Ahmedabad University
School of Engineering and Applied Science

Data Structures and Algorithms - Lab

FINAL PROJECT REPORT

BATCH : 2

GROUP NUMBER : 21

1) Title of project:

Spelling checker &Auto correction of words

Team Members (Name and ID):

Janvi Patel (201501072)
Nishi Patel (201501076)

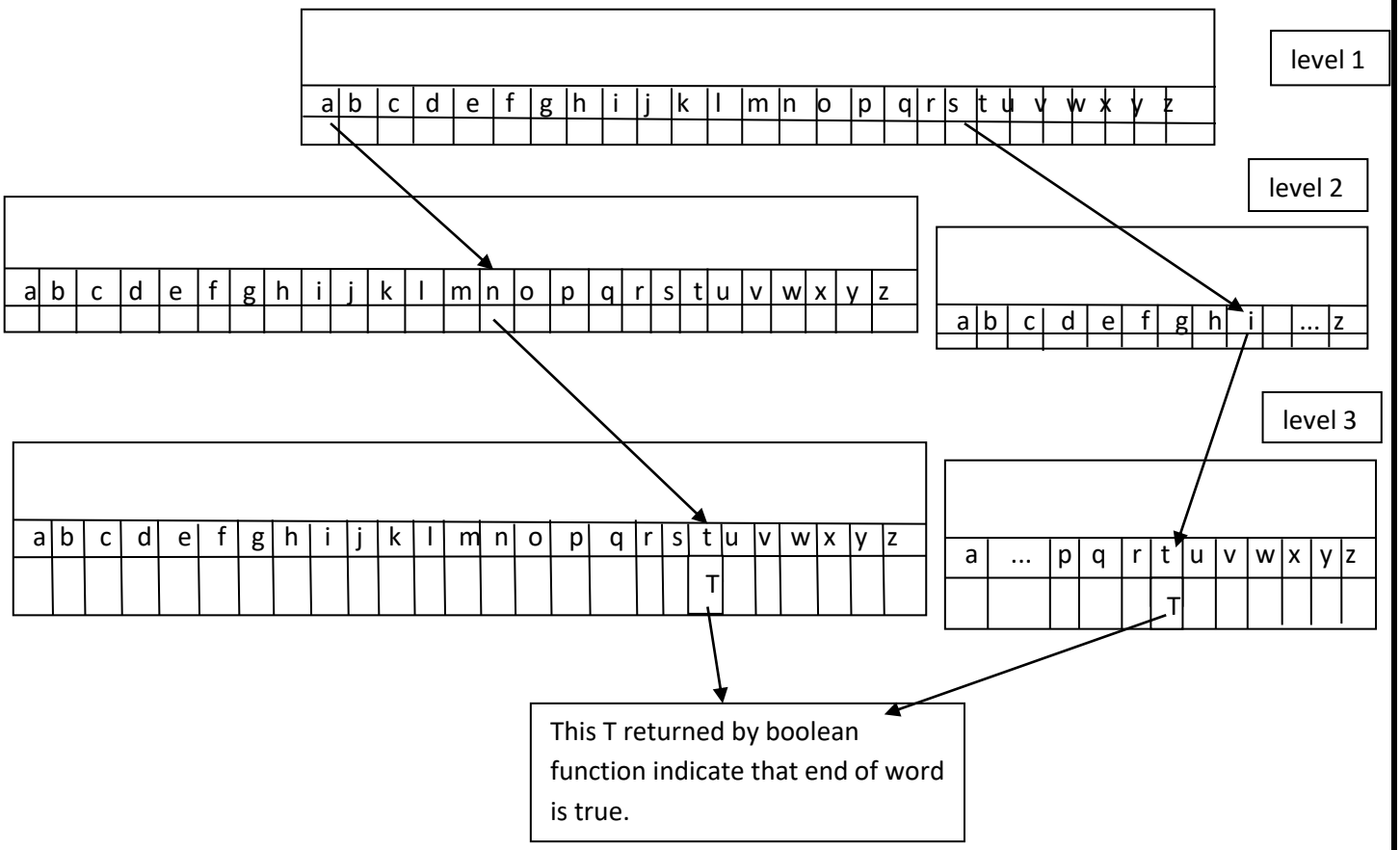
2) Problem Statement:

Some people in their daily life make mistakes in English language spellings. If we think of some big industries, spelling mistakes should not be there. Poor spelling can be embarrassing at some level. In today's fast generation we have lots of work to do in shortage of time. That's why to avoid mistakes of spellings we have to make system that auto correct the word. Correct spelling improves the overall presentation in your work and will help with your confidence in writing spellings. So firstly we have to check whether the word is correct or not and if the word is not correct then we can change that word by correction of word that is called auto correction. Our project is on Spelling checker and Auto correction of word using tree data structure. Given a list of words, firstly the system will make TRIE tree data structure and then ask user to enter a word. Then our application will check whether the spelling of that word is correct or not. If the word is correct then the system will response that spelling is true otherwise if the word is not correct then the system will response that you are doing mistake of spelling and give suggestions of that related spellings. In this way this system will help you to correct your spelling from incorrect spelling. This system will be present in every electronic appliance like mobile and computer. We have 60000 words in our list. If user want to add words in list he/she can add word in list.

3) List of Data Structures used with logical design:

1.	Structure
2.	TRIE tree data structure
3.	Array
4.	Link list

Logic design: spelling ant and sit :



Here we are do one type of thing that if we enter word in TRIE tree and if it is suppose 'transform' , at the level 9 in which r will point to m and at node in which m exists there will be boolean function isleaf 'T'. Now if we want to add spelling like 'transformation' so in our TRIE tree it will traverse through character 'm' and create new nodes to complete the spelling 'transformation'.

4) Operations to be perform on each data structure :

- Insertion :

  Firstly we insert all the words in TRIE tree and one spelling is complete then we make sure that boolean function isleaf will return T. So, it will help us for searching the related words in TRIE tree for Auto correction. We insert spelling in TRIE tree in the way of logic diagram.

- Updation :

  We use updation of the list . We can update list by inserting new words in the list of 60000 words.

- Search for a specific word :

  We used simple traversal to search that word in our list. We also use search for giving suggestion of words.

- Print suggested words :

  In our application if the word is incorrect then we print the words which are related to that word.

- Keeping words in a specific order (ascending order) :

  Our list of words is given in ascending order . But if user insert the word in the list , the list will no more in ascending order.

- Save each data structure in a file and regenerate data structure from a file

  In our project, we used file for list of word. Using list of words we make TRIE tree.

5) Pseudo code and flow charts for each logical process :

Pseudo code :

1) TrieNode ()

(This function create new node of a TRIE tree)

```
struct TrieNode *getNode(void)
struct TrieNode *pNode← NULL
pNode = (struct TrieNode *)malloc(sizeof(struct TrieNode))
if (pNode)
   pNode->isLeaf← 0
   for i←0 to ALPHABET_SIZE
            pNode->children[i] ← NULL
end of if
   return pNode
end of structure
```

2) Insert(root,key)

(This function insert word in TRIE)

```
void insert(struct TrieNode *root, const char *key)
struct TrieNode *curr← root
   for level ← 0 to length
         index ← CHAR_TO_INDEX(key[level])
         if (!curr->children[index])
                  curr->children[index] ←getNode()
         curr←curr->children[index]
   end of for loop
curr->isLeaf← 1
end of insert function
```

3) search(root,key)

(This function search a word in TRIE)

```
bool search(struct TrieNode *root, const char *key)
struct TrieNode *curr← root
   for level ← 0 to length
      index = CHAR_TO_INDEX(key[level])
      if (!curr->children[index])
         return false
      curr←curr->children[index]
   end of for loop
   return (curr != NULL &curr->isLeaf)
 end of search function
```

4) autocorrect_repeat(root,word)

(This function autocorrect word by double all characters of the word and then check word from TRIE tree.

eg. If we enter aple then it will check for aaple,apple,aplle,aplee)

```
void autocorrect_repeat(structTrieNode *root,char word[100])
length_word←strlen(word)
   while cn<length_word
        i←0
        while i<=cn
                new[i]←word[i]
                i++
        end of while loop
        for j ←i-1 to length_word
                new[i]←word[j]
        for i←0 to length_word
                new1[i]←new[i]
        new1[length_word+1] ← 0
        if(search(root, new1)=1)
                printf new1 word
                cn++
   end of while loop
 end of function
```

5) autocorrect_swap(root,word)

(This function autocorrect word by swaping characters of a word and then check it from TRIE tree.

eg. if we enter appel then it will check for apple)

```
void autocorrect_swap(structTrieNode *root,char word[100])
length_word←strlen(word)
for i←0 to length_word
        original[i]←word[i]
for i←0 to length_word
        for j←i+1 to length_word
                temp←word[i]
                word[i]←word[j]
                word[j]←temp
            if (search(root, word)=1)
```

```
                    print suggested words
                for k←0 to length_word
                    word[k]←original[k]
            end of for loop
        end of for loop
    end of function
```

6) autocorrect_letter(root,word)

(This function autocorrect word by putting all characters from a to z inplace of all chatacters of a word.

eg. If we enter realise then it will check for realize)

```
void autocorrect_letter(struct TrieNode *root,char word[100])
length_word←strlen(word)
  while cn<length_word
        j←0
        while j<26
                i←0
                while i<cn
                    new2[i]←word[i]
                    i++
                end of while loop
                c ← j + 'a'
                new2[i]←c
                i++
                 while i<length_word
                    new2[i]←word[i]
                    i++
                end of while loop
                new2[i]←0
                 if(search(root, new2)=1)
                    print new2 word
                end of if
                j++
        end of while loop
```

```
cn++
end of while loop
        end of function
```

7) autocorrect_hiddenletter(root,word)

(This function autocorrect word by putting extra character in a word.
eg. If we enter nock then it will check for knock)

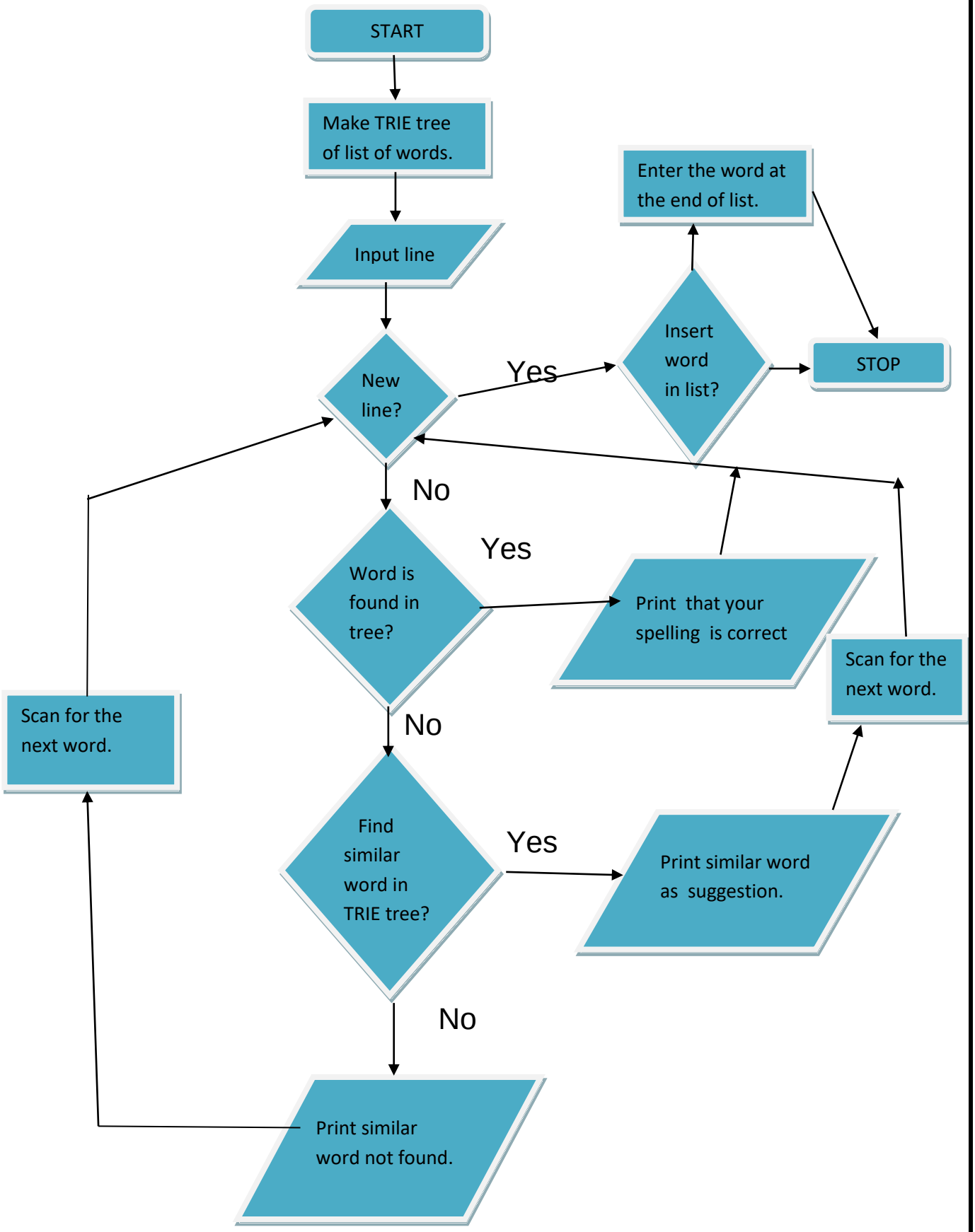
```
void autocorrect_hiddenletter(structTrieNode *root,char word[100])
length_word←strlen(word)
        while cn←length_word
          j←0
          while j < 26
                i←0
              while i<cn
                new3[i]←word[i]
                i++
              end of while loop
              c ← j + 'a'
              new3[i]←c
              i++
              for k←i-1 to length_word
                new3[i]←word[k]
                i++
              end of for loop
              for i←0 to length_word
                 new4[i]←new3[i]
              end of for loop
              new4[length_word+1]←0
              if(search(root,new4)=1)
                print new4 word
              end of if

                j++
```

```
            end of while loop
        cn++
          end of while loop
        end of function
```

8) main function

```
void main()
   struct TrieNode *root←getNode()
fmain←fopen("trie.txt" , "r")
   while fscanf(fmain,"%s",ch) != EOF
      insert(root,ch)
   end of while loop
fclose(fmain)
   scan word in a line which we want to search
while temp_word != NULL
   print the word if it exist in tree
   if(search(root, word)=0)
            for cnt←0 to 3
                  if(cnt=0)
                        autocorrect_swap(root,word)
                  else if(cnt=1)
                        autocorrect_hiddenletter(root,word)
                  else if(cnt=2)
                        autocorrect_repeat(root,word)
                  else
                        autocorrect_letter(root,word)
                  end of else
            end of for loop
   end of if
end of while loop
ask do you want to enter a new word in TRIE tree?
If yes then check if the word is already exist or not
      If word already exist then print word already there
      Else insert word in TRIE tree
end of main function
```

flow chart:


START
Make TRIE tree of list of words.
Input line
New line?
Yes
Insert word in list?
Enter the word at the end of list.
STOP
No
Word is found in tree?
Yes
Print that your spelling is correct
Scan for the next word.
No
Find similar word in TRIE tree?
Yes
Print similar word as suggestion.
Scan for the next word.
No
Print similar word not found.

6) List of program :

- **Filename_of_each_program :**
  1) autocorrect.c
  2) trie.txt

- **Input data**:
  In our code we have already given list of words. In list 60000 words are already in the list . That is the input data for our project. For making the TRIE tree we need list.

- **Output data**:
  The word which is entered by user ,our application will find it in our TRIE tree and if the word is found that means the spelling is correct and it will print that " Your word is correct spelled".

  If the spelling is not correct then it firstly print that "incorrect spelling" and then search for related words in the TRIE tree and print the list of suggestion of words.

- **List of bugs or defects**:

  1) Our application will not work for upper case words.

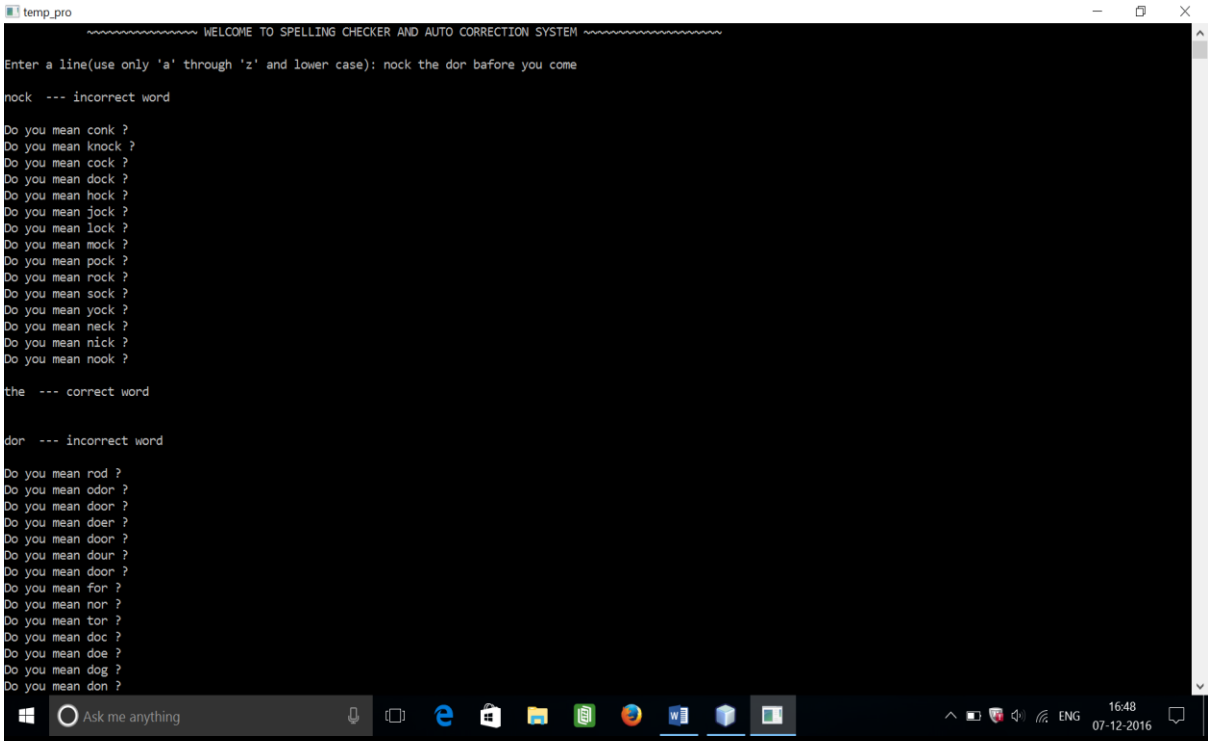
  2) Our application will take time complexity for searching is
     O(L*n) in worst case .
     O(L) in best case.

     Where L = length of word
     n = no. of words

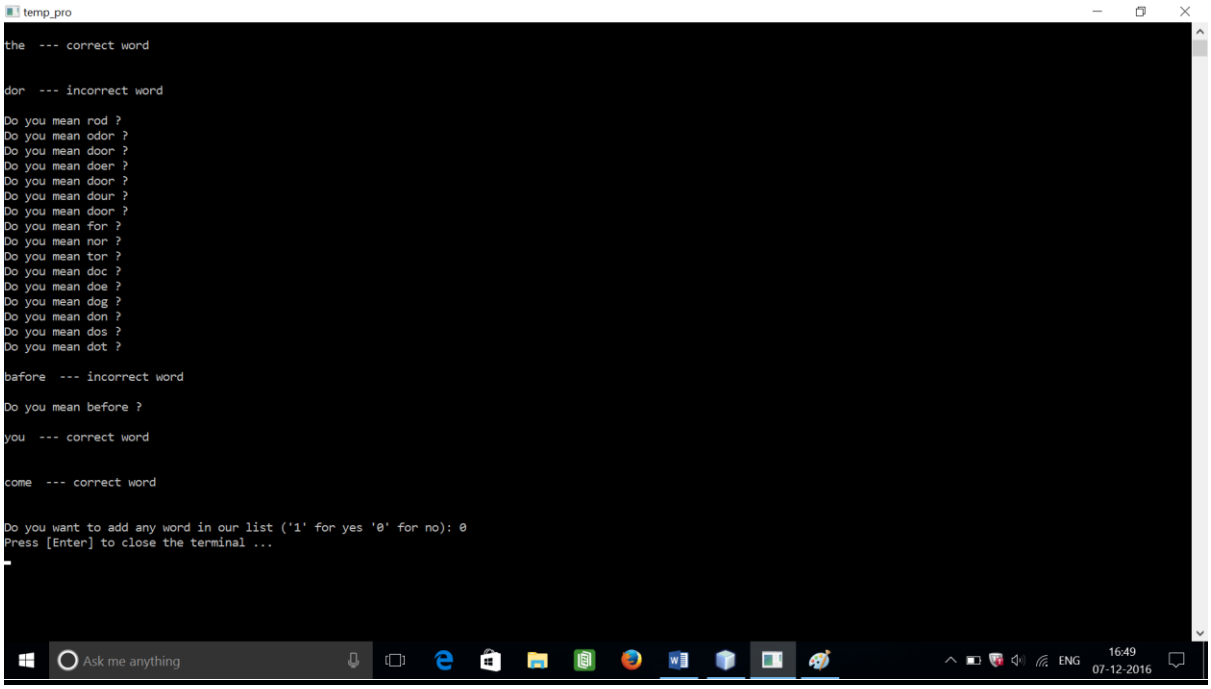
  3) Our application will not generate auto correction while we are writing a spelling it will just suggest the list of spellings after we enter i

7) Test results in the form of snapshot :

user entered : nock the dor bafore you come

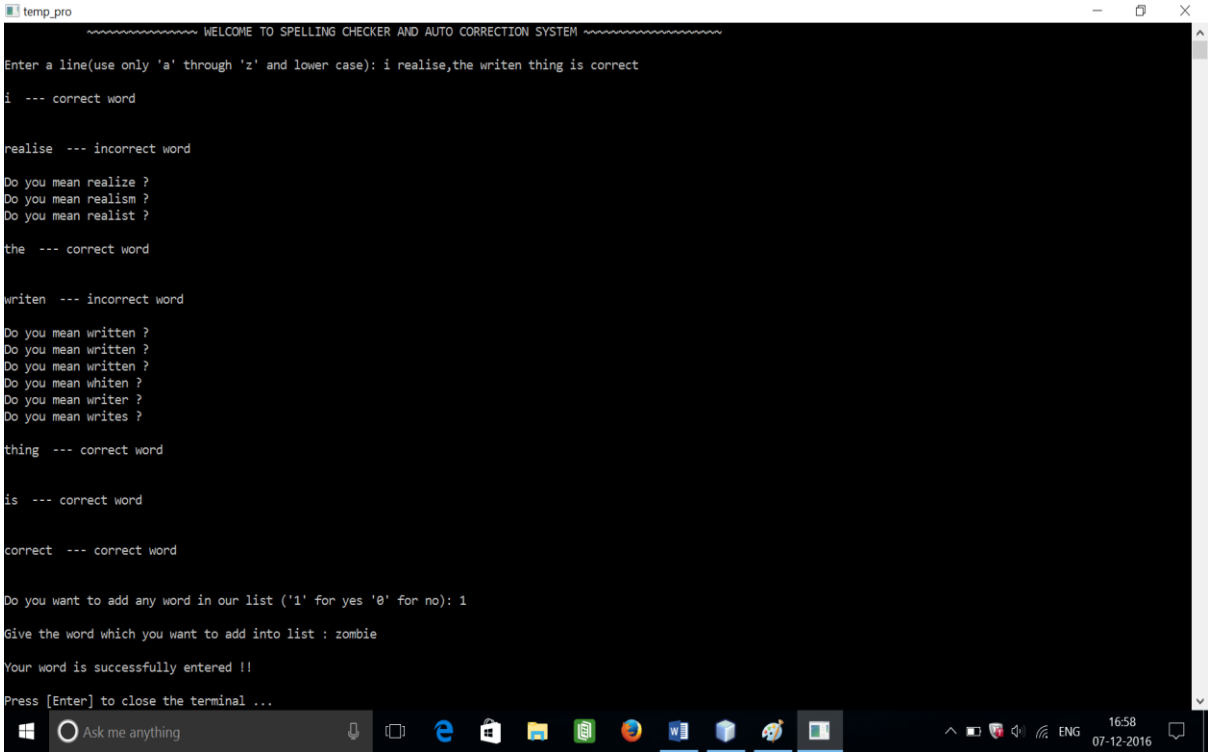


```
temp_pro
~~~~~~~~~~~~~~~~ WELCOME TO SPELLING CHECKER AND AUTO CORRECTION SYSTEM ~~~~~~~~~~~~~~~~~~~~

Enter a line(use only 'a' through 'z' and lower case): nock the dor bafore you come

nock  --- incorrect word

Do you mean conk ?
Do you mean knock ?
Do you mean cock ?
Do you mean dock ?
Do you mean hock ?
Do you mean jock ?
Do you mean lock ?
Do you mean mock ?
Do you mean pock ?
Do you mean rock ?
Do you mean sock ?
Do you mean yock ?
Do you mean neck ?
Do you mean nick ?
Do you mean nook ?

the  --- correct word


dor  --- incorrect word

Do you mean rod ?
Do you mean odor ?
Do you mean door ?
Do you mean doer ?
Do you mean door ?
Do you mean dour ?
Do you mean door ?
Do you mean for ?
Do you mean nor ?
Do you mean tor ?
Do you mean doc ?
Do you mean doe ?
Do you mean dog ?
Do you mean don ?
```



```
temp_pro
the  --- correct word


dor  --- incorrect word

Do you mean rod ?
Do you mean odor ?
Do you mean door ?
Do you mean doer ?
Do you mean door ?
Do you mean dour ?
Do you mean door ?
Do you mean for ?
Do you mean nor ?
Do you mean tor ?
Do you mean doc ?
Do you mean doe ?
Do you mean dog ?
Do you mean don ?
Do you mean dos ?
Do you mean dot ?

bafore  --- incorrect word

Do you mean before ?

you  --- correct word


come  --- correct word


Do you want to add any word in our list ('1' for yes '0' for no): 0
Press [Enter] to close the terminal ...
```

User entered : I realise,the writen thing is correct



8) References :

❖ sites:

http://stackoverflow.com/questions/5431941/why-is-while-feof-file-always-wrong

https://www.youtube.com/watch?v=AXjmTQ8LEoI

http://www.serif.com/appresources/PPX8/Help/en-gb/help/autocorrect.htm

https://www.youtube.com/watch?v=RIUY7ieyH40

https://cswithandroid.withgoogle.com/content/register

http://stackoverflow.com/questions/9629473/c-extracting-words-from-string

http://web.stanford.edu/class/cs97si/suffix-array.pdf