

Problem

Result

Code : Search in Tries

[Send Feedback](#)

Implement the function SearchWord for the Trie class.

For a trie, write the function for searching a word.
Return true if found successfully otherwise return false.

Note : main function is given for your reference
which we are using internally to test the code.

```
1 // #include "TrieNode.h"
2 #include <string>
3
4 // *****
5 * Main function -
6
7 int main() {
8     int choice;
9     cin >> choice;
10    Trie t;
11
12    //cout << "asasas";
13    while(choice != -1){
14        string word;
15        bool ans;
16        switch(choice) {
17            case 1 : // insert
18                // getline(cin, word);
19                cin >> word;
20                t.insertWord(word);
21                break;
22            case 2 : // search
23                // getline(cin, word);
24                cin >> word;
25                ans = t.search(word);
26                if (ans) {
27                    cout << "true" << endl;
28                } else {
29                    cout << "false" << endl;
30                }
31                break;
32            default :
33                return 0;
34        }
35        cin >> choice;
36    }
37    return 0;
38 }
```



< PREVIOUS

> NEXT

CUSTOM INPUT

SUBMIT SOLUTION

```
41 class TrieNode {
42 public :
43     char data;
44     TrieNode **children;
45     bool isTerminal;
46
47     TrieNode(char data) {
48         this->data = data;
49         children = new TrieNode*[26];
50         for(int i = 0; i < 26; i++) {
51             children[i] = NULL;
52         }
53         isTerminal = false;
54     }
55 };
56
57 class Trie {
58     TrieNode *root;
59
60 public :
61
62     Trie() {
63         root = new TrieNode('\0');
64     }
65
66     void insertWord(TrieNode *root, string word) {
67         // Base case
68         if(word.size() == 0) {
69             root->isTerminal = true;
70             return;
71         }
72     }
```

CUSTOM INPUT

SUBMIT SOLUTION

```
76     if(root->children[index] != NULL) {
77         child = root->children[index];
78     }
79     else {
80         child = new TrieNode(word[0]);
81         root->children[index] = child;
82     }
83
84     // Recursive call
85     insertWord(child, word.substr(1));
86 }
87
88 // For user
89 void insertWord(string word) {
90     insertWord(root, word);
91 }
92
93 bool searchHelp(TrieNode *root, string word){
94     if(word.size() == 0) {
95         return root->isTerminal;
96     }
97     int index = word[0] - 'a';
98     if(root->children[index] != NULL) {
99         return searchHelp(root->children[index],word.substr(1));
100     }
101     else {
102         return false;
103     }
104 }
105
106
107 bool search(string word) {
108     // Write your code here
109     return searchHelp(root,word);
110 }
111
112
113
114
115
116
117
```

