基本的字典树，字典树又称单词查找树，Trie树，是一种树形结构，是一种哈希树的变种。典型应用是用于统计，排序和保存大量的字符串（但不仅限于字符串），所以经常被**[搜索引擎](http://lib.csdn.net/base/searchengine" \o "搜索引擎知识库" \t "http://blog.csdn.net/cambridgeacm/article/details/_blank)**系统用于文本词频统计。它的优点是：利用字符串的公共前缀来节约存储空间，最大限度地减少无谓的字符串比较，查询效率比哈希表高。

# 1250统计难题

**Problem Description**

Ignatius最近遇到一个难题,老师交给他很多单词(只有小写字母组成,不会有重复的单词出现),现在老师要他统计出以某个字符串为前缀的单词数量(单词本身也是自己的前缀).

**Input**

输入数据的第一部分是一张单词表,每行一个单词,单词的长度不超过10,它们代表的是老师交给Ignatius统计的单词,一个空行代表单词表的结束.第二部分是一连串的提问,每行一个提问,每个提问都是一个字符串.  
注意:本题只有一组测试数据,处理到文件结束.

**Output**

对于每个提问,给出以该字符串为前缀的单词的数量.

**Sample Input**

banana

band

bee

absolute

acm

ba

b

band

abc

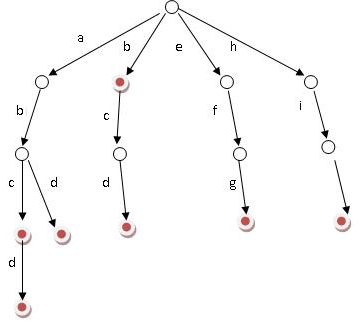
**Sample Output**

2

3

1

0



//字典树

#include<iostream>

#include<cstdio>

#include<cstring>

#include<cstdlib>

using namespace std;

struct node

{

    int count;

    node \*childs[26];

    node()

    {

        count=0;

        int i;

        for(i=0;i<26;i++)

        childs[i]=NULL;

    }

};

node \*root=new node;

node \*current,\*newnode;

void insert(char \*str)

{

    int i,m;

    current=root;

    for(i=0;i<strlen(str);i++)

    {

        m=str[i]-'a';

        if(current->childs[m]!=NULL)

        {

            current=current->childs[m];

            ++(current->count);

        }

        else

        {

            newnode=new node;

            ++(newnode->count);

            current->childs[m]=newnode;

            current=newnode;

        }

    }

}

int search(char \*str)

{

    int i,m;

    current=root;

    for(i=0;i<strlen(str);i++)

    {

        m=str[i]-'a';

        if(current->childs[m]==NULL)

        return 0;

        current=current->childs[m];

    }

    return current->count;

}

int main()

{

    char str[20];

    while(gets(str),strcmp(str,""))

    insert(str);

    while(gets(str)!=NULL)

    printf("%d\n",search(str));

    return 0;

}