#include <stdio.h>

#include <malloc.h>

struct node

{

char \*data[10000];

int length;

};

int main()

{

struct node \*L;

int n=-1,i,j,t,k,flag,q,num;

int ttt[1000];

char a[20];

L=(struct node\*)malloc(sizeof(struct node));

for(n=0;n<10000;n++)

{

L->data[n]=(char\*)malloc(16\*sizeof(char));

scanf("%s",L->data[n]);

if(L->data[n][0]=='#') break;

}

scanf("%s",a);

while(a[0]!='#')

{

num=0;

for(k=0;k<n;k++)

{

flag=0;

for(i=0;a[i]!='\0';i++);

for(j=0;L->data[k][j]!='\0';j++);

t=i-j;

if(t!=1&&t!=-1&&t!=0) flag=0;

else if(t==0)

{

for(t=0;t<i;t++)

{

if(a[t]!=L->data[k][t])

{

flag++;

if(flag>1) break;

}

}

if(t==i&&flag==0) flag=3;

if(flag==1) {ttt[num]=k;num++;}

}

else if(t==1)

{

for(t=0,q=0;t<i;)

{

if(a[t]!=L->data[k][q])

{

t++;

flag++;

if(flag>1) break;

}

else {t++;q++;}

}

if(flag==1) {ttt[num]=k;num++;}

}

else

{

for(t=0,q=0;q<j;)

{

if(a[t]!=L->data[k][q])

{

q++;

flag++;

if(flag>1) break;

}

else {t++;q++;}

}

if(flag==1) {ttt[num]=k;num++;}

}

if(flag==3)

{printf("%s is correct\n",a);break;}

}

if(flag==3) {scanf("%s",a);continue;}

else

{

printf("%s:",a);

for(i=0;i<num;i++)

printf(" %s",L->data[ttt[i]]);

}

printf("\n");

scanf("%s",a);

}

return 0;

}

**问题 I: 单词检查(Ⅰ)- 顺序表实现**

时间限制: 1 Sec 内存限制: 128 MB  
提交: 2308 解决: 768  
[[提交](http://acm.hnust.cn/JudgeOnline/submitpage.php?cid=1569&pid=8&langmask=0)][[状态](http://acm.hnust.cn/JudgeOnline/problemstatus.php?id=1825)][[讨论版](http://acm.hnust.cn/JudgeOnline/bbs.php?pid=1825&cid=1569)]

**题目描述**

许多应用程序，如字处理软件，邮件客户端等，都包含了单词检查特性。单词检查是根据字典，找出输入文本中拼错的单词，我们认为凡是不出现在字典中的单词都是错误单词。不仅如此，一些检查程序还能给出类似拼错单词的修改建议单词。 例如字典由下面几个单词组成：  
bake cake main rain vase  
如果输入文件中有词vake ，检查程序就能发现其是一个错误的单词，并且给出 bake, cake或vase做为修改建议单词。  
修改建议单词可以采用如下生成技术：  
(1)在每一个可能位置插入‘a-'z'中的一者  
(2)删除单词中的一个字符  
(3)用‘a'-'z'中的一者取代单词中的任一字符  
很明显拼写检查程序的核心操作是在字典中查找某个单词，如果字典很大，性能无疑是非常关键的。  
你写的程序要求读入字典文件，然后对一个输入文件的单词进行检查，列出其中的错误单词并给出修改建议。  
  
课程设计必须采用如下技术完成并进行复杂度分析及性能比较。  
(1)朴素的算法,用线性表维护字典  
(2)使用二叉排序树维护字典  
(3)采用hash技术维护字典  
  
本题要求使用顺序表实现。

**输入**

输入分为两部分。  
第一部分是字典，每个单词占据一行，最后以仅包含'#'的一行表示结束。所有的单词都是不同的，字典中最多10000个单词。  
  
输入的第二部分包含了所有待检测的单词，单词数目不超过50。每个单词占据一行，最后以仅包含'#'的一行表示结束。  
  
字典中的单词和待检测的单词均由小写字母组成，并且单词最大长度为15。

**输出**

按照检查次序每个单词输出一行，该行首先输出单词自身。如果单词在字典中出现，接着输出" is correct"。如果单词是错误的，那么接着输出':'，如果字典中有建议修改单词，则按照字典中出现的先后次序输出所有的建议修改单词（每个前面都添加一个空格），如果无建议修改单词，在':'后直接换行。

**样例输入**

i

is

has

have

be

my

more

contest

me

too

if

award

#

me

aware

m

contest

hav

oo

or

i

fi

mre

#

**样例输出**

me is correct

aware: award

m: i my me

contest is correct

hav: has have

oo: too

or:

i is correct

fi: i

mre: more me