

```
1 class Trie {
2 private:
3     struct node {
4         char val; map<char, node*> childs_mp; ll deg; node *parent; int cnt;
5     };
6     bool erase_leaf(node *ptr) {
7         if (ptr->val != '\0') {
8             do {
9                 char v = ptr->val;
10                ptr->cnt--;
11                ptr = ptr->parent;
12                if (ptr->childs_mp[v]->cnt == 0) {
13                    delete(ptr->childs_mp[v]);
14                    ptr->childs_mp.erase(v);
15                }
16            } while (ptr != root);
17            ptr->cnt--;
18            return true;
19        }
20        else return false;
21    }
22    node *root;
23    bool multi_flag;
24 public:
25     Trie(bool multi_flag) {
26         root = new node{ '\0', {}, 0, nullptr, 0 };
27         Trie::multi_flag = multi_flag;
28     }
29     void add(string s) {
30         node *a = root;
31         Loop(i, s.length()) {
32             char c = s[i];
33             if (a->childs_mp.find(c) == a->childs_mp.end()) {
34                 node *node_buf = new node{ c, {}, a->deg + 1, a, 0 };
35                 a->childs_mp[c] = node_buf;
36             }
37             a->cnt++;
38             a = a->childs_mp[c];
39         }
40         if (a->childs_mp.find('\0') == a->childs_mp.end()) {
41             node *nil = new node{ '\0', {}, a->deg + 1, a, 0 };
42             a->childs_mp['\0'] = nil;
43         }
44         a->cnt++;
45         a = a->childs_mp['\0'];
46         a->cnt++;
47         if (!multi_flag && a->cnt >= 2) erase_leaf(a);
48     }
49     bool find(string s) {
50         node *a = root;
51         Loop(i, s.length()) {
52             char c = s[i];
53             if (a->childs_mp.find(c) == a->childs_mp.end()) return false;
54             else a = a->childs_mp[c];
55         }
56         if (a->childs_mp.find('\0') != a->childs_mp.end()) return true;
57         else return false;
58     }
59     bool erase(string s) {
60         node *a = root;
61         Loop(i, s.length()) {
62             char c = s[i];
63             if (a->childs_mp.find(c) == a->childs_mp.end()) return false;
64             else a = a->childs_mp[c];
65         }
66         if (a->childs_mp.find('\0') != a->childs_mp.end()) {
67             if (erase_leaf(a)) return true;
68             else return false;
69         }
70         else return false;
71     }
```

72 };