“If I had an hour to solve a problem I'd spend 55 minutes thinking about the problem and 5 minutes thinking about solutions.” - Albert Einstein

NSU INDIVIDUAL CONTEST #4

“It's so much easier to suggest solutions when you don't know too much about the problem.” - Malcolm S. Forbes

Problem A. Odd Man Out

Problem

You are hosting a party with G guests and notice that there is an odd number of guests! When planning the party you deliberately invited only couples and gave each couple a unique number C on their invitation. You would like to single out whoever came alone by asking all of the guests for their invitation numbers.

Input

The first line of input gives the number of cases, **N**. **N** test cases follow. For each test case there will be:

- One line containing the value **G** the number of guests.
- One line containing a space-separated list of **G** integers. Each integer **C** indicates the invitation code of a guest.

Output

For each test case, output one line containing "Case **#x**: " followed by the number C of the guest who is alone.

Limits

1 ≤ N ≤ 50
0 < C ≤ 2147483647
3 ≤ G < 1000

Sample

Input	**Output**
3 3 1 2147483647 2147483647 5 3 4 7 4 3 5 2 10 2 10 5	Case #1: 1 Case #2: 7 Case #3: 5

He who asks a question may be a fool for five minutes, but he who never asks a question remains a fool forever. - **Tom Connelly**

Problem C. Minimum Scalar Product

Problem

You are given two vectors $v_1=(x_1,x_2,...,x_n)$ and $v_2=(y_1,y_2,...,y_n)$. The scalar product of these vectors is a single number, calculated as $x_1y_1+x_2y_2+...+x_ny_n$.

Suppose you are allowed to permute the coordinates of each vector as you wish. Choose two permutations such that the scalar product of your two new vectors is the smallest possible, and output that minimum scalar
product.

Input

The first line of the input file contains integer number **T** - the number of test cases. For each test case, the first line contains integer number **n**. The next two lines contain **n** integers each, giving the coordinates of v_1 and v_2 respectively.

Output

For each test case, output a line
Case **#X**: **Y**
where **X** is the test case number, starting from 1, and **Y** is the minimum scalar product of all permutations of the two given vectors.

Limits

T = 1000
$1 \le \mathbf{n} \le 8$
$-1000 \le \mathbf{x}_i, \mathbf{y}_i \le 1000$

Sample

Input	Output
2 3 1 3 -5 -2 4 1 5 1 2 3 4 5 1 0 1 0 1	Case #1: -25 Case #2: 6

No problem can stand the assault of sustained thinking. - **Voltaire**

Problem D. All Your Base

Problem

In A.D. 2100, aliens came to Earth. They wrote a message in a cryptic language, and next to it they wrote a series of symbols. We've come to the conclusion that the symbols indicate a number: the number of seconds before war begins!

Unfortunately we have no idea what each symbol means. We've decided that each symbol indicates one digit, but we aren't sure what each digit means or what base the aliens are using. For example, if they wrote "ab2ac999", they could have meant "31536000" in base 10 -- exactly one year -- or they could have meant "12314555" in base 6 -- 398951 seconds, or about four and a half days. We are sure of three things: the number is positive; like us, the aliens will never start a number with a zero; and they aren't using unary (base 1). Your job is to determine the minimum possible number of seconds before war begins.

Input

The first line of input contains a single integer, **T**. **T** test cases follow. Each test case is a string on a line by itself. The line will contain only characters in the 'a' to 'z' and '0' to '9' ranges (with no spaces and no punctuation), representing the message the aliens left us. The test cases are independent, and can be in different bases with the symbols meaning different things.

Output

For each test case, output a line in the following format:

Case #**X**: **V**

Where **X** is the case number (starting from 1) and **V** is the minimum number of seconds before war begins.

Limits

$1 \leq \mathbf{T} \leq 100$

The answer will never exceed 10^{18}

$1 \leq$ the length of each line < 61

Sample

Input	Output
3 11001001 cats zig	Case #1: 201 Case #2: 75 Case #3: 11

How you think about a problem is more important than the problem itself - so always think positively. - **Norman Vincent Peale**

Problem E. File Fix-it

Problem

On Unix computers, data is stored in *directories*. There is one *root directory*, and this might have several directories contained inside of it, each with different names. These directories might have even more directories contained inside of them, and so on.
A directory is uniquely identified by its name and its parent directory (the directory it is directly contained in). This is usually encoded in a *path*, which consists of several parts each preceded by a forward slash ('/'). The final part is the name of the directory, and everything else gives the path of its parent directory. For example, consider the path:
/home/gcj/finals

This refers to the directory with name "finals" in the directory described by "/home/gcj", which in turn refers to the directory with name "gcj" in the directory described by the path "/home". In this path, there is only one part, which means it refers to the directory with the name "home" in the root directory.
To create a directory, you can use the *mkdir* command. You specify a path, and then *mkdir* will create the directory described by that path, but *only if* the parent directory already exists. For example, if you wanted to create the "/home/gcj/finals" and "/home/gcj/quals" directories from scratch, you would need four commands:

mkdir /home
mkdir /home/gcj
mkdir /home/gcj/finals
mkdir /home/gcj/quals

Given the full set of directories already existing on your computer, and a set of new directories you want to create if they do not already exist, how many *mkdir* commands do you need to use?

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each case begins with a line containing two integers **N** and **M**, separated by a space.
The next **N** lines each give the path of one directory that already exists on your computer. This list will include every directory already on your computer other than the root directory. (The root directory is on every computer, so there is no need to list it explicitly.)
The next **M** lines each give the path of one directory that you want to create.
Each of the paths in the input is formatted as in the problem statement above. Specifically, a path consists of one or more lower-case alpha-numeric strings (i.e., strings containing only the symbols 'a'-'z' and '0'-'9'), each preceded by a single forward slash. These alpha-numeric strings are never empty.

Output

For each test case, output one line containing "Case #x: y", where x is the case number (starting from 1)

and y is the number of *mkdir* you need.

Limits

1 ≤ **T** ≤ 100.
No path will have more than 100 characters in it.
No path will appear twice in the list of directories already on your computer, or in the list of directories you wish to create. A path may appear once in both lists however. (See example case #2 below).
If a directory is listed as being on your computer, then its parent directory will also be listed, unless the parent is the root directory.
The input file will be no longer than 100,000 bytes in total.

Limit

0 ≤ **N** ≤ 100.
1 ≤ **M** ≤ 100.

Sample

Input

```
3
0 2
/home/gcj/finals
/home/gcj/quals
2 1
/chicken
/chicken/egg
/chicken
1 3
/a
/a/b
/a/c
/b/b
```

Output

```
Case #1: 4
Case #2: 0
Case #3: 4
```

It's not that I'm so smart, it's just that I stay with problems longer. - **Albert Einstein**

Problem F. Alien Language

Problem

After years of study, scientists at Google Labs have discovered an alien language transmitted from a faraway planet. The alien language is very unique in that every word consists of exactly L lowercase letters. Also, there are exactly D words in this language.

Once the dictionary of all the words in the alien language was built, the next breakthrough was to discover that the aliens have been transmitting messages to Earth for the past decade. Unfortunately, these signals are weakened due to the distance between our two planets and some of the words may be misinterpreted. In order to help them decipher these messages, the scientists have asked you to devise an algorithm that will determine the number of possible interpretations for a given pattern.

A pattern consists of exactly L tokens. Each token is either a single lowercase letter (the scientists are very sure that this is the letter) or a group of unique lowercase letters surrounded by parenthesis (and). For example: (ab)d(dc) means the first letter is either a or b, the second letter is definitely d and the last letter is either d or c. Therefore, the pattern (ab)d(dc) can stand for either one of these 4 possibilities: add, adc, bdd, bdc.

Input

The first line of input contains 3 integers, L, D and N separated by a space. D lines follow, each containing one word of length L. These are the words that are known to exist in the alien language. N test cases then follow, each on its own line and each consisting of a pattern as described above. You may assume that all known words provided are unique.

The first line of input contains 3 integers, L, D and N separated by a space. D lines follow, each containing one word of length L. These are the words that are known to exist in the alien language. N test cases then follow, each on its own line and each consisting of a pattern as described above. You may assume that all known words provided are unique.

Output

For each test case, output

Case #X: K

where X is the test case number, starting from 1, and K indicates how many words in the alien language match the pattern.

Limits

$1 \le \mathbf{L} \le 15$

$1 \le \mathbf{D} \le 5000$

$1 \le \mathbf{N} \le 500$

Sample

Input	Output
3 5 4 abc bca dac dbc cba (ab)(bc)(ca) abc (abc)(abc)(abc) (zyx)bc	Case #1: 2 Case #2: 1 Case #3: 3 Case #4: 0