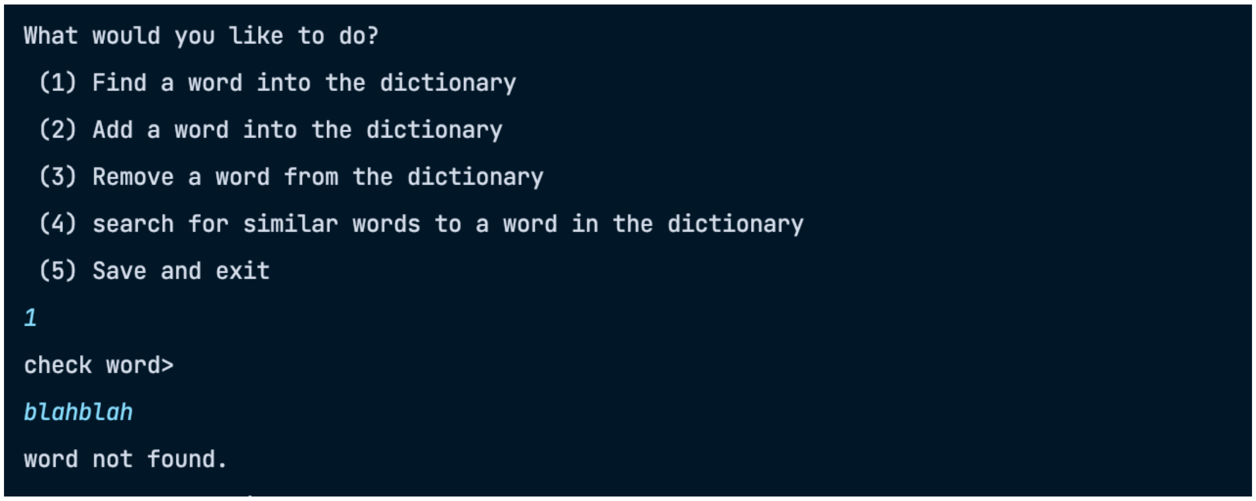

```
Welcome to your dictionary intilizer
Would you like to load a file? (y/n):
y
Enter filename>
blabh.txt
Error: dictionary not loaded successfully /Users/genio/Desktop/231-ICS202-Labproject-description/blabh.txt
Would you like to load a file? (y/n):
```



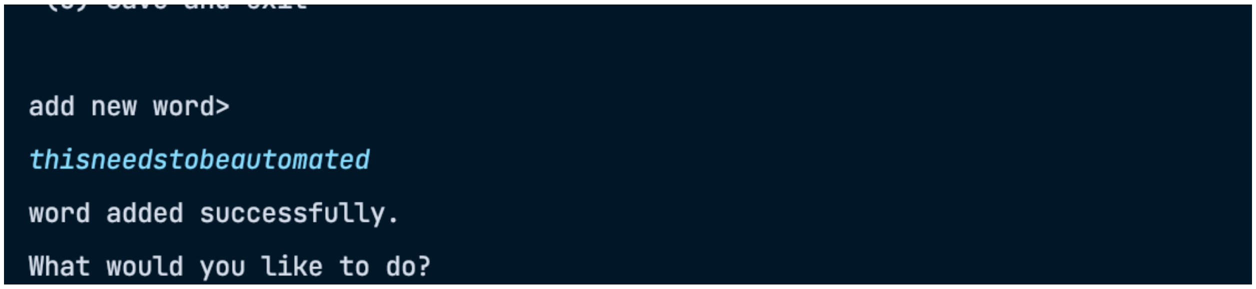
```
Would you like to load a file? (y/n):
y
Enter filename>
mydictionary.txt
dictionary loaded successfully.
What would you like to do?
 (1) Find a word into the dictionary
 (2) Add a word into the dictionary
 (3) Remove a word from the dictionary
 (4) search for similar words to a word in the dictionary
 (5) Save and exit
```



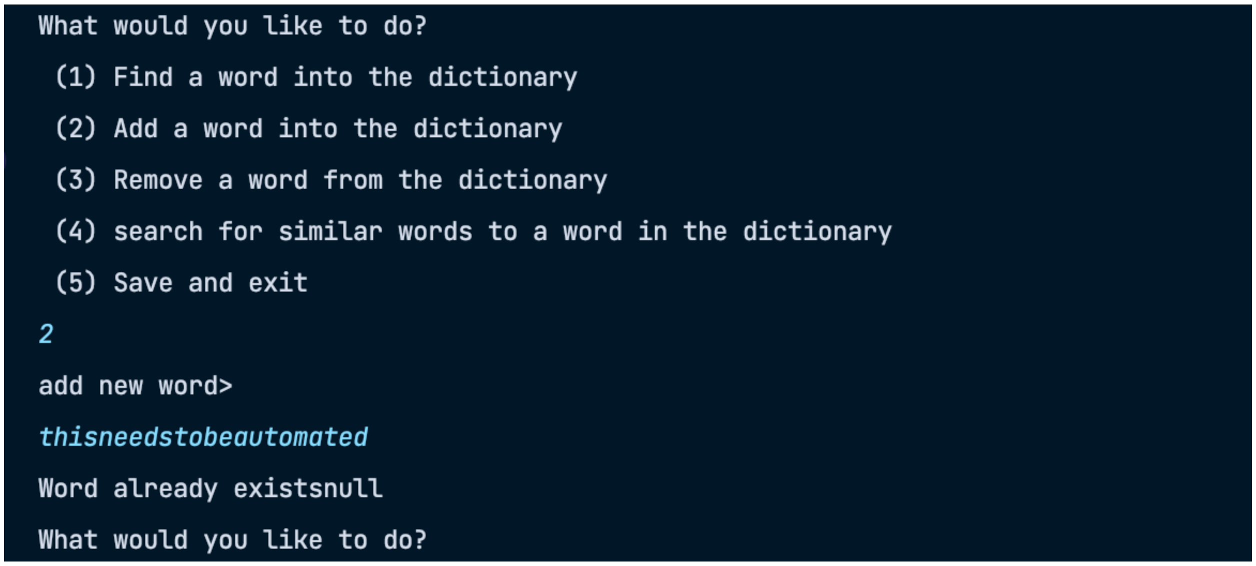
```
What would you like to do?
 (1) Find a word into the dictionary
 (2) Add a word into the dictionary
 (3) Remove a word from the dictionary
 (4) search for similar words to a word in the dictionary
 (5) Save and exit
1
check word>
xenon
found word
What would you like to do?
```



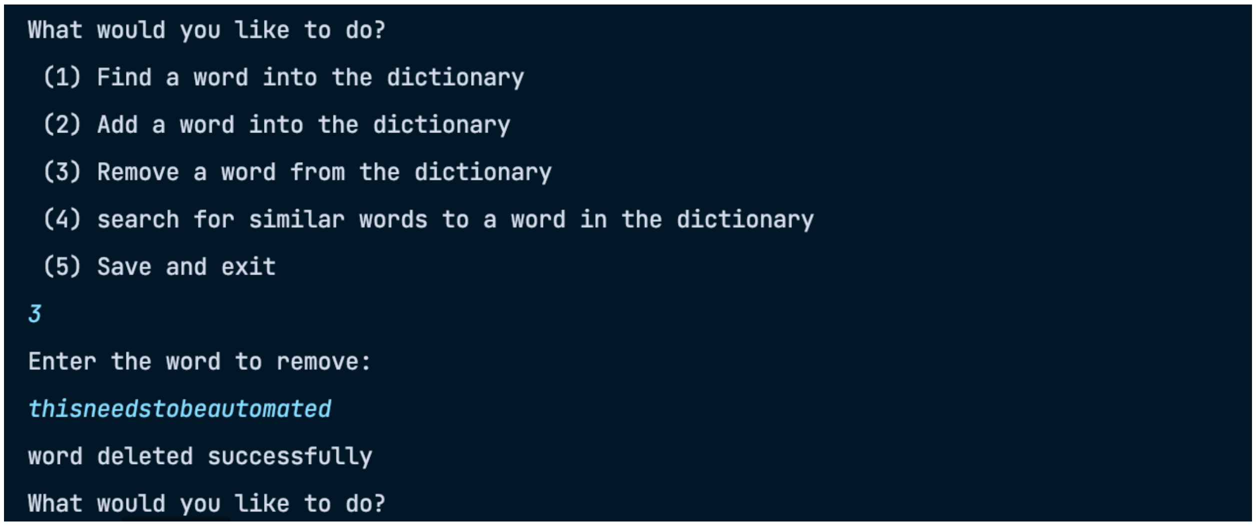
```
What would you like to do?
 (1) Find a word into the dictionary
 (2) Add a word into the dictionary
 (3) Remove a word from the dictionary
 (4) search for similar words to a word in the dictionary
 (5) Save and exit
1
check word>
blahblah
word not found.
```



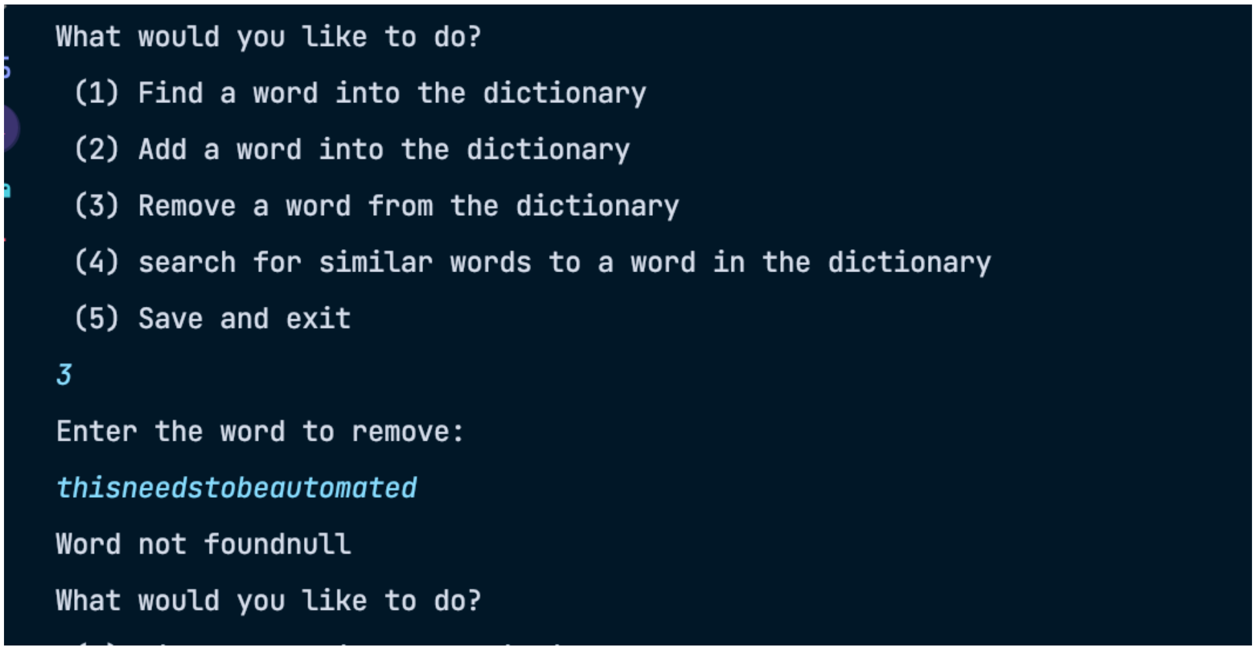
```
add new word>
thisneedstobeautomated
word added successfully.
What would you like to do?
```



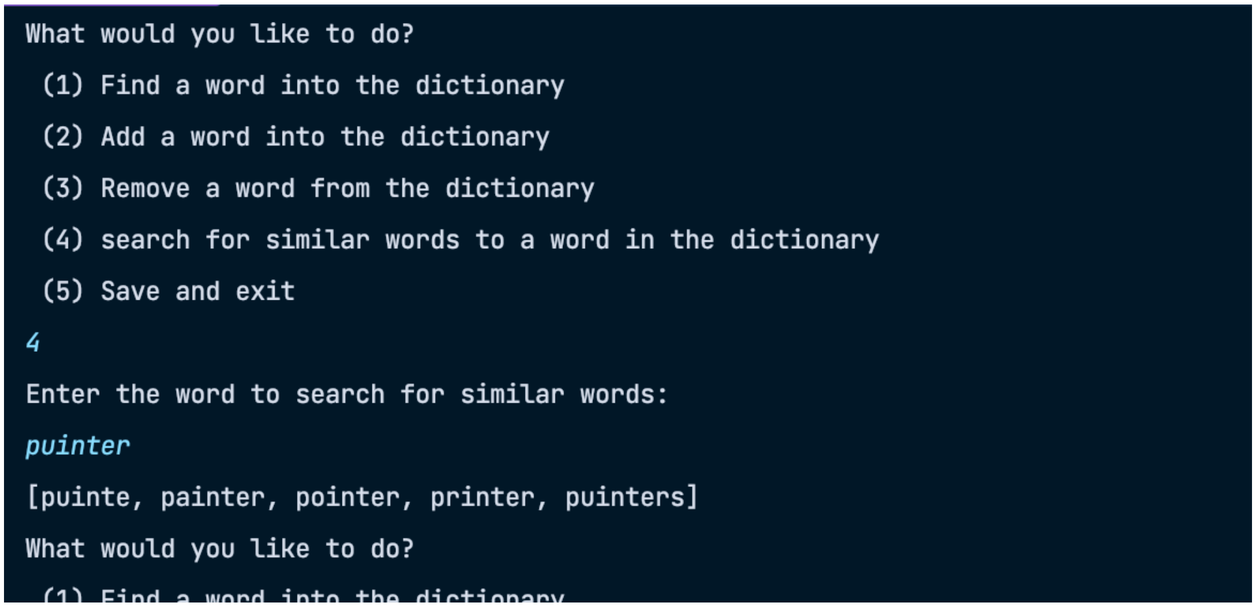
```
What would you like to do?
 (1) Find a word into the dictionary
 (2) Add a word into the dictionary
 (3) Remove a word from the dictionary
 (4) search for similar words to a word in the dictionary
 (5) Save and exit
2
add new word>
thisneedstobeautomated
Word already existsnull
What would you like to do?
```



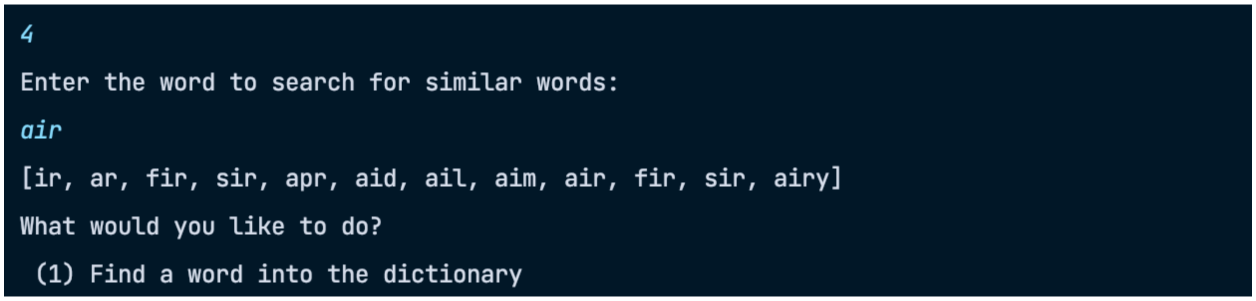
```
What would you like to do?
 (1) Find a word into the dictionary
 (2) Add a word into the dictionary
 (3) Remove a word from the dictionary
 (4) search for similar words to a word in the dictionary
 (5) Save and exit
3
Enter the word to remove:
thisneedstobeautomated
word deleted successfully
What would you like to do?
```



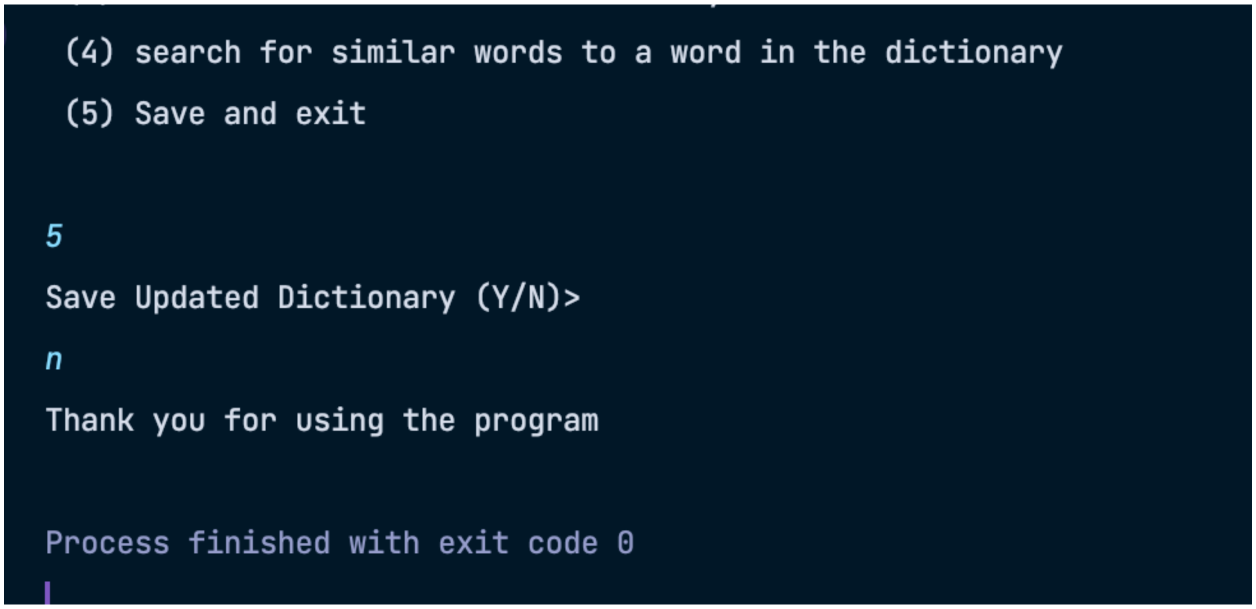
```
What would you like to do?
 (1) Find a word into the dictionary
 (2) Add a word into the dictionary
 (3) Remove a word from the dictionary
 (4) search for similar words to a word in the dictionary
 (5) Save and exit
3
Enter the word to remove:
thisneedstobeautomated
Word not foundnull
What would you like to do?
```



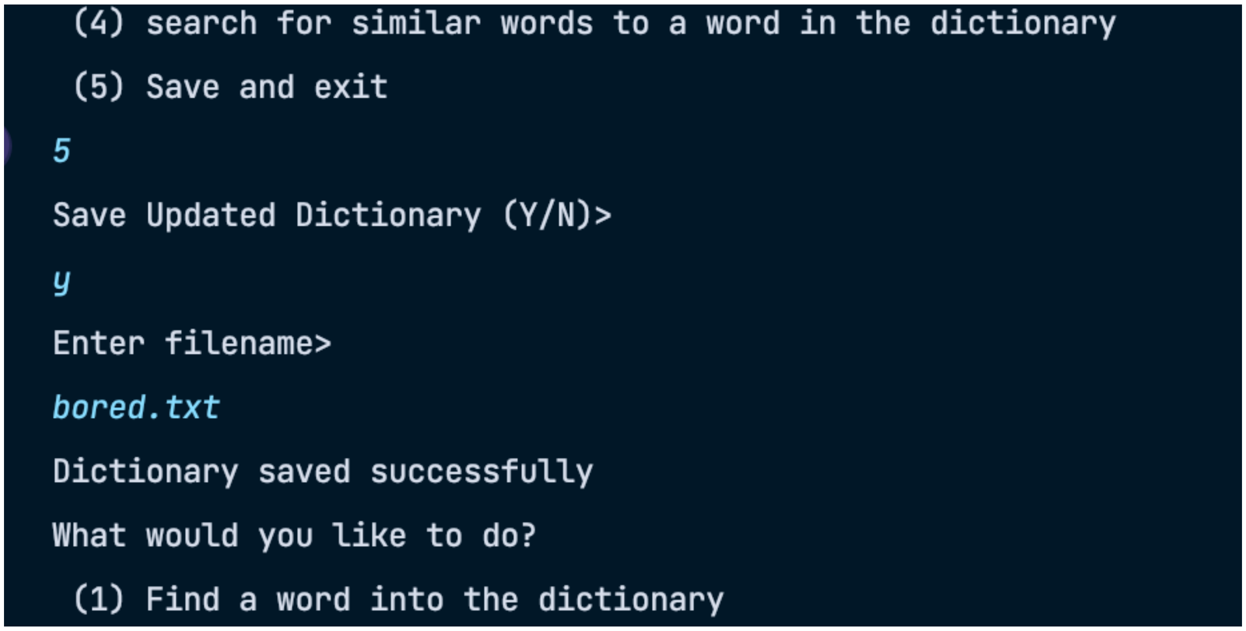
```
What would you like to do?
 (1) Find a word into the dictionary
 (2) Add a word into the dictionary
 (3) Remove a word from the dictionary
 (4) search for similar words to a word in the dictionary
 (5) Save and exit
4
Enter the word to search for similar words:
puinter
[puinte, painter, pointer, printer, puinters]
What would you like to do?
 (1) Find a word into the dictionary
```



```
4
Enter the word to search for similar words:
air
[ir, ar, fir, sir, apr, aid, ail, aim, air, fir, sir, airy]
What would you like to do?
 (1) Find a word into the dictionary
```



```
 (4) search for similar words to a word in the dictionary
 (5) Save and exit


5
Save Updated Dictionary (Y/N)>
n
Thank you for using the program


Process finished with exit code 0
```



```
 (4) search for similar words to a word in the dictionary
 (5) Save and exit
5
Save Updated Dictionary (Y/N)>
y
Enter filename>
bored.txt
Dictionary saved successfully
What would you like to do?
 (1) Find a word into the dictionary
```

Challenges faced

I initially created a comprehensive solution involving a tree structure for letters, with each letter containing an AVL tree for words. However, I later simplified the project as I realized my initial approach was overly complex, considering that creativity wasn't a key factor. I'll be preserving my original solution, refining it further, and sharing it on GitHub. The most challenging part was the "find similar" method, which I managed to resolve by breaking it down into smaller operations. Although I encountered additional difficulties in my primary solution, I won't delve into those details since they are not the focus of our discussion.