ASK/VIEW DOUBT | SOLUTION | HINT | Your Code | Main Code | Code Pair | C++ (g++ 5)

Problem | Result

Auto complete

Send Feedback

Givan n number of words and an incomplete word w.
You need to auto-complete that word w.

That means, find and print all the possible words which can be formed using the incomplete word w.

Input Format :

Line 1 : Integer n
Line 2 : n words (separated by space)
Line 3 : Word w

Output Format :

All possible words (in different lines)

Note : Order of words doesn't matter.

Sample Input 1 :

7
do dont no not note notes den
no

PREVIOUS | NEXT | CUSTOM INPUT | SUBMIT SOLUTION

```cpp
// #include "TrieNode.h"
#include <string>
#include <vector>
class TrieNode {
    public :
    char data;
    TrieNode **children;
    bool isTerminal;

    TrieNode(char data) {
        this -> data = data;
        children = new TrieNode*[26];
        for(int i = 0; i < 26; i++) {
            children[i] = NULL;
        }
        isTerminal = false;
    }
};

class Trie {
    TrieNode *root;

    public :
    int count;

    Trie() {
        this->count = 0;
        root = new TrieNode('\0');
    }

    bool insertWord(TrieNode *root, string word) {
        // Base case
        if(word.size() == 0) {
            if (!root->isTerminal) {
                root -> isTerminal = true;
                return true;
            } else {
                return false;
            }
        }
        // Small Calculation
        int index = word[0] - 'a';
        TrieNode *child;
        if(root -> children[index] != NULL) {
            child = root -> children[index];
        }
        else {
            child = new TrieNode(word[0]);
            root -> children[index] = child;
        }

        // Recursive call
        return insertWord(child, word.substr(1));
    }

    // For user
    void insertWord(string word) {
        if (insertWord(root, word)) {
            this->count++;
        }
    }

    TrieNode*  print(TrieNode *root, string word){
        if(word.size() == 0) {
            return root;
        }
        int index = word[0] - 'a';
        if(root -> children[index] != NULL) {
            return print(root->children[index],word.substr(1));
        }
        else {
            return NULL;
        }
    }

    void printall(TrieNode *root, string pattern){

//         pattern += root->data;
//         cout<<pattern;









        //string temp = s;
        //cout<<pattern<<" "<<root->data<<" "<<endl;
        pattern += root->data;
        for(int i=0;i<26;i++){
            if(root->children[i] != NULL){
                printall(root->children[i],pattern);
            }
        }
         if(root->isTerminal){
            cout<<pattern<<endl;
         }
    }


    void autoComplete(vector<string> arr, string pattern) {
        // Complete this function
        // Print the output as specified in question


        for(int i=0;i<arr.size();i++){
            insertWord(arr[i]);
        }
         TrieNode *last = print(root,pattern);
        if(last!=NULL){
            string s = pattern.substr(0,pattern.length()-1);
            printall(last,s);
        }
    }

```

```
118
119
120
121
122 };
```

CUSTOM INPUT SUBMIT SOLUTION