```
3
4  class Node{
5      char data;
6      unordered_map<char,Node*> m;
7      bool isTerminal;
8
9  public:
10
11     Node(char data){
12         data = d;
13         isTerminal = false;
14     }
15 };
16
17
18 class Trie{
19     Node*root;
20
21 public:
22     Trie(){
23         root = new Node('\0');
24     }
25     //Insertion
26     void insert(string word){
27         Node* temp = root;
```

```
        Node* temp = root;

        for(char ch : word){
            if(temp->m.count(ch)==0){
                Node*n = new Node(ch);
                temp->m[ch] = n;
            }
            temp = temp->m[ch];
        }
        temp->isTerminal = true;
    }
    //Searching

    bool search(string word){

        Node*temp = root;
        for(char ch:word){
            if(temp->m.count(ch)==0){
                return false;
            }
            temp = temp->m[ch];
        }
        return temp->isTerminal;
    }

};
```