**Corporate Identity**

**Time Limit: 9000/3000 MS (Java/Others)    Memory Limit: 65536/32768 K (Java/Others)  
Total Submission(s): 929    Accepted Submission(s): 366**

Problem Description

Beside other services, ACM helps companies to clearly state their “corporate identity”, which includes company logo but also other signs, like trademarks. One of such companies is Internet Building Masters (IBM), which has recently asked ACM for a help with their new identity. IBM do not want to change their existing logos and trademarks completely, because their customers are used to the old ones. Therefore, ACM will only change existing trademarks instead of creating new ones.  
  
After several other proposals, it was decided to take all existing trademarks and find the longest common sequence of letters that is contained in all of them. This sequence will be graphically emphasized to form a new logo. Then, the old trademarks may still be used while showing the new identity.  
  
Your task is to find such a sequence.

Input

The input contains several tasks. Each task begins with a line containing a positive integer N, the number of trademarks (2 ≤ N ≤ 4000). The number is followed by N lines, each containing one trademark. Trademarks will be composed only from lowercase letters, the length of each trademark will be at least 1 and at most 200 characters.  
  
After the last trademark, the next task begins. The last task is followed by a line containing zero.

Output

For each task, output a single line containing the longest string contained as a substring in all trademarks. If there are several strings of the same length, print the one that is lexicographically smallest. If there is no such non-empty string, output the words “IDENTITY LOST” instead.

Sample Input

3

aabbaabb

abbababb

bbbbbabb

2

xyz

abc

0

Sample Output

abb

IDENTITY LOST

给你n个字符串 让你找他们的公共子串 如果有长度相同的子串 则输出字典树最小的（当初WA了两次 就是因为忽略的这个条件）

AC代码：

#include <stdio.h>

#include <string.h>

#include <algorithm>

**using namespace** std**;**

**int main()**

**{**

**int** t**,**n**,**i**,**j**,**k**,**MIN**,**f**,**len**,**MAX**;**

**char** str**[**1005**][**1005**],**s1**[**1005**],**s2**[**1005**];**

///t组测试数据

**while(~**scanf**(**"%d"**,&**n**))**

**{**

///n个字符串

**if(**n**==**0**)**

**break;**

MIN **=** 1000**;**

**for(**i **=** 0**;** i**<**n**;** i**++)**

**{**

scanf**(**"%s"**,**str**[**i**]);**

len **=** strlen**(**str**[**i**]);**

**if(**MIN**>**len**)**///找到最小串

**{**

MIN **=** len**;**

f **=** i**;**

**}**

**}**

len **=** strlen**(**str**[**f**]);** ///最小串的长度

**int** flag **=** 1**;**

MAX **=** 0**;**

**for(**i **=** 0**;**i **<** len**;** i**++)**///作为标本串子串的头

**{**

**for(**j **=** i**;**j **<** len**;** j**++)**///子串的尾

**{**

**for(**k **=** i**;**k **<=** j**;** k**++)**///复制为两个串，顺序串s1，逆序串s2

**{**

s1**[**k**-**i**] =** str**[**f**][**k**];** ///s1正序 s2倒序

**}**

s1**[**j**-**i**+**1**] =** '\0'**;** ///'\0'之前是s1串和s2串的实际长度

**int** l **=** strlen**(**s1**);**

**for(**k **=** 0**;**k **<** n**;** k**++)**///枚举所有串

**{**

///strstr(s,s1) 在s串中找s1串出现的位置 返回值 bool型 找到 true 否则 fal

**if(!**strstr**(**str**[**k**],**s1**))**

**{**

flag **=** 0**;**

**break;**

**}**

**}**

**if(**l**>**MAX **&&** flag**)**

**{**

MAX **=** l**;**

strcpy**(**s2**,**s1**);**

**}**

**else if(**MAX**==**l**&&**flag**&&**strcmp**(**s2**,**s1**)>**0**)**

**{**

strcpy**(**s2**,**s1**);**

**}**

flag **=** 1**;**

**}**

**}**

**if(**MAX**!=**0**)**

printf**(**"%s\n"**,**s2**);**

**else**

printf**(**"IDENTITY LOST\n"**);**

**}**

**return** 0**;**

**}**