

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  #define ll long long
4  #define inf (int)1e9
5
6  TrieTree:
7  struct TrieTree{
8      struct Node{
9          char data; // change type of data
10         int wordCount;
11         int prefixCount;
12         map<char , Node*> edge;
13
14         Node(char a){
15             this->data = a;
16             this->wordCount = 0;
17             this->prefixCount = 0;
18         }
19     };
20
21     Node* base;
22
23     TrieTree(vector<string> a){
24         this->base = new Node('*');
25         for (int i = 0; i < a.size(); i++)
26             addWord(a[i]);
27     }
28
29     void addWord(string s){
30         Node* cur = base;
31         for (int i = 0; i < s.size(); i++)
32         {
33             if(cur->edge[s[i]]==NULL) cur->edge[s[i]] = new Node(s[i]);
34             cur = cur->edge[s[i]];
35             cur->prefixCount++;
36         }
37         cur->wordCount++;
38     }
39
40     // delete word count only not prefix count
41     void deleteWord(string s){
42         Node* cur = base;
43         for (int i = 0; i < s.size(); i++){
44             if(cur->edge[s[i]]==NULL) return;
45             cur = cur->edge[s[i]];
46         }
47         if(cur->wordCount!=0)
48             cur->wordCount--;
49     }
50
51     // return count of word and prefix equal s
52     // word prefix
53     pair<int , int> countWord(string s){
54         Node* cur = base;
55         for (int i = 0; i < s.size(); i++)
56         {
57             if(cur->edge[s[i]]==NULL) return {0 , 0};
58             cur = cur->edge[s[i]];
59         }
60         return make_pair(cur->wordCount , cur->prefixCount);
61     }

```