PROJECT REPORT: DICTIONARY

DATA STRUCTURES

Contents

Introduction

The aim of this project is to design and implement a dictionary using a Trie data structure. The dictionary provides functionalities such as updating words, adding new words, deleting words, and suggesting words based on a given input prefix. The choice of a Trie data structure is justified by its efficiency in handling dictionary-related operations.

Trie Data Structure

A Trie, also known as a digital tree or prefix tree, is a tree-like data structure that is used to store a dynamic set of strings. Each node in the Trie represents a character in a word, if the node is not null than it means that the character exists and if the node is null than the character doesn't exists . The root is empty. The path from the root to a particular node where, Boolean variable endOfWord is true, forms the word corresponding to that node.

```
struct Node
{
public:

    Node* alphabets[26]; //each node having 26 further nodes
    bool isEndOfWord; //a flag for isEndOfWord of the string
    string meaning;
    Node();
};
```

The same thing could have been done with Hash Table, but Trie Tree can perform these operations more efficiently and Trie can also be used for prefix-based searching (as used word suggestion function) whereas it cannot be done in Hash Table or AVL.

Advantages of Trie

1. In Tries the keys are searched using common prefixes. So, it is faster.
2. We can efficiently do prefix search with Trie, whereas in AVL, Hash Table, BST it is not efficient and it difficult.
3. We can easily print all words in alphabetical order which is not easily possible with hashing.
4. Tries take less space when they contain many short strings. As nodes are shared between the keys.
5. Finding a data in a Tire is faster in worst case as compared to imperfect hash table.
6. Insertion's, Deletion's, and Searching's Time Complexity is **O(Length of String),** which is better than BST or AVL (as Trie don't require rotations).

Time Complexity

Operation	Trie Tree
Insert	**O(L)**
Delete	**O(L)**
Search	**O(L)**
Update Word	**O(L)**
Suggest Word	**O(N*L)**

L is the length of the string, N is number of words in trie. So, it is faster as compared to other data structures.

Functions

```
void ___Color()
```

```
printf("\033[1;31m");     //redColor
printf("\033[1;32m");     //greenColor
printf("\033[0m");        //whiteColor
printf("\033[1;30m");     //grayColor
```

```
Node();
```

Default constructor of Node class. It initializes `bool isEndOfWord` as false, `string meaning` as empty and `Node* alphabets[26]` as nullptr.

```
Dictionary();
```

Default constructor of Dictionary. It loads the Dictionary Data from file to Tree.

```
void mainMenu();
```

This Function is the mainMenu. This is used to access dictionary action. It takes input from the user on which operation to perform and performs the action.

```
void lowerCaseString(string& str);
```

This Function is used to convert the string in lower case as the word APPLE and apple are same in dictionary.

```
bool isNonAlpha(const string str);
```

This Function is returns true if the string contain's non alphabet character.

```
bool deleteWord(string word);
```

This function is used to delete the Word from the File. If the Word is found it calls the `void deleteWordFromFile(string word)` function and deletes the word and return true else it will return false.

```
void deleteWordFromFile(string word);
```

This functions find the word in file and count the line no on which the word is present than it calls the `void deleteLineFromFile(const string& filePath, int deleteLineNo)` function and send the file name and the line number to delete.

```
void deleteLineFromFile(const string& filePath, int deleteLineNo);
```

This function takes the file name and line no to delete than it creates new file and copies all the data from the original file to new file except the line no which contains the word that has to be deleted than it rename the new file and deletes the old file.

```
bool insertWordInTree(string word, string meaning);
```

This functions insert a new word in Tree in the tree and then calls the `void insertWordInFile(string word, string meaning)` function to insert that word in file and return true. But if the word it already present in Tree than it asks to update it's meaning, if Yes than `void updateMeaning(string word, string newMeaning)` function is called to update meaning.

`void insertWordInFile(string word, string meaning);`

The function inserts the word and meaning in file.

`Node* searchNode(string word);`

This function returns the Node pointer to the node where the given word is ending. Then it is used in deleting Word, suggesting Word and Updating Word Functions.

`string searchMeaning(string word);`

This function searches for the word and if the word is found then it returns its meaning or else it returns empty string.

`void searchWordAndPrintMeaning(string word);`

This function calls the `string searchMeaning(string word)` function and prints the word with its meaning.

`void updateMeaning(string word, string newMeaning);`

This function searches word with the help of `Node* searchNode(string word)` if the word is found it deletes the old Word from file using `bool deleteWord(string word)` function, than updates the meaning of the word and inserts the word in Tree and File using `void insertWordInFile(string word, string meaning)` function.

`void suggestWord(string prefix);`

This Function find the Node of prefix where endOfWord is true and passes it to `void suggestWord(Node* curr, string prefix, int& count)` function to print 10 word suggestions .

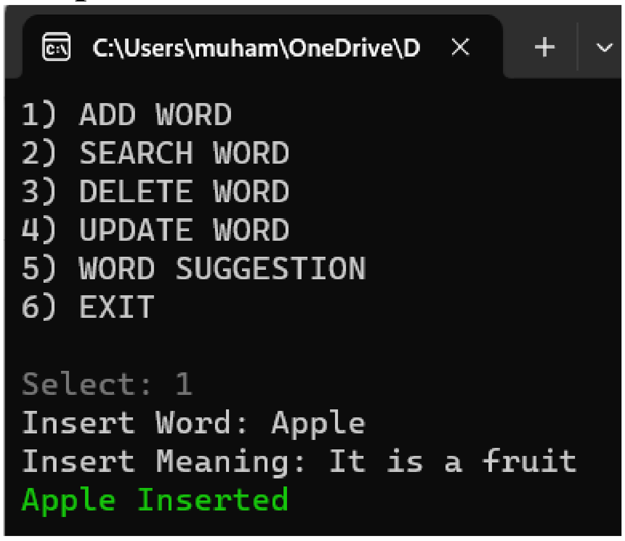
`void suggestWord(Node* curr, string prefix, int& count);`

This function recursively prints 10 words and their meaning with the prefix word passed as parameter.

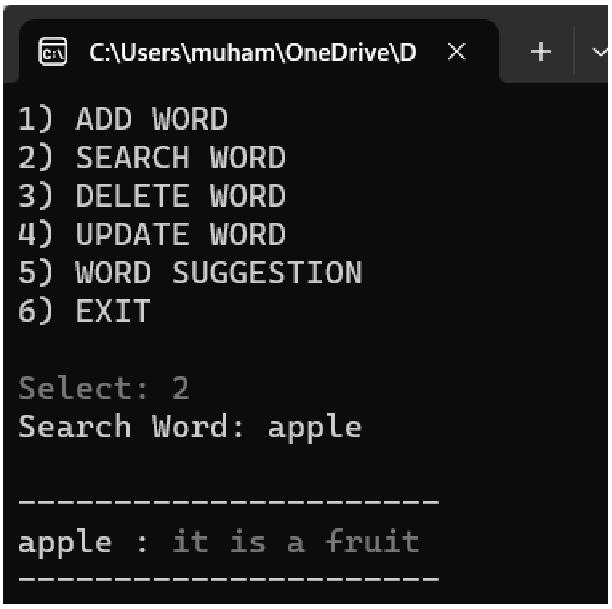
Conclusion

In conclusion, the implementation of a dictionary using a Trie data structure offers several advantages in performing dictionary related operations such as add word, delete word, or word suggestion. The Trie Structure provides faster search operations as compared to other data structures. Trie performs prefix-based searching, which is not easily achievable with other data structures.

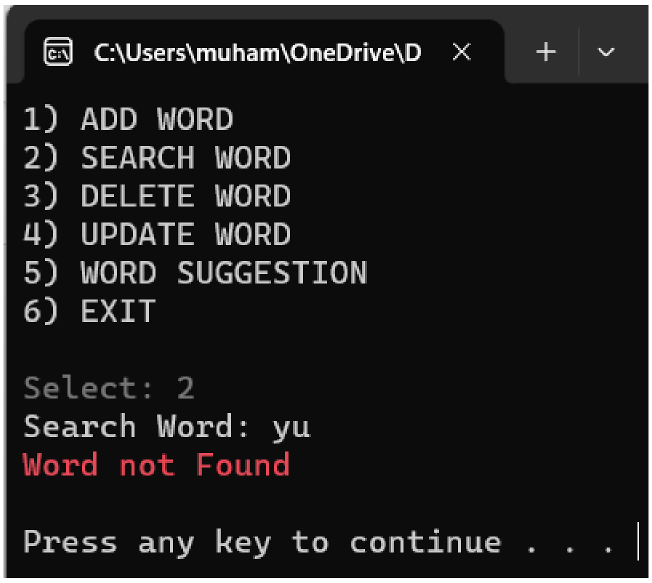
Outputs



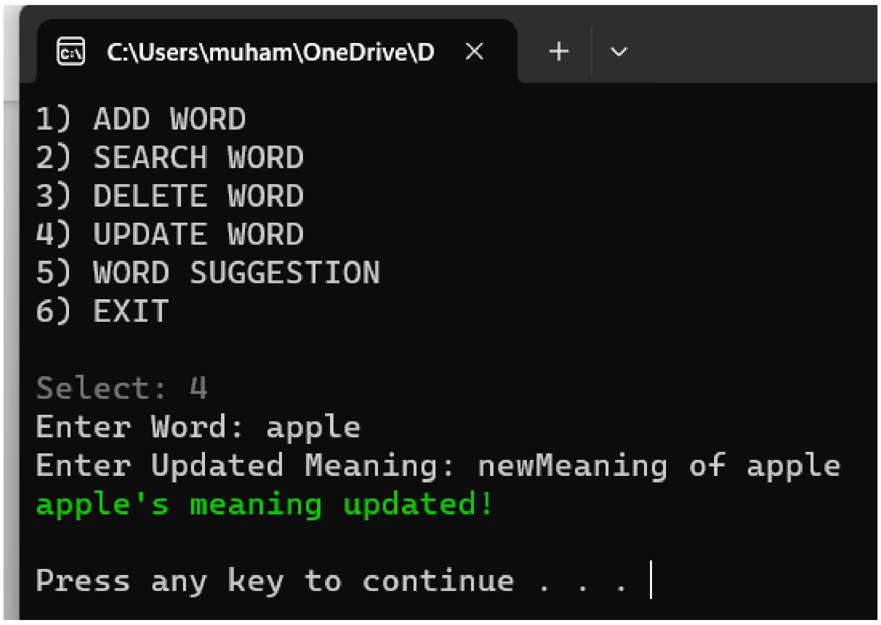
```
1) ADD WORD
2) SEARCH WORD
3) DELETE WORD
4) UPDATE WORD
5) WORD SUGGESTION
6) EXIT

Select: 1
Insert Word: Apple
Insert Meaning: It is a fruit
Apple Inserted
```



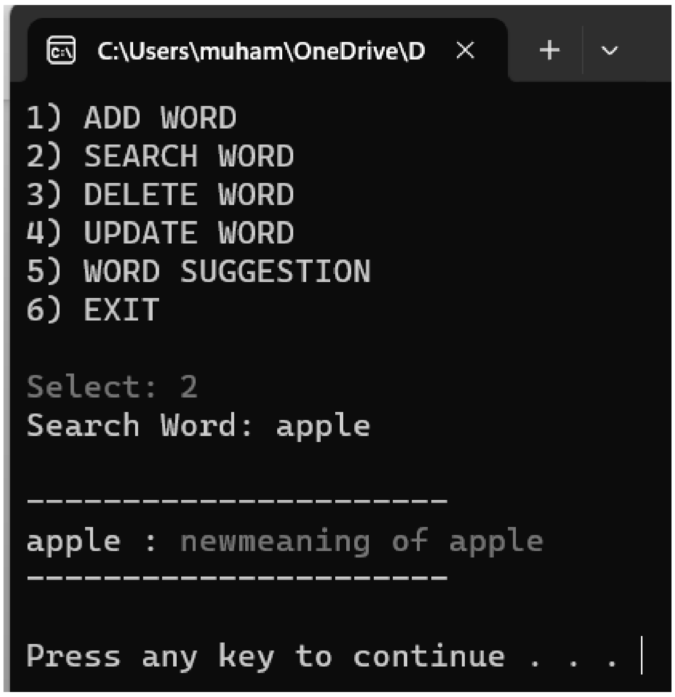
```
1) ADD WORD
2) SEARCH WORD
3) DELETE WORD
4) UPDATE WORD
5) WORD SUGGESTION
6) EXIT

Select: 2
Search Word: apple

----------------------
apple : it is a fruit
----------------------
```



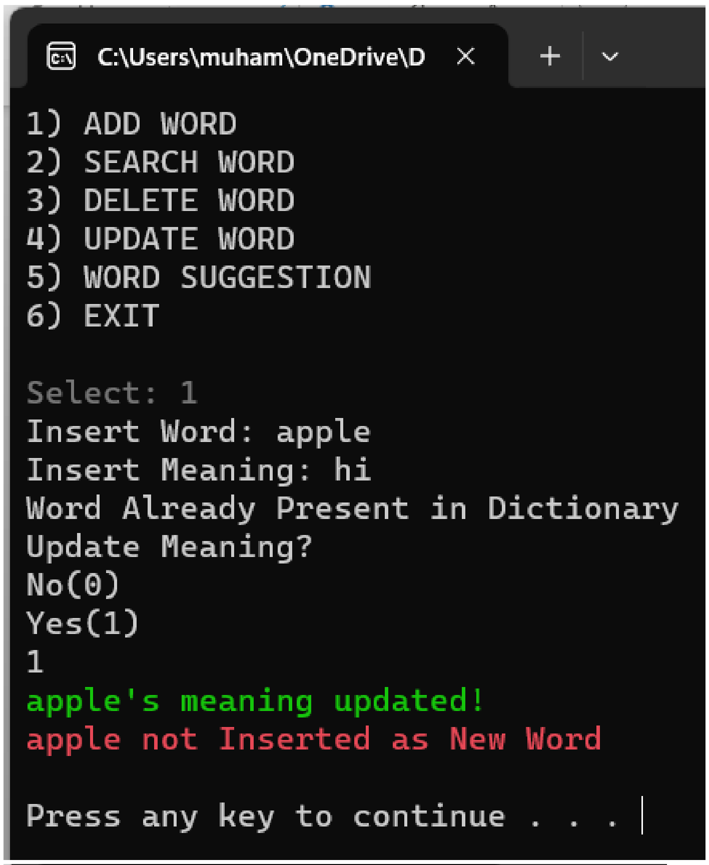
```
1) ADD WORD
2) SEARCH WORD
3) DELETE WORD
4) UPDATE WORD
5) WORD SUGGESTION
6) EXIT

Select: 2
Search Word: yu
Word not Found

Press any key to continue . . .
```



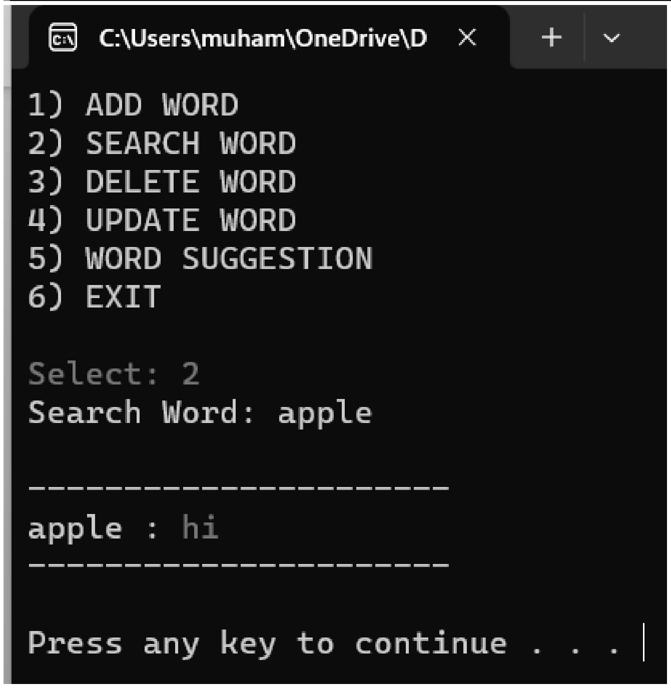
```
1) ADD WORD
2) SEARCH WORD
3) DELETE WORD
4) UPDATE WORD
5) WORD SUGGESTION
6) EXIT

Select: 4
Enter Word: apple
Enter Updated Meaning: newMeaning of apple
apple's meaning updated!

Press any key to continue . . .
```



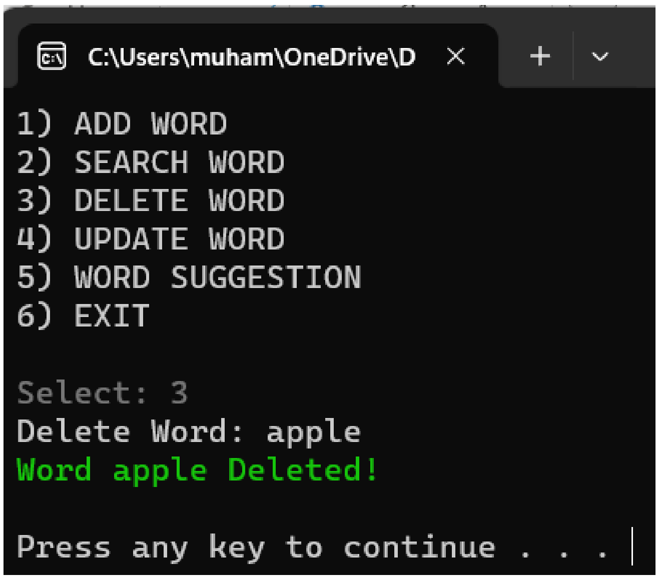
```
1) ADD WORD
2) SEARCH WORD
3) DELETE WORD
4) UPDATE WORD
5) WORD SUGGESTION
6) EXIT

Select: 2
Search Word: apple

----------------------
apple : newmeaning of apple
----------------------

Press any key to continue . . .
```



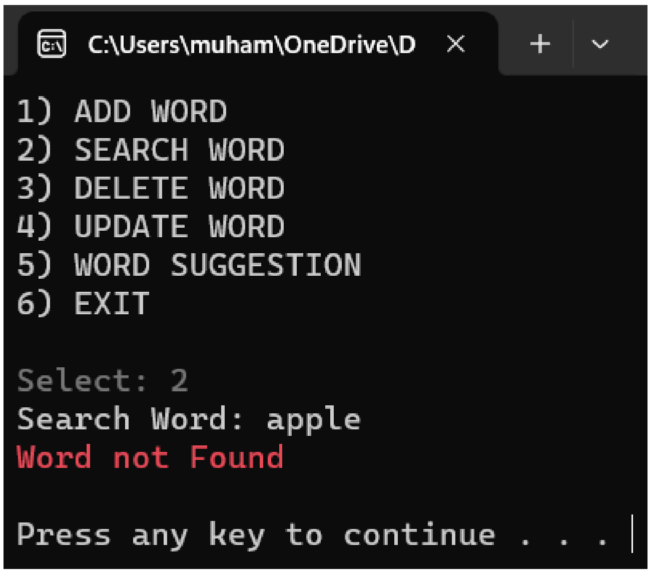
```
1) ADD WORD
2) SEARCH WORD
3) DELETE WORD
4) UPDATE WORD
5) WORD SUGGESTION
6) EXIT

Select: 1
Insert Word: apple
Insert Meaning: hi
Word Already Present in Dictionary
Update Meaning?
No(0)
Yes(1)
1
apple's meaning updated!
apple not Inserted as New Word

Press any key to continue . . .
```



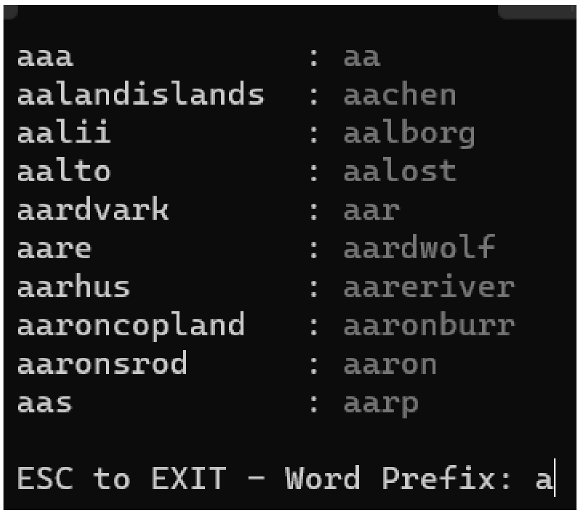
```
1) ADD WORD
2) SEARCH WORD
3) DELETE WORD
4) UPDATE WORD
5) WORD SUGGESTION
6) EXIT

Select: 2
Search Word: apple

----------------------
apple : hi
----------------------

Press any key to continue . . .
```



```
1) ADD WORD
2) SEARCH WORD
3) DELETE WORD
4) UPDATE WORD
5) WORD SUGGESTION
6) EXIT

Select: 3
Delete Word: apple
Word apple Deleted!

Press any key to continue . . .
```

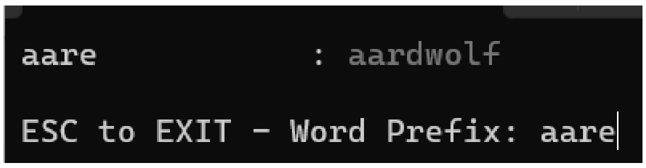


```
1) ADD WORD
2) SEARCH WORD
3) DELETE WORD
4) UPDATE WORD
5) WORD SUGGESTION
6) EXIT

Select: 2
Search Word: apple
Word not Found

Press any key to continue . . .
```



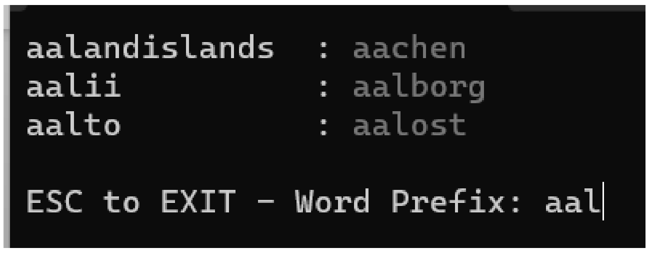
```
aaa             : aa
aalandislands   : aachen
aalii           : aalborg
aalto           : aalost
aardvark        : aar
aare            : aardwolf
aarhus          : aareriver
aaroncopland    : aaronburr
aaronsrod       : aaron
aas             : aarp

ESC to EXIT - Word Prefix: a
```

Figure 1 Word Suggestion



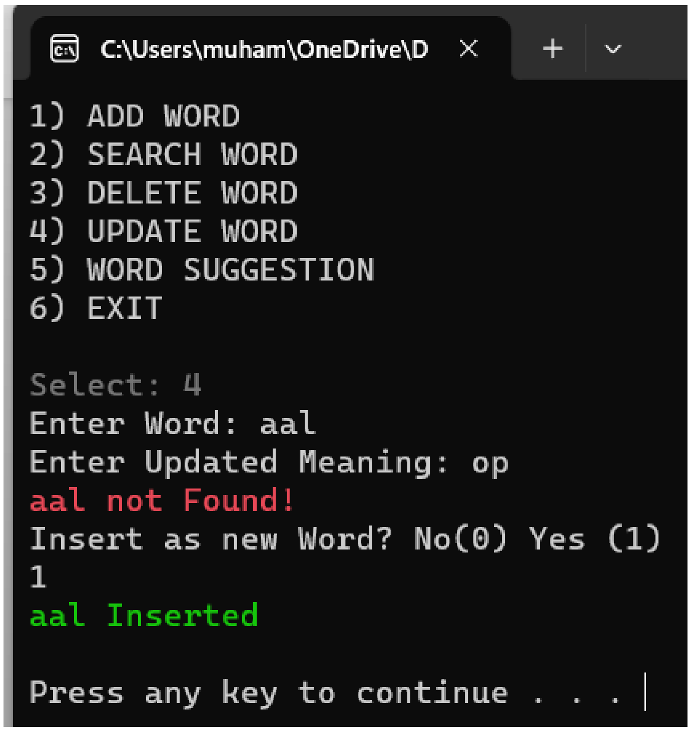
```
aare            : aardwolf

ESC to EXIT - Word Prefix: aare
```

Figure2 Word Suggestion

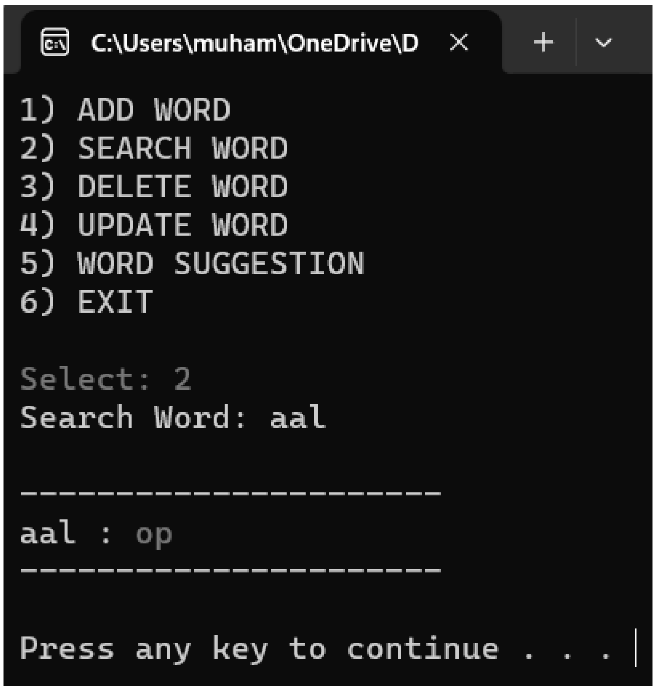


```
aalandislands  : aachen
aalii          : aalborg
aalto          : aalost

ESC to EXIT - Word Prefix: aal
```

Figure 3 Word Suggestion



```
1) ADD WORD
2) SEARCH WORD
3) DELETE WORD
4) UPDATE WORD
5) WORD SUGGESTION
6) EXIT

Select: 4
Enter Word: aal
Enter Updated Meaning: op
aal not Found!
Insert as new Word? No(0) Yes (1)
1
aal Inserted

Press any key to continue . . .
```



```
1) ADD WORD
2) SEARCH WORD
3) DELETE WORD
4) UPDATE WORD
5) WORD SUGGESTION
6) EXIT

Select: 2
Search Word: aal

----------------------
aal : op
----------------------

Press any key to continue . . .
```