From Skiena and Revilla, Programming Challenges: *The Programming Contest Training Manual*
Springer-Verlag, New York, 2003. ISBN: 0-387-00163-8

110302 Where's Waldorf?

Given an m by n grid of letters and a list of words, find the location in the grid at which the word can be found.

A word matches a straight, uninterrupted line of letters in the grid. A word can match the letters in the grid regardless of case (i.e., upper- and lowercase letters are to be treated as the same). The matching can be done in any of the eight horizontal, vertical, or diagonal directions through the grid.

Input

The input begins with a single positive integer on a line by itself indicating the number of cases, followed by a blank line. There is also a blank line between each two consecutive cases.

Each case begins with a pair of integers m followed by n on a single line, where $1 \leq m, n \leq 50$ in decimal notation. The next m lines contain n letters each, representing the grid of letters where the words must be found. The letters in the grid may be in upper- or lowercase. Following the grid of letters, another integer k appears on a line by itself ($1 \leq k \leq 20$). The next k lines of input contain the list of words to search for, one word per line. These words may contain upper- and lowercase letters only – no spaces, hyphens, or other non-alphabetic characters.

Output

For each word in each test case, output a pair of integers representing its location in the corresponding grid. The integers must be separated by a single space. The first integer is the line in the grid where the first letter of the given word can be found (1 represents the topmost line in the grid, and m represents the bottommost line). The second integer is the column in the grid where the first letter of the given word can be found (1 represents the leftmost column in the grid, and n represents the rightmost column in the grid). If a word can be found more than once in the grid, then output the location of the uppermost occurrence of the word (i.e., the occurrence which places the first letter of the word closest to the top of the grid). If two or more words are uppermost, output the leftmost of these occurrences. All words can be found at least once in the grid.

The output of two consecutive cases must be separated by a blank line.

Sample Input

```
1

8 11
abcDEFGhigg
hEbkWalDork
FtyAwaldORm
FtsimrLqsrc
byoArBeDeyv
Klcbqwikomk
strEBGadhrb
yUiqlxcnBjf
4
Waldorf
Bambi
Betty
Dagbert
```

Sample Output

```
2 5
2 3
1 2
7 8
```

110303 Common Permutation

Given two strings a and b, print the longest string x of letters such that there is a permutation of x that is a subsequence of a and there is a permutation of x that is a subsequence of b.

Input

The input file contains several cases, each case consisting of two consecutive lines. This means that lines 1 and 2 are a test case, lines 3 and 4 are another test case, and so on. Each line contains one string of lowercase characters, with first line of a pair denoting a and the second denoting b. Each string consists of at most 1,000 characters.

Output

For each set of input, output a line containing x. If several x satisfy the criteria above, choose the first one in alphabetical order.

Sample Input

```
pretty
women
walking
down
the
street
```

Sample Output

```
e
nw
et
```

110306 File Fragmentation

Your friend, a biochemistry major, tripped while carrying a tray of computer files through the lab. All of the files fell to the ground and broke. Your friend picked up all the file fragments and called you to ask for help putting them back together again.

Fortunately, all of the files on the tray were identical, all of them broke into exactly two fragments, and all of the file fragments were found. Unfortunately, the files didn't all break in the same place, and the fragments were completely mixed up by their fall to the floor.

The original binary fragments have been translated into strings of ASCII `1`'s and `0`'s. Your job is to write a program that determines the bit pattern the files contained.

Input

The input begins with a single positive integer on its own line indicating the number of test cases, followed by a blank line. There will also be a blank line between each two consecutive cases.

Each case will consist of a sequence of "file fragments," one per line, terminated by the end-of-file marker or a blank line. Each fragment consists of a string of ASCII `1`'s and `0`'s.

Output

For each test case, the output is a single line of ASCII `1`'s and `0`'s giving the bit pattern of the original files. If there are $2N$ fragments in the input, it should be possible to concatenate these fragments together in pairs to make N copies of the output string. If there is no unique solution, any of the possible solutions may be output.

Your friend is certain that there were no more than 144 files on the tray, and that the files were all less than 256 bytes in size.

The output from two consecutive test cases will be separated by a blank line.

Sample Input

```
1

011
0111
01110
111
0111
10111
```

Sample Output

```
01110111
```