Problem A. **Alien Numbers**

This contest is open for practice. You can try every problem as many times as you like, though we won't keep track of which problems you solve. Read the Quick-Start Guide to get started.

Small input 40 points	Solve A-small
Large input 80 points	Solve A-large

Problem

The decimal numeral system is composed of ten digits, which we represent as "0123456789" (the digits in a system are written from lowest to highest). Imagine you have discovered an alien numeral system composed of some number of digits, which may or may not be the same as those used in decimal. For example, if the alien numeral system were represented as "oF8", then the numbers one through ten would be (F, 8, Fo, FF, F8, 8o, 8F, 88, Foo, FoF). We would like to be able to work with numbers in arbitrary alien systems. More generally, we want to be able to convert an arbitrary number that's written in one alien system into a second alien system.

Input

The first line of input gives the number of cases, **N**. **N** test cases follow. Each case is a line formatted as

```
alien_number source_language target_language
```

Each language will be represented by a list of its digits, ordered from lowest to highest value. No digit will be repeated in any representation, all digits in the alien number will be present in the source language, and the first digit of the alien number will not be the lowest valued digit of the source language (in other words, the alien numbers have no leading zeroes). Each digit will either be a number 0-9, an uppercase or lowercase letter, or one of the following symbols `!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~`

Output

For each test case, output one line containing "Case #**x**: " followed by the alien number translated from the source language to the target language.

Limits

1 ≤ **N** ≤ 100.

Small dataset

1 ≤ num digits in **alien_number** ≤ 4,
2 ≤ num digits in **source_language** ≤ 16,
2 ≤ num digits in **target_language** ≤ 16.

Large dataset

1 ≤ **alien_number** (in decimal) ≤ 1000000000,
2 ≤ num digits in **source_language** ≤ 94,
2 ≤ num digits in **target_language** ≤ 94.

Sample

Input	Output
4 9 0123456789 oF8 Foo oF8 0123456789 13 0123456789abcdef 01 CODE O!CDE? A?JM!.	Case #1: Foo Case #2: 9 Case #3: 10011 Case #4: JAM!

Practice Problems

A. Alien Numbers
B. Always Turn Left
C. Egg Drop
D. Shopping Plan

Questions asked

Submissions

Alien Numbers

40pt	Not attempted **320/432 users** correct (74%)
80pt	Not attempted **271/338 users** correct (80%)

Always Turn Left

40pt	Not attempted **108/135 users** correct (80%)
80pt	Not attempted **96/114 users** correct (84%)

Egg Drop

40pt	Not attempted **56/82 users** correct (68%)
80pt	Not attempted **26/53 users** correct (49%)

Shopping Plan

40pt	Not attempted **43/67 users** correct (64%)
80pt	Not attempted **16/52 users** correct (31%)

Top Scores

sclo	480
jdmetz	480
lordmonsoon	480
ardiankp	480
krijgertje	480
ilyakor	400
Edu	400
Jonick	400
zibada	400
gpascale	400

Problem B. Always Turn Left

This contest is open for practice. You can try every problem as many times as you like, though we won't keep track of which problems you solve. Read the Quick-Start Guide to get started.

Small input 40 points	Solve B-small
Large input 80 points	Solve B-large

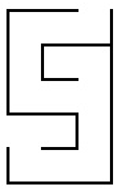
Problem

You find yourself standing outside of a perfect maze. A maze is defined as "perfect" if it meets the following conditions:

1. It is a rectangular grid of rooms, **R** rows by **C** columns.
2. There are exactly two openings on the outside of the maze: the entrance and the exit. The entrance is always on the north wall, while the exit could be on any wall.
3. There is exactly one path between any two rooms in the maze (that is, exactly one path that does not involve backtracking).

You decide to solve the perfect maze using the "always turn left" algorithm, which states that you take the leftmost fork at every opportunity. If you hit a dead end, you turn right twice (180 degrees clockwise) and continue. (If you were to stick out your left arm and touch the wall while following this algorithm, you'd solve the maze without ever breaking contact with the wall.) Once you finish the maze, you decide to go the extra step and solve it again (still always turning left), but starting at the exit and finishing at the entrance.

The path you take through the maze can be described with three characters: 'W' means to walk forward into the next room, 'L' means to turn left (or counterclockwise) 90 degrees, and 'R' means to turn right (or clockwise) 90 degrees. You begin outside the maze, immediately adjacent to the entrance, facing the maze. You finish when you have stepped outside the maze through the exit. For example, if the entrance is on the north and the exit is on the west, your path through the following maze would be `WRWWLWWLWWLWLWRRWRWWWRWWRWLW`:



If the entrance and exit were reversed such that you began outside the west wall and finished out the north wall, your path would be `WWRRWLWLWWLWWLWWRWWRWWLW`. Given your two paths through the maze (entrance to exit and exit to entrance), your code should return a description of the maze.

Input

The first line of input gives the number of cases, **N**. **N** test cases follow. Each case is a line formatted as

```
entrance_to_exit exit_to_entrance
```

All paths will be at least two characters long, consist only of the characters 'W', 'L', and 'R', and begin and end with 'W'.

Output

For each test case, output one line containing "Case #**x**:" by itself. The next **R** lines give a description of the **R** by **C** maze. There should be **C** characters in each line, representing which directions it is possible to walk from that room. Refer to the following legend:

Character	Can walk north?	Can walk south?	Can walk west?	Can walk east?
1	Yes	No	No	No
2	No	Yes	No	No
3	Yes	Yes	No	No
4	No	No	Yes	No
5	Yes	No	Yes	No
6	No	Yes	Yes	No
7	Yes	Yes	Yes	No

8	No	No	No	Yes
9	Yes	No	No	Yes
a	No	Yes	No	Yes
b	Yes	Yes	No	Yes
c	No	No	Yes	Yes
d	Yes	No	Yes	Yes
e	No	Yes	Yes	Yes
f	Yes	Yes	Yes	Yes

Limits

1 ≤ **N** ≤ 100.

Small dataset

2 ≤ len(entrance_to_exit) ≤ 100,
2 ≤ len(exit_to_entrance) ≤ 100.

Large dataset

2 ≤ len(entrance_to_exit) ≤ 10000,
2 ≤ len(exit_to_entrance) ≤ 10000.

Sample

Input
```
2
WRWWLWWLWWLWLWRRWRWWWRWWRWLW WWRRWLWLWWLWWLWWRWWRWWLW
WW WW
```

Output
```
Case #1:
ac5
386
9c7
e43
9c5
Case #2:
3
```

Practice Problems

A. Alien Numbers
B. Always Turn Left
C. Egg Drop
D. Shopping Plan

Questions asked

Submissions

Alien Numbers

40pt	Not attempted **320/432 users** correct (74%)
80pt	Not attempted **271/338 users** correct (80%)

Always Turn Left

40pt	Not attempted **108/135 users** correct (80%)
80pt	Not attempted **96/114 users** correct (84%)

Egg Drop

40pt	Not attempted **56/82 users** correct (68%)
80pt	Not attempted **26/53 users** correct (49%)

Shopping Plan

40pt	Not attempted **43/67 users** correct (64%)
80pt	Not attempted **16/52 users** correct (31%)

Top Scores

sclo	480
jdmetz	480
lordmonsoon	480
ardiankp	480
krijgertje	480
ilyakor	400
Edu	400
Jonick	400
zibada	400
gpascale	400

Problem C. Egg Drop

This contest is open for practice. You can try every problem as many times as you like, though we won't keep track of which problems you solve. Read the Quick-Start Guide to get started.

Small input 40 points	Solve C-small
Large input 80 points	Solve C-large

Problem

Imagine that you are in a building with **F** floors (starting at floor 1, the lowest floor), and you have a large number of identical eggs, each in its own identical protective container. For each floor in the building, you want to know whether or not an egg dropped from that floor will break. If an egg breaks when dropped from floor *i*, then all eggs are guaranteed to break when dropped from any floor *j* ≥ *i*. Likewise, if an egg doesn't break when dropped from floor *i*, then all eggs are guaranteed to never break when dropped from any floor *j* ≤ *i*.

We can define *Solvable(F, D, B)* to be true if and only if there exists an algorithm to determine whether or not an egg will break when dropped from any floor of a building with **F** floors, with the following restrictions: you may drop a maximum of **D** eggs (one at a time, from any floors of your choosing), and you may break a maximum of **B** eggs. You can assume you have at least **D** eggs in your possession.

Input

The first line of input gives the number of cases, **N**. **N** test cases follow. Each case is a line formatted as:

```
F D B
```

Solvable(F, D, B) is guaranteed to be true for all input cases.

Output

For each test case, output one line containing "Case #**x**: " followed by three space-separated integers: F_{max}, D_{min}, and B_{min}. The definitions are as follows:

- F_{max} is defined as the largest value of **F'** such that *Solvable(**F'**, D, B)* is true, or -1 if this value would be greater than or equal to 2^{32} (4294967296).
(In other words, F_{max} = -1 if and only if *Solvable(2^{32}, D, B)* is true.)
- D_{min} is defined as the smallest value of **D'** such that *Solvable(F, **D'**, B)* is true.
- B_{min} is defined as the smallest value of **B'** such that *Solvable(F, D, **B'**)* is true.

Limits

1 ≤ **N** ≤ 100.

Small dataset

1 ≤ **F** ≤ 100,
1 ≤ **D** ≤ 100,
1 ≤ **B** ≤ 100.

Large dataset

1 ≤ **F** ≤ 2000000000,
1 ≤ **D** ≤ 2000000000,
1 ≤ **B** ≤ 2000000000.

Sample

Input	Output
2 3 3 3 7 5 3	Case #1: 7 2 1 Case #2: 25 3 2

Practice Problems

A. Alien Numbers
B. Always Turn Left
C. Egg Drop
D. Shopping Plan

Questions asked

Submissions

Alien Numbers
40pt Not attempted **320/432 users** correct (74%)
80pt Not attempted **271/338 users** correct (80%)

Always Turn Left
40pt Not attempted **108/135 users** correct (80%)
80pt Not attempted **96/114 users** correct (84%)

Egg Drop
40pt Not attempted **56/82 users** correct (68%)
80pt Not attempted **26/53 users** correct (49%)

Shopping Plan
40pt Not attempted **43/67 users** correct (64%)
80pt Not attempted **16/52 users** correct (31%)

Top Scores

sclo	480
jdmetz	480
lordmonsoon	480
ardiankp	480
krijgertje	480
ilyakor	400
Edu	400
Jonick	400
zibada	400
gpascale	400

Problem D. Shopping Plan

Small input 40 points — Solve D-small

Large input 80 points — Solve D-large

Problem

You have a list of items you need to buy today, and you know the locations (represented as points on a cartesian grid) of a few stores in the area. You also know which of these stores are selling each item on your list, and at what price each store sells it. Given the price of gas, what is the minimum amount you need to spend in order to buy all the items on your shopping list and then drive back home? You start and end the journey at your house, which is located at (0,0).

To make matters interesting, some of the items on your list may be perishable. Whenever you make a purchase that includes one or more perishable items, you cannot drive to another store without first stopping back at your house. Every item on your shopping list is guaranteed to be sold by at least one store, so the trip will always be possible.

Input

The first line of input gives the number of cases, **N**. **N** test cases follow. Each case starts with a line formatted as

```
num_items num_stores price_of_gas
```

The next line contains the **num_items** items on your shopping list. The items will be space separated, and each item will consist of only lowercase letters. If an item is perishable, its name will be followed by a single exclamation point. There will be no duplicate items on your list. The next **num_stores** lines will each be formatted as

```
x_pos y_pos item1:price1 item2:price2 ...
```

Each of these lines gives the location of one store, along with the items available at that store and their corresponding prices. Only items which are on your shopping list will appear in these lists. Perishable items will not end with exclamation points on these lists. No item will be repeated in a store's list. Each store will offer at least one item for sale. No two stores will be at the same location, and no store will be located at (0,0).

Output

For each test case, output one line containing "Case #**x**: " followed by the minimum possible cost of the trip, rounded to seven decimal places. Don't forget about **price_of_gas**, which is the amount of money you must spend per unit distance that you drive.

Limits

1 ≤ **N** ≤ 100,
0 ≤ **price_of_gas** ≤ 1000,
-1000 ≤ **x_pos** ≤ 1000,
-1000 ≤ **y_pos** ≤ 1000,
1 ≤ price of each item ≤ 1000.

Small dataset

1 ≤ **num_items** ≤ 5,
1 ≤ **num_stores** ≤ 10.

Large dataset

1 ≤ **num_items** ≤ 15,
1 ≤ **num_stores** ≤ 50.

Sample

Input	Output
2 1 2 10 cookies	Case #1: 400.0000000 Case #2: 519.2920690

```
0 2 cookies:400
4 0 cookies:320
3 3 5
cookies milk! cereal
0 2 cookies:360 cereal:110
4 0 cereal:90 milk:150
-3 -3 milk:200 cookies:200
```