Submissions

Speaking in Tongues

15pt	Not attempted **17356/19464 users** correct (89%)

Dancing With the Googlers

10pt	Not attempted **12384/13899 users** correct (89%)
10pt	Not attempted **10762/12138 users** correct (89%)

Recycled Numbers

10pt	Not attempted **11747/12327 users** correct (95%)
15pt	Not attempted **6811/10604 users** correct (64%)

Hall of Mirrors

15pt	Not attempted **551/879 users** correct (63%)
25pt	Not attempted **184/259 users** correct (71%)

Top Scores

hos.lyric	100
qnighy	100
DjinnKahn	100
levlam	100
iwiskimo	100
mystic	100
TripleM	100
aleksey	100
royf	100
krijgertje	100

Problem A. Speaking in Tongues

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
15 points | Solve A-small

Problem

We have come up with the best possible language here at Google, called Googlerese. To translate text into Googlerese, we take any message and replace each English letter with another English letter. This mapping is *one-to-one* and *onto*, which means that the same input letter always gets replaced with the same output letter, and different input letters always get replaced with different output letters. A letter may be replaced by itself. Spaces are left as-is.

For example (and here is a hint!), our awesome translation algorithm includes the following three mappings: 'a' -> 'y', 'o' -> 'e', and 'z' -> 'q'. This means that "a zoo" will become "y qee".

Googlerese is based on the best possible replacement mapping, and we will never change it. It will always be the same. In every test case. We will not tell you the rest of our mapping because that would make the problem too easy, but there are a few examples below that may help.

Given some text in Googlerese, can you translate it to back to normal text?

Solving this problem

Usually, Google Code Jam problems have 1 Small input and 1 Large input. This problem has only **1 Small input**. Once you have solved the Small input, you have finished solving this problem.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow, one per line.

Each line consists of a string **G** in Googlerese, made up of one or more words containing the letters 'a' - 'z'. There will be exactly one space (' ') character between consecutive words and no spaces at the beginning or at the end of any line.

Output

For each test case, output one line containing "Case #**X**: **S**" where **X** is the case number and **S** is the string that becomes **G** in Googlerese.

Limits

$1 \leq \mathbf{T} \leq 30$.
G contains at most 100 characters.
None of the text is guaranteed to be valid English.

Sample

Input
```
3
ejp mysljylc kd kxveddknmc re jsicpdrysi
rbcpc ypc rtcsra dkh wyfrepkym veddknkmkrkcd
de kr kd eoya kw aej tysr re ujdr lkgc jv
```

Output
```
Case #1: our language is impossible to understand
Case #2: there are twenty six factorial possibilities
Case #3: so it is okay if you want to just give up
```

Submissions

Speaking in Tongues
15pt Not attempted **17356/19464 users** correct (89%)

Dancing With the Googlers
10pt Not attempted **12384/13899 users** correct (89%)
10pt Not attempted **10762/12138 users** correct (89%)

Recycled Numbers
10pt Not attempted **11747/12327 users** correct (95%)
15pt Not attempted **6811/10604 users** correct (64%)

Hall of Mirrors
15pt Not attempted **551/879 users** correct (63%)
25pt Not attempted **184/259 users** correct (71%)

Top Scores

hos.lyric	100
qnighy	100
DjinnKahn	100
levlam	100
iwiskimo	100
mystic	100
TripleM	100
aleksey	100
royf	100
krijgertje	100

Problem B. Dancing With the Googlers

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
10 points
Solve B-small

Large input
10 points
Solve B-large

Problem

You're watching a show where Googlers (employees of Google) dance, and then each dancer is given a *triplet of scores* by three judges. Each triplet of scores consists of three integer scores from 0 to 10 inclusive. The judges have very similar standards, so it's *surprising* if a triplet of scores contains two scores that are 2 apart. No triplet of scores contains scores that are more than 2 apart.

For example: (8, 8, 8) and (7, 8, 7) are not surprising. (6, 7, 8) and (6, 8, 8) are surprising. (7, 6, 9) will never happen.

The *total points* for a Googler is the sum of the three scores in that Googler's triplet of scores. The *best result* for a Googler is the maximum of the three scores in that Googler's triplet of scores. Given the total points for each Googler, as well as the number of surprising triplets of scores, what is the maximum number of Googlers that could have had a best result of at least **p**?

For example, suppose there were 6 Googlers, and they had the following total points: `29, 20, 8, 18, 18, 21`. You remember that there were 2 surprising triplets of scores, and you want to know how many Googlers could have gotten a best result of 8 or better.

With those total points, and knowing that two of the triplets were surprising, the triplets of scores could have been:

```
10 9 10
6 6 8 (*)
2 3 3
6 6 6
6 6 6
6 7 8 (*)
```

The cases marked with a (*) are the surprising cases. This gives us 3 Googlers who got at least one score of 8 or better. There's no series of triplets of scores that would give us a higher number than 3, so the answer is 3.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each test case consists of a single line containing integers separated by single spaces. The first integer will be **N**, the number of Googlers, and the second integer will be **S**, the number of surprising triplets of scores. The third integer will be **p**, as described above. Next will be **N** integers $\mathbf{t_i}$: the total points of the Googlers.

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 maximum number of Googlers who could have had a best result of greater than or equal to **p**.

Limits

$1 \le \mathbf{T} \le 100$.
$0 \le \mathbf{S} \le N$.
$0 \le \mathbf{p} \le 10$.
$0 \le \mathbf{t_i} \le 30$.
At least **S** of the $\mathbf{t_i}$ values will be between 2 and 28, inclusive.

Small dataset

$1 \le \mathbf{N} \le 3$.

Large dataset

$1 \le \mathbf{N} \le 100$.

Sample

Input	Output

```
4                        Case #1: 3
3 1 5 15 13 11           Case #2: 2
3 0 8 23 22 21           Case #3: 1
2 1 1 8 0                Case #4: 3
6 2 8 29 20 8 18 18 21
```

Submissions

Speaking in Tongues

15pt	Not attempted **17356/19464 users** correct (89%)

Dancing With the Googlers

10pt	Not attempted **12384/13899 users** correct (89%)
10pt	Not attempted **10762/12138 users** correct (89%)

Recycled Numbers

10pt	Not attempted **11747/12327 users** correct (95%)
15pt	Not attempted **6811/10604 users** correct (64%)

Hall of Mirrors

15pt	Not attempted **551/879 users** correct (63%)
25pt	Not attempted **184/259 users** correct (71%)

Top Scores

hos.lyric	100
qnighy	100
DjinnKahn	100
levlam	100
iwiskimo	100
mystic	100
TripleM	100
aleksey	100
royf	100
krijgertje	100

Problem C. Recycled 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 10 points	Solve C-small
Large input 15 points	Solve C-large

Problem

Do you ever become frustrated with television because you keep seeing the same things, recycled over and over again? Well I personally don't care about television, but I do sometimes feel that way about numbers.

Let's say a pair of distinct positive integers (n, m) is *recycled* if you can obtain m by moving some digits from the back of n to the front without changing their order. For example, (12345, 34512) is a recycled pair since you can obtain 34512 by moving 345 from the end of 12345 to the front. Note that n and m must have the same number of digits in order to be a recycled pair. Neither n nor m can have leading zeros.

Given integers **A** and **B** with the same number of digits and no leading zeros, how many distinct recycled pairs (n, m) are there with $\mathbf{A} \le n < m \le \mathbf{B}$?

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each test case consists of a single line containing the integers **A** and **B**.

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 recycled pairs (n, m) with $\mathbf{A} \le n < m \le \mathbf{B}$.

Limits

$1 \le \mathbf{T} \le 50$.
A and **B** have the same number of digits.

Small dataset

$1 \le \mathbf{A} \le \mathbf{B} \le 1000$.

Large dataset

$1 \le \mathbf{A} \le \mathbf{B} \le 2000000$.

Sample

```
Input          Output

4              Case #1: 0
1 9            Case #2: 3
10 40          Case #3: 156
100 500        Case #4: 287
1111 2222
```

Are we sure about the output to Case #4?

Yes, we're sure about the output to Case #4.

Submissions

Speaking in Tongues

15pt	Not attempted **17356/19464 users** correct (89%)

Dancing With the Googlers

10pt	Not attempted **12384/13899 users** correct (89%)
10pt	Not attempted **10762/12138 users** correct (89%)

Recycled Numbers

10pt	Not attempted **11747/12327 users** correct (95%)
15pt	Not attempted **6811/10604 users** correct (64%)

Hall of Mirrors

15pt	Not attempted **551/879 users** correct (63%)
25pt	Not attempted **184/259 users** correct (71%)

Top Scores

hos.lyric	100
qnighy	100
DjinnKahn	100
levlam	100
iwiskimo	100
mystic	100
TripleM	100
aleksey	100
royf	100
krijgertje	100

Problem D. Hall of Mirrors

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 15 points	Solve D-small
Large input 25 points	Solve D-large

Problem

You live in a 2-dimensional plane, and one of your favourite places to visit is the Hall of Mirrors. The Hall of Mirrors is a room (a 2-dimensional room, of course) that is laid out in a grid. Every square on the grid contains either a square mirror, empty space, or you. You have width 0 and height 0, and you are located in the exact centre of your grid square.

Despite being very small, you can see your reflection when it is reflected back to you exactly. For example, consider the following layout, where `'#'` indicates a square mirror that completely fills its square, `'.'` indicates empty space, and the capital letter `'X'` indicates you are in the center of that square:

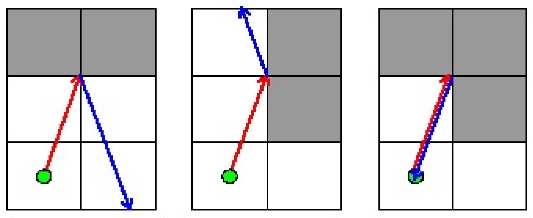
```
######
#..X.#
#.#..#
#...##
######
```

If you look straight up or straight to the right, you will be able to see your reflection.

Unfortunately in the Hall of Mirrors it is very foggy, so you can't see further than **D** units away. Suppose **D**=3. If you look up, your reflection will be 1 unit away (0.5 to the mirror, and 0.5 back). If you look right, your reflection will be 3 units away (1.5 to the mirror, and 1.5 back), and you will be able to see it. If you look down, your reflection will be 5 units away and you won't be able to see it.

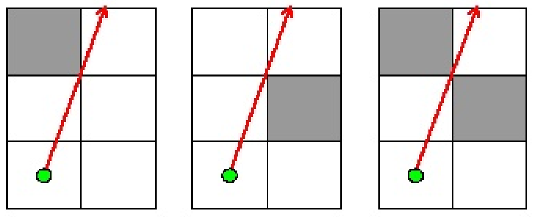
It's important to understand how light travels in the Hall of Mirrors. Light travels in a straight line until it hits a mirror. If light hits any part of a mirror but its corner, it will be reflected in the normal way: it will bounce off with an angle of reflection equal to the angle of incidence. If, on the other hand, the light would touch the corner of a mirror, the situation is more complicated. The following diagrams explain the cases:

In the following cases, light approaches a corner and is reflected, changing its direction:



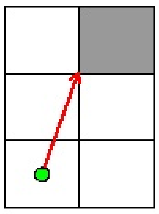
In the first two cases, light approached two adjacent mirrors at the point where they met. Light was reflected in the same way as if it had hit the middle of a long mirror. In the third case, light approached the corners of three adjacent mirrors, and returned in exactly the direction it came from.

In the following cases, light approaches the corners of one or more mirrors, but does not bounce, and instead continues in the same direction:



This happens when light reaches distance 0 from the corner of a mirror, but would not have to pass through the mirror in order to continue in the same direction. In this way, a ray of light can pass between two mirrors that are diagonally adjacent to each other -- effectively going through a space of size 0. Good thing it's of size 0 too, so it fits!

In the final case, light approaches the corner of one mirror and is destroyed:



The mirror was in the path of the light, and the ray of light didn't approach the corners of any other mirrors.

Note that light stops when it hits you, but it has to hit the exact centre of your grid square.

How many images of yourself can you see?

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each test case starts with a line containing three space-separated integers, **H**, **W** and **D**. **H** lines follow, and each contains **W** characters. The characters constitute a map of the Hall of Mirrors for that test case, as described above.

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 reflections of yourself you can see.

Limits

1 ≤ **T** ≤ 100.
3 ≤ **H**, **W** ≤ 30.
1 ≤ **D** ≤ 50.
All characters in each map will be `'#'`, `'.'`, or `'X'`.
Exactly one character in each map will be `'X'`.
The first row, the last row, the first column and the last column of each map will be entirely filled with `'#'` characters.

Small dataset

There will be no more than `2W+2H-4` `'#'` characters.

Large dataset

The restriction from the Small dataset does not apply.

Sample

```
Input      Output

6          Case #1: 4
3 3 1      Case #2: 8
###        Case #3: 68
#X#        Case #4: 0
###        Case #5: 2
3 3 2      Case #6: 28
###
#X#
###
4 3 8
###
#X#
#.#
###
7 7 4
#######
#.....#
#.....#
#..X..#
#....##
#.....#
#######
5 6 3
######
#..X.#
#.#..#

#...##
######
5 6 10
######
```

```
#..X.#
#.#..#
#...##
######
```

In the first case, light travels exactly distance 1 if you look directly up, down, left or right.

In the second case, light travels distance 1.414... if you look up-right, up-left, down-right or down-left. Since light does not travel through you, looking directly up only shows you one image of yourself.

In the fifth case, while the nearby mirror is close enough to reflect light back to you, light that hits the corner of the mirror is destroyed rather than reflected.

Submissions

Password Problem

10pt	Not attempted **3506/3846 users** correct (91%)
10pt	Not attempted **2329/3371 users** correct (69%)

Kingdom Rush

15pt	Not attempted **1908/3460 users** correct (55%)
18pt	Not attempted **1616/1844 users** correct (88%)

Cruise Control

17pt	Not attempted **65/312 users** correct (21%)
30pt	Not attempted **22/42 users** correct (52%)

Top Scores

SnapDragon	100
pieguy	100
dzhulgakov	100
squark	100
wata	100
Plagapong	100
omeometo	100
ploh	100
cedriclin	100
MRoizner	100

Problem A. Password Problem

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 10 points	Solve A-small
Large input 10 points	Solve A-large

Problem

I have a really long password, and sometimes I make a mistake when I type it. Right now I've typed part of my password, but I might have made some mistakes. In particular, I might have pressed the wrong key while typing one or more of the previous characters. Given how likely I was to get each character right, what should I do?

I have three options:

1. Finish typing the password, then press "enter". I know I'll type the rest of the characters perfectly. If it turns out that one of the earlier characters was wrong, I'll have to retype the whole thing and hit "enter" again -- but I know I'll get it right the second time.
2. Hit "backspace" some number of times, deleting the last character(s) I typed, and then complete the password and press "enter" as in option 1. If one of the characters I didn't delete was wrong, I'll have to retype the whole thing and press "enter", knowing I'll get it right the second time.
3. Give up by pressing "enter", retyping the password from the start, and pressing "enter" again. I know I'll get it right this time.

I want to minimize the *expected* number of keystrokes needed. Each character in the password costs 1 keystroke; each "backspace" costs 1 keystroke; pressing "enter" to complete an attempt or to give up costs 1 keystroke.

Note: The "expected" number of keystrokes is the average number of keystrokes that would be needed if the same situation occurred a very large number of times. See the example below.

Example

Suppose my password is "guest" and I have already typed the first two characters, but I had a 40% chance of *making a mistake* when typing each of them. Then there are four cases:

- I typed "gu" without error. This occurs with probability 0.6 * 0.6 = 0.36.
- I typed the 'g' correctly but I made a mistake typing the 'u'. Then I have two letters typed still, but the second one is wrong: "gX". (Here, the 'X' character represents a mistyped letter.) This occurs with probability 0.6 * 0.4 = 0.24.
- I typed the 'u' correctly but I made a mistake typing the 'g': "Xu". This occurs with probability 0.4 * 0.6 = 0.24.
- I made a mistake typing both letters, so I have two incorrect letters: "XX". This occurs with probability 0.4 * 0.4 = 0.16.

I don't know how many mistakes I actually made, but for any strategy, I can calculate the *expected* number of keys required to use it. This is shown in the table below.

	"gu"	"gX"	"Xu"	"XX"	Expected
Probability	0.36	0.24	0.24	0.16	-
Keystrokes if I keep typing	4	10	10	10	7.84
Keystrokes if I press backspace once	6	6	12	12	8.4
Keystrokes if I press backspace twice	8	8	8	8	8
Keystrokes if I press enter right away	7	7	7	7	7

If I keep typing, then there is an 0.36 probability that I will need 4 keystrokes, and an 0.64 probability that I will need 10 keystrokes. If I repeated the trial many times, then I would use 4 keystrokes 36% of the time, and 10 keystrokes the remaining 64% of the time, so the average number of keystrokes needed would be 0.36 * 4 + 0.64 * 10 = 7.84. In this case however, it is better to just press enter right away, which requires 7 keystrokes.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each test case begins with a line containing two integers, **A** and **B**. **A** is the number of characters that I have already typed, and **B** is the total number of characters in my password.

This is followed by a line containing **A** real numbers: p_1, p_2, ..., $p_{\mathbf{A}}$. p_i represents the probability that I *correctly* typed the ith letter in my password. These real numbers will consist of decimal digits and at most one decimal point. The decimal point will never be the first or the last character in a number.

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 expected number of additional keystrokes I need, not counting the letters I have typed so far, and assuming I choose the optimal strategy. y must be correct to within an absolute or relative error of 10^{-6}.

Limits

$1 \leq \mathbf{T} \leq 20$.
$0 \leq p_i \leq 1$ for all i.

Small dataset

$1 \leq \mathbf{A} \leq 3$.
$\mathbf{A} < \mathbf{B} \leq 100$.

Large dataset

$1 \leq \mathbf{A} \leq 99999$.
$\mathbf{A} < \mathbf{B} \leq 100000$.

Sample

Input	Output

3
2 5
0.6 0.6
1 20
1
3 4
1 0.9 0.1
	Case #1: 7.000000
	Case #2: 20.000000
	Case #3: 4.500000

Submissions

Password Problem

10pt	Not attempted **3506/3846 users** correct (91%)
10pt	Not attempted **2329/3371 users** correct (69%)

Kingdom Rush

15pt	Not attempted **1908/3460 users** correct (55%)
18pt	Not attempted **1616/1844 users** correct (88%)

Cruise Control

17pt	Not attempted **65/312 users** correct (21%)
30pt	Not attempted **22/42 users** correct (52%)

Top Scores

SnapDragon	100
pieguy	100
dzhulgakov	100
squark	100
wata	100
Plagapong	100
omeometo	100
ploh	100
cedriclin	100
MRoizner	100

Problem B. Kingdom Rush

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 15 points	Solve B-small
Large input 18 points	Solve B-large

Problem

Ryan is playing Kingdom Rush, a single-player tower defense game developed by Ironhide Game Studio. In Kingdom Rush, players earn stars by completing levels, in a way described below. Having more stars makes the player more powerful; so while Ryan might not be able to complete level 2 right away, he might be able to complete it after earning stars from level 1.

The real game Kingdom Rush doesn't work in quite the same way as this problem. It isn't important to have played the game in order to solve the problem.

In this problem's version of Kingdom Rush, when a player completes a level, he or she is given a 1-star rating or a 2-star rating. That rating might allow the player to earn stars as follows:

- If the player has never completed the level before and completes it with a 1-star rating, that player earns 1 star.
- If the player has never completed the level before and completes it with a 2-star rating, that player earns 2 stars.
- If the player has only completed the level before with a 1-star rating and completes it this time with a 2-star rating, the player earns 1 more star.

Otherwise there is no way for a player to earn stars.

Ryan might not be able to complete every level right away. For each level, before he can complete it with a 1-star rating, he needs to have earned a certain number of stars; and he will need a larger or equal number of stars to complete that level with a 2-star rating.

For example, suppose there are two levels:

- Level 1 requires 0 stars to complete with a 1-star rating, and 1 star to complete with a 2-star rating.
- Level 2 requires 0 stars to complete with a 1-star rating, and 2 stars to complete with a 2-star rating.

Here's a possible series of events for Ryan:

1. Ryan starts with 0 stars. He can choose to complete either level 1 or level 2 with a 1-star rating. He chooses to complete level 1 with a 1-star rating. Now he has 1 star.
2. Now Ryan can either complete level 2 with a 1-star rating, or level 1 with a 2-star rating. He chooses to complete level 1 with a 2-star rating. Now he has 2 stars.
3. Now Ryan can complete level 2 with a 2-star rating. He does that, and now he has 4 stars.
4. Now he is done, having completed all levels with 2-star ratings and earned 4 stars (2 per level). He has completed levels 3 times: level 1 twice, and level 2 once.

Ryan is great at tower defense games, but he needs some help to beat Kingdom Rush as quickly as possible. Your job is to figure out how many times he needs to complete levels in order to earn a 2-star rating on every level.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each test case starts with a line containing a single integer **N**, indicating how many levels are in the game. **N** lines follow. The ith line contains two integers $\mathbf{a_i}$ and $\mathbf{b_i}$: the number of stars it takes to earn a one-star rating or a two-star rating, respectively, on level i.

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 minimum number of times Ryan must complete levels in order to have earned a 2-star rating on every level. If it is impossible for Ryan to earn a 2-star rating on every level, y should instead be the string "Too Bad" (without the " characters, but with that exact capitalization). This indicates that Ryan is too bad at Kingdom Rush to finish the whole game.

Limits

1 ≤ **T** ≤ 100.
0 ≤ $\mathbf{a_i}$ ≤ $\mathbf{b_i}$ ≤ 2001.

Small dataset

1 ≤ **N** ≤ 10.

Large dataset

1 ≤ **N** ≤ 1000.

Sample

```
Input     Output

4         Case #1: 3
2         Case #2: 3
0 1       Case #3: Too Bad
0 2       Case #4: 6
3
2 2
0 0
4 4
1
1 1
5
0 5
0 1
1 1
4 7
5 6
```

Kingdom Rush was created by Ironhide Game Studio. Ironhide Game Studio does not endorse and has no involvement with Google Code Jam.

Submissions

Password Problem

10pt	Not attempted **3506/3846 users** correct (91%)
10pt	Not attempted **2329/3371 users** correct (69%)

Kingdom Rush

15pt	Not attempted **1908/3460 users** correct (55%)
18pt	Not attempted **1616/1844 users** correct (88%)

Cruise Control

17pt	Not attempted **65/312 users** correct (21%)
30pt	Not attempted **22/42 users** correct (52%)

Top Scores

SnapDragon	100
pieguy	100
dzhulgakov	100
squark	100
wata	100
Plagapong	100
omeometo	100
ploh	100
cedriclin	100
MRoizner	100

Problem C. Cruise Control

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 17 points	Solve C-small
Large input 30 points	Solve C-large

Problem

Cruise control is a system that allows a car to go at a constant speed, while the driver controls only the steering wheel. The driver can, of course, turn off the cruise control to avoid collisions.

In this problem, we will consider a one-way road with two lanes, and **N** cars using cruise control on the road. Each car is 5 meters long and goes at some constant speed. A car can change lanes at any time if it would not cause the car to collide with some other car (touching does not count as collision). Assume that changing lanes is instantaneous and simply causes the car to switch to the other lane. We are interested in whether any driver will have to turn off cruise control eventually to avoid a collision, or is it possible for all of them to drive (possibly switching lanes, but at constant speed) without collisions indefinitely. Note that even though changing lanes is instantaneous, two cars driving side by side *cannot* exchange places by changing lanes at the same time.

Input

The first line of the input file gives the number of test cases, **T**. **T** test cases follow. Each test case begins with the number **N**. **N** lines follow, each describing a single car. Each line contains a character $\mathbf{C_i}$ (denoting whether the car is initially in the left or the right lane), two integers describing the speed $\mathbf{S_i}$ of the car (in meters per second), and the initial position $\mathbf{P_i}$ of the car (in meters), denoting the distance between the rear end of the car and some fixed line across the road. All the cars are moving away from this line, and no car is behind the line.

Output

For each test case output one line containing "Case #x: y", where x is the case number (starting from 1) and y is either the word "Possible" (quotes for clarity only), if the cars can drive at the given constant speeds indefinitely, or the maximum number of seconds they can drive before somebody has to change speed to avoid a collision. Answers accurate to within 10^{-5} absolute or relative error will be accepted.

Limits

$1 \le \mathbf{T} \le 30$.
$1 \le \mathbf{S_i} \le 1000$.
$0 \le \mathbf{P_i} \le 10000$.
Each of the $\mathbf{C_i}$ characters will be either *L*, denoting the left lane, or *R*, denoting the right lane.
Initially the cars' positions are such that they do not collide, that is, if two cars *i* and *j* have the same initial starting lane (that is, $C_i = C_j$), then $|P_i - P_j| \ge 5$.

Small dataset

$1 \le \mathbf{N} \le 6$.

Large dataset

$1 \le \mathbf{N} \le 50$.

Sample

```
Input          Output

4              Case #1: Possible
2              Case #2: 10
L 5 10         Case #3: 1.4
L 100 0        Case #4: 12
3
L 100 0
R 100 0
L 50 505

6
L 30 0
```

```
R 30 2
L 10 39
R 10 42
L 25 13
L 15 29
4
L 4 0
L 2 29
L 1 35
L 1 44
```

In the first case, the faster car can shift over to the right lane and easily overtake the slower one. In the second case, the two cars driving side-by-side at 100 m/s will reach the car going 50 m/s in 10 seconds, and somebody will have to change speed, as both lanes will be blocked.

Submissions

Safety in Numbers

10pt	Not attempted **2687/5608 users** correct (48%)
11pt	Not attempted **2008/2680 users** correct (75%)

Tide Goes In, Tide Goes Out

18pt	Not attempted **682/892 users** correct (76%)
18pt	Not attempted **619/670 users** correct (92%)

Equal Sums

6pt	Not attempted **2257/2531 users** correct (89%)
37pt	Not attempted **149/853 users** correct (17%)

Top Scores

Gennady.Korotkevich	100
bmerry	100
hansonw	100
marcina	100
ZhukovDmitry	100
random.johnnyh	100
yeputons	100
rng..58	100
pashka	100
mikhailOK	100

Problem A. Safety in 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 10 points	Solve A-small
Large input 11 points	Solve A-large

Problem

There are **N** contestants in a reality TV show. Each contestant is assigned a point value by the judges and receives votes from the audience. The point value given by the judges and the audience's votes are combined to form a *final score* for the contestant, in the following way:

Let X be the sum of the judge-assigned point values of all contestants. Now suppose a contestant got J points from the judges, and that she received a fraction Y (between 0 and 1, inclusive) of the audience's votes (Y might be, for example, 0.3). Then that contestant's final score is J+X*Y. Note that the sum of all contestants' audience vote fractions must be 1.

The contestant with the lowest score is eliminated.

Given the points contestants got from judges, your job is to find out, for each contestant, the minimum percentage of audience votes he/she must receive in order for him/her to be guaranteed **not to be eliminated**, no matter how the rest of the audience's votes are distributed.

If the lowest score is shared by multiple contestants, no contestants will be eliminated.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow, one per line. Each line starts with an integer **N**, the number of contestants, followed by a space, followed by **N** integers $\mathbf{s}_0$, $\mathbf{s}_1$, ..., $\mathbf{s}_{N-1}$, separated by single spaces. The integer $\mathbf{s}_i$ is the point value assigned to contestant `i` by the judges.

Output

For each test case, output one line containing "Case #x: " followed by **N** real numbers: $\mathbf{m}_i$s. The value x is the case number (starting from 1). The value $\mathbf{m}_i$ is the smallest percentage of audience votes required for contestant `i` to definitely avoid elimination.

Answers within an absolute or relative error of 10^{-5} of the correct answer will be accepted.

Limits

$0 \le \mathbf{s}_i \le 100$.
$\mathbf{s}_i > 0$ for some i. This means at least one contestant will have a point value greater than 0.

Small dataset

$1 \le \mathbf{T} \le 20$.
$2 \le \mathbf{N} \le 10$.

Large dataset

$1 \le \mathbf{T} \le 50$.
$2 \le \mathbf{N} \le 200$.

Sample

```
Input              Output

4                  Case #1: 33.333333 66.666667
2 20 10            Case #2: 0.000000 100.000000
2 10 0             Case #3: 25.0 25.0 25.0 25.0
4 25 25 25 25      Case #4: 34.666667 26.666667 38.666667
3 24 30 21
```

Problem B. Tide Goes In, Tide Goes Out

Problem

You are kayaking through a system of underground caves and suddenly realize that the tide is coming in and you are trapped! Luckily, you have a map of the cave system. You are stuck until the tide starts going out, so you will be here for a while. In the meantime, you want to determine the fastest way to the exit once the tide starts going out.

The cave system is an **N** by **M** grid. Your map consists of two **N** by **M** grids of numbers: one that specifies the height of the ceiling in each grid square, and one that specifies the height of the floor in each grid square. The floor of the cave system is porous, which means that as the water level falls, no water will remain above the water level.

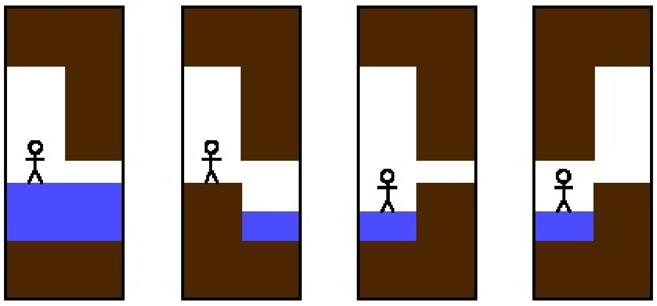
You are trapped at the north-west corner of the map. The current water level is **H** centimeters, and once it starts going down, it will drop at a constant rate of 10 centimeters per second, down to zero. The exit is at the south-east corner of the map. It is now covered by water, but it will become passable as soon as the water starts going down.

At any time, you can move north, south, east or west to an adjacent square with the following constraints:

- The water level, the floor height of your current square, and the floor height of the adjacent square must all be at least 50 centimeters lower than the ceiling height of the adjacent square. Note: this means that you will never be able to enter a square with less than 50 centimeters between the floor and the ceiling.
- The floor height of the adjacent square must be at least 50 centimeters below the ceiling height of your current square as well.
- You can never move off the edge of the map.

Note that you can go up or down as much as you want with your kayak. (You're very athletic from all this kayaking!) For example, you can go from a square with floor at height 10 centimeters to an adjacent square with floor at height 9000 centimeters (assuming the constraints given above are met).

These constraints are illustrated below:



- In the first image, you can't move to the right because the water level is less than 50 centimeters below the ceiling height of the adjacent square.
- In the second image, you can't move to the right because the floor height of your current square is less than 50 centimeters below the ceiling height of the adjacent square.
- In the third image, you can't move to the right because the floor height of the adjacent square is less than 50 centimeters below the ceiling height of the adjacent square. You'll never be able to enter that square from any direction.
- In the fourth image, you can't move to the right because the floor height of the adjacent square is less than 50 centimeters below the ceiling height of the current square.

When moving from one square to another, if there are at least 20 centimeters of water remaining on the current square when you start moving from it, it takes 1 second to complete the move (you can use your kayak). Otherwise, it takes 10 seconds (you have to drag your kayak). Note that the time depends only on the water level in the square you are leaving, not in the square you are entering.

It will be a while before the tide starts going out, and so you can spend as much time moving as you want before the water starts going down. What matters is how much time you will need from the moment the water starts going down until the moment you reach the exit. Can you calculate this time?

Input

- The first line will contain a single integer, **T**: the number of test cases
- It is followed by **T** test cases, each starting with a line containing integers **H**, **N** and **M**, representing the initial water level height, in centimeters, and the map dimensions. The following 2**N** lines contain the ceiling and floor heights as follows:
  - The next **N** lines contain **M** space-separated integers each. The *j*th integer in the *i*th row represents $\mathbf{C_{ij}}$, the height of the ceiling in centimeters at grid location *(j, i)*, where increasing *i* coordinates go South, and increasing *j* coordinates go East.
  - The next **N** lines contain **M** space-separated integers representing the heights of the floor, in the same format.
- At the starting location, there will always be at least 50 cm of air between the ceiling and the starting water level, and at least 50 cm between the ceiling and the floor.
- The exit location will always have at least 50 cm of air between the ceiling and the floor.
- There will always be a way out (you got in, after all!).

Output

For each test case, output one line containing "Case #x: t", where x is the case number (starting from 1), and t is the time, in seconds, starting from when the tide begins going out, that it takes you to make your way out of the cave system. Answers within an absolute or relative error of 10^{-6} of the correct answer will be accepted.

Notes

It is possible that you can go through the whole cave system before the tide starts dropping. In this case you will be able to wait at the exit for the tide to start dropping, so the answer in this case should be zero (this is the case in the fourth of the sample test cases).

Limits

Small dataset

$1 \leq \mathbf{T} \leq 50$.
$1 \leq \mathbf{N}, \mathbf{M} \leq 10$.
$1 \leq \mathbf{H} \leq 1000$.
$1 \leq \mathbf{F_{xy}} \leq \mathbf{C_{xy}} \leq 1000$.

Large dataset

$1 \leq \mathbf{T} \leq 50$.
$1 \leq \mathbf{N}, \mathbf{M} \leq 100$.
$1 \leq \mathbf{H} \leq 10000$.
$1 \leq \mathbf{F_{xy}} \leq \mathbf{C_{xy}} \leq 10000$.

Sample

Input	Output

```
4
200 1 2
250 233
180 100
100 3 3
500 500 500
500 500 600
500 140 1000
10 10 10
10 10 490
10 10 10
100 3 3
500 100 500
100 100 500
500 500 500
10 10 10
10 10 10
10 10 10
100 2 2
1000 1000
1000 1000
100 900
900 100
```

```
Case #1: 11.7
Case #2: 3.0
Case #3: 18.0
Case #4: 0.0
```

In the first sample test case, there are initially only 33 centimeters between the water level and the ceiling of the eastern square, so after the tide starts going down, you have to wait for at 1.7 seconds to enter it. Once it is accessible, you can start going in - but the water level in the western square is now so low (only 3 centimeters above the floor) that you have to drag your kayak for the next 10 seconds to get to the exit point.

The initial situation in the second case is better - you have a lot of headroom in adjacent squares, so you can move, for example, to (1, 1) before the tide starts dropping. Once there, you have to wait for the tide to start going down, and the water level to go down to 90cm (that takes one second). Then you can kayak south and then east and get out (in a total of three seconds). Note that you cannot go through the cave at (2, 1), even though the ceiling there is high enough, because there is too little space between the floor of this cave and the ceiling of any caves you could try to enter from ((1, 1) and (2, 0)) - only 10 centimeters in each case.

The third case is somewhat similar to the first - you have to wait at the starting position until the tide goes down to 50cm. After that you can kayak for the exit - but after three moves (taking three seconds) the water is at 20cm, which is only 10cm above the floor, which means the fourth move will be dragging instead of kayaking.

In the fourth case you are really lucky! You can immediately go the exit, even before the tide starts leaving, and wait there.

Submissions

Safety in Numbers

10pt	Not attempted **2687/5608 users** correct (48%)
11pt	Not attempted **2008/2680 users** correct (75%)

Tide Goes In, Tide Goes Out

18pt	Not attempted **682/892 users** correct (76%)
18pt	Not attempted **619/670 users** correct (92%)

Equal Sums

6pt	Not attempted **2257/2531 users** correct (89%)
37pt	Not attempted **149/853 users** correct (17%)

Top Scores

Gennady.Korotkevich	100
bmerry	100
hansonw	100
marcina	100
ZhukovDmitry	100
random.johnnyh	100
yeputons	100
rng..58	100
pashka	100
mikhailOK	100

Problem C. Equal Sums

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 6 points	Solve C-small
Large input 37 points	Solve C-large

Problem

I have a set of positive integers **S**. Can you find two non-empty, distinct subsets with the same sum?

Note: A subset is a set that contains only elements from **S**, and two subsets are distinct if they do not have exactly the same elements.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow, one per line. Each test case begins with **N**, the number of positive integers in **S**. It is followed by **N** distinct positive integers, all on the same line.

Output

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

- If there are two different subsets of **S** that have the same sum, then output these subsets, one per line. Each line should contain the numbers in one subset, separated by spaces.
- If it is impossible, then you should output the string "Impossible" on a single line.

If there are multiple ways of choosing two subsets with the same sum, any choice is acceptable.

Limits

No two numbers in **S** will be equal.
$1 \leq \mathbf{T} \leq 10$.

Small dataset

N is *exactly* equal to 20.
Each number in **S** will be a positive integer less than 10^5.

Large dataset

N is *exactly* equal to 500.
Each number in **S** will be a positive integer less than 10^{12}.

Sample

```
Input
2
20 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
20 120 266 858 1243 1657 1771 2328 2490 2665 2894 3117
4210 4454 4943 5690 6170 7048 7125 9512 9600
```

```
Output
Case #1:
1 2
3
Case #2:
3117 4210 4943
2328 2894 7048
```

Submissions

Diamond Inheritance

14pt	Not attempted **3062/4215 users** correct (73%)
14pt	Not attempted **2374/3030 users** correct (78%)

Out of Gas

10pt	Not attempted **467/762 users** correct (61%)
27pt	Not attempted **73/250 users** correct (29%)

Box Factory

12pt	Not attempted **1064/1800 users** correct (59%)
23pt	Not attempted **308/786 users** correct (39%)

Top Scores

mystic	100
sourspinach	100
meret	100
FloppyCat	100
Yao	100
AS1	100
fuseidenamida	100
Tan909090909090	100
AdrianRoos	100
MaxBuzz	100

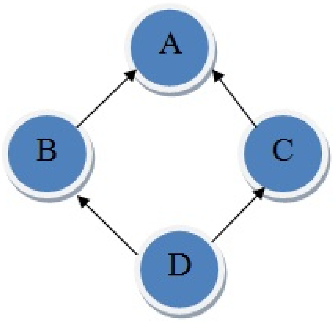
Problem A. **Diamond Inheritance**

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 14 points	Solve A-small
Large input 14 points	Solve A-large

Problem

You are asked to help diagnose class diagrams to identify instances of diamond inheritance. The following example class diagram illustrates the property of diamond inheritance. There are four classes: A, B, C and D. An arrow pointing from X to Y indicates that class X inherits from class Y.



In this class diagram, D inherits from both B and C, B inherits from A, and C also inherits from A. An inheritance path from X to Y is defined as a sequence of classes X, C_1, C_2, C_3, ..., C_n, Y where X inherits from C_1, C_i inherits from C_{i+1} for 1 ≤ i ≤ n - 1, and C_n inherits from Y. There are two inheritance paths from D to A in the example above. The first path is D, B, A and the second path is D, C, A.

A class diagram is said to contain a diamond inheritance if there exists a pair of classes X and Y such that there are at least two different inheritance paths from X to Y. The above class diagram is a classic example of diamond inheritance. Your task is to determine whether or not a given class diagram contains a diamond inheritance.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow, each specifies a class diagram. The first line of each test case gives the number of classes in this diagram, **N**. The classes are numbered from 1 to **N**. **N** lines follow. The i^th line starts with a non-negative integer $\mathbf{M_i}$ indicating the number of classes that class *i* inherits from. This is followed by $\mathbf{M_i}$ distinct positive integers each from 1 to **N** representing those classes. You may assume that:

- If there is an inheritance path from X to Y then there is no inheritance path from Y to X.
- A class will never inherit from itself.

Output

For each diagram, output one line containing "Case #x: y", where x is the case number (starting from 1) and y is "Yes" if the class diagram contains a diamond inheritance, "No" otherwise.

Limits

1 ≤ **T** ≤ 50.
0 ≤ $\mathbf{M_i}$ ≤ 10.

Small dataset

1 ≤ **N** ≤ 50.

Large dataset

1 ≤ **N** ≤ 1,000.

Sample

```
Input      Output

3          Case #1: No
3          Case #2: Yes
1 2        Case #3: Yes
1 3
0
5
2 2 3
1 4
1 5
1 5
0
3
2 2 3
1 3
0
```

Submissions

Diamond Inheritance

14pt	Not attempted **3062/4215 users** correct (73%)
14pt	Not attempted **2374/3030 users** correct (78%)

Out of Gas

10pt	Not attempted **467/762 users** correct (61%)
27pt	Not attempted **73/250 users** correct (29%)

Box Factory

12pt	Not attempted **1064/1800 users** correct (59%)
23pt	Not attempted **308/786 users** correct (39%)

Top Scores

mystic	100
sourspinach	100
meret	100
FloppyCat	100
Yao	100
AS1	100
fuseidenamida	100
Tan909090909090	100
AdrianRoos	100
MaxBuzz	100

Problem B. **Out of Gas**

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 10 points	Solve B-small
Large input 27 points	Solve B-large

Problem

Your car is out of gas, and you want to get home as quickly as possible! Fortunately, your home is at the bottom of a hill and you (in your car) are at the top of it. Unfortunately, there is a car in front of you, and you can't move past it. Fortunately, your brakes are working and they are *very* powerful.

You *start* at the top of the hill with speed 0 m/s at time 0 seconds. Gravity is pulling your car down the hill with a constant acceleration. At any time, you can use your brakes to reduce your speed, or temporarily reduce your acceleration, by any amount.

How quickly can you reach your home if you use your brakes in the best possible way?

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. The first line of each test case contains three space-separated numbers: a real-valued number **D**, the distance in meters to your home down the hill; and two integers, **N** and **A**. The distance **D** will be given in *exactly 6 decimal places*.

N lines follow, each of which contains two space-separated, real-valued numbers: a time $\mathbf{t_i}$ in seconds, and a position $\mathbf{x_i}$ in meters. The $\mathbf{t_i}$ and $\mathbf{x_i}$ values will be given in *exactly 6 decimal places*.

One line follows, with **A** space-separated, real-valued numbers $\mathbf{a_i}$, which are accelerations in `m/s`^{2}. The accelerations will be given in *exactly 2 decimal places*.

The other car's position is specified by the ($\mathbf{t_i}$, $\mathbf{x_i}$) pairs. The car's position at time $\mathbf{t_i}$ seconds is $\mathbf{x_i}$ meters measured from the top of the hill (i.e. your initial position). The car travels at constant speed between time $\mathbf{t_i}$ and $\mathbf{t_{i+1}}$. The positions and times will both be given in increasing order, with $\mathbf{t_0}$=0.

For example, if t_5=10, x_5=20, t_6=20, x_6=40, then 10 seconds after the *start*, the other car is 20 meters down the hill; 15 seconds after the *start*, the other car is 30 meters down the hill; and 20 seconds after the *start*, the other car is 40 meters down the hill.

Output

For each test case, output one line containing "Case #c:", where c is the case number (starting from 1). Then output **A** lines, the i^th^ of which contains the minimum number of seconds it takes you to reach your home if your acceleration down the hill due to gravity is $\mathbf{a_i}$, and you use your brakes in the best possible way. Answers within an absolute or relative error of 10^{-6} of the correct answer will be accepted. There should be no blank lines in the output.

Notes

Position and Acceleration: An object with a constant acceleration a `m/s`^{2} and starting speed of v_0 `m/s` will move a distance of v_0*t + 0.5*a*t^2 after t seconds.

Distance on the slope: All the distances and accelerations are given with respect to the straight line down the hill. They are **not**, for example, horizontal distances; so if your car is accelerating at 2 `m/s`^{2} with an initial speed of `0` `m/s`, and the other car is stopped at x=1, it will take exactly 1 second to reach the other car.

The other car: You may never pass the other car, which means that at no time shall your distance down the hill be greater than that of the other car. It may be equal. The cars should be considered as point masses.

Output values: You can print as many decimal places as you like in the output. We will read and compare your answers with ours, and at that time we will be using 10^{-6} as a threshold for inaccuracy. So 25, 25.0 and 25.000000 are the same from our perspective. Trailing zeros after the decimal point does not matter.

Limits

$1 \le \mathbf{T} \le 20$.
$1.0 \le \mathbf{D} \le 10^4$.
$1.0 \le \mathbf{a}_i \le 9.81$.
$0.0 \le \mathbf{t}_i \le 10^5$.
$0.0 \le \mathbf{x}_i \le 10^5$.
$\mathbf{t}_i < \mathbf{t}_{i+1}$.
$\mathbf{x}_i < \mathbf{x}_{i+1}$.
$\mathbf{t}_0 = 0$
$\mathbf{x}_{N-1} \ge \mathbf{D}$.

Small dataset

$1 \le \mathbf{N} \le 2$.
$1 \le \mathbf{A} \le 10$.

Large dataset

$1 \le \mathbf{N} \le 2000$.
$1 \le \mathbf{A} \le 250$.

Sample

Input

```
3
1000.000000 2 3
0.000000 20.500000
25.000000 1000.000000
1.00 5.00 9.81
50.000000 2 2
0.000000 0.000000
100000.000000 100.000000
1.00 1.01
10000.000000 3 1
0.000000 0.000000
10000.000000 0.100000
10000.100000 100000.000000
1.00
```

Output

```
Case #1:
44.7213595
25.000000
25
Case #2:
50000
50000
Case #3:
10140.974143
```

Top Scores

mystic	100
sourspinach	100
meret	100
FloppyCat	100
Yao	100
AS1	100
fuseidenamida	100
Tan909090909090	100
AdrianRoos	100
MaxBuzz	100

Problem C. **Box Factory**

Small input 12 points	Solve C-small
Large input 23 points	Solve C-large

Problem

You own a factory with two assembly lines. The first assembly line makes boxes, and the second assembly line makes toys to put in those boxes. Each type of box goes with one type of toy and vice-versa.

At the beginning, you pick up a box from the first assembly line and a toy from the second assembly line. You then have a few options.

- You can always throw out the box and pick up the next one.
- You can always throw out the toy and pick up the next one.
- If the box and toy are the same type, you can put the toy in the box, and send it out to customers.

You always pick boxes up in the order in which they are made, and similarly for toys. You know the order in which boxes and toys are made, and you want to plan out a strategy that will allow you to send as many boxed toys as possible to customers.

Warning: The two assembly lines make a *lot* of boxes and toys. However, they tend to make one kind of thing for a long period of time before switching.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow.

Each test case begins with a line contains two integers **N** and **M**. It is followed by a line containing 2 * **N** integers $\mathbf{a_1}$, $\mathbf{A_1}$, $\mathbf{a_2}$, $\mathbf{A_2}$, ..., $\mathbf{a_N}$, $\mathbf{A_N}$, and another line containing 2 * **M** integers $\mathbf{b_1}$, $\mathbf{B_1}$, $\mathbf{b_2}$, $\mathbf{B_2}$, ..., $\mathbf{b_M}$, $\mathbf{B_M}$.

This means that the first assembly line will make $\mathbf{a_1}$ boxes of type $\mathbf{A_1}$, then $\mathbf{a_2}$ boxes of type $\mathbf{A_2}$, etc., until it finishes with $\mathbf{a_N}$ boxes of type $\mathbf{A_N}$. Similarly, the second assembly will make $\mathbf{b_1}$ toys of type $\mathbf{B_1}$, followed by $\mathbf{b_2}$ toys of type $\mathbf{B_2}$, etc., until it finishes with $\mathbf{b_M}$ toys of type $\mathbf{B_M}$.

A toy can be matched with a box if and only if they have the same type number.

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 largest number of boxed toys that you can send out to customers.

Limits

$1 \le \mathbf{T} \le 100$.
$1 \le \mathbf{a_i}, \mathbf{b_i} \le 10^{16}$.
$1 \le \mathbf{A_i}, \mathbf{B_i} \le 100$.

Small dataset

$1 \le \mathbf{N} \le 3$.
$1 \le \mathbf{M} \le 100$.

Large dataset

$1 \le \mathbf{N}, \mathbf{M} \le 100$.

Sample

Input	Output

4
3 3
10 1 20 2 25 3
10 2 30 3 20 1
3 5
 	
10 1 6 2 10 1
5 1 3 2 10 1 3 2 5 1
	Case #1: 35
	Case #2: 20
	Case #3: 21
	Case #4: 0

```
3 5
10 1 6 2 10 1
5 1 6 2 10 1 6 2 5 1
1 1
5000000 10
5000000 100
```

Submissions

Swinging Wild

5pt	Not attempted **2006/2307 users** correct (87%)
9pt	Not attempted **1587/1995 users** correct (80%)

Aerobics

6pt	Not attempted **1124/1509 users** correct (74%)
15pt	Not attempted **741/1067 users** correct (69%)

Mountain View

13pt	Not attempted **435/888 users** correct (49%)
14pt	Not attempted **196/375 users** correct (52%)

Descending in the Dark

8pt	Not attempted **106/170 users** correct (62%)
30pt	Not attempted **0/79 users** correct (0%)

Top Scores

hos.lyric	70
LayCurse	70
eatmore	70
Gennady.Korotkevich	70
ACRushTC	70
mikhailOK	70
dolphinigle	70
Chmel.Tolstiy	70
EgorKulikov	70
Eryx	70

Problem A. Swinging Wild

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 5 points	Solve A-small
Large input 9 points	Solve A-large

Problem

You are standing on a ledge in the jungle, and your one true love is standing on a similar ledge at the other side of a swamp infested with snakes, crocodiles and a variety of other unpleasant denizens. Fortunately, there is a number of vines hanging from the canopy of the jungle over the swamp, even more fortunately, you somehow managed to get hold of the first of these vines (see figures below). The canopy of the jungle is at a constant height, and both the ledges are at the same height as the canopy. The vines are simply lines hanging from the canopy at certain points, with differing lengths.

If you happened to be a fictional hero, you would just go swinging wildly and yelling, at some point let go of the vine you hold, fly in the air for some time, catch another vine, swing again, and after a few repetitions you would be holding your one true love in your arms. Unfortunately, you are not a fictional hero, and if you tried that, probably yelling would be the only part you would manage well.

Your plan is a bit more cautious. You will swing on the vine you hold, but instead of letting go, you will catch hold of another vine. Then you will slowly and carefully climb up your original vine, so that the new vine you are holding will become horizontal - either to its full length, or up to the distance between the two vines, whichever is smaller. Then you will rest for a bit, and swing again, to repeat the process. Note that you do not have to catch the first vine you come up against while swinging, you might prefer to swing a bit further and catch some further-off vine instead. You can also climb up the vine you're currently swinging back and forth on to reduce the distance between you and the root of the vine. In effect, this means that you can catch any vine that your vine crosses while swinging. Note that you will not climb down a vine while swinging.

One other thing that sets you apart from any fictional hero is that before you start the whole rather risky procedure you would like to know whether it is actually possible to reach the other side of the jungle this way. And this is the question you have to answer in this problem.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. The first line of each test case contains the number **N** of vines. **N** lines describing the vines follow, each with a pair of integers $\mathbf{d_i}$ and $\mathbf{l_i}$ - the distance of the vine from your ledge, and the length of the vine, respectively. The last line of the test case contains the distance **D** to the ledge with your one true love. You start by holding the first vine in hand.

Output

For each test case, output one line containing "Case #x: y", where x is the case number (starting from 1) and y is a YES or a NO. Indicating whether it is possible for you to reach your one true love, given the rules above.

Limits

$0 < \mathbf{d_i}, \mathbf{l_i}, \mathbf{D} \leq 10^9$.
$\mathbf{T} \leq 30$.
$\mathbf{d_i} < \mathbf{d_{i+1}}$.
As you hold the first vine, $\mathbf{d_0} \leq \mathbf{l_0}$.
$\mathbf{d_{N-1}} < \mathbf{D}$.

Small dataset

$1 \leq \mathbf{N} \leq 100$.

Large dataset

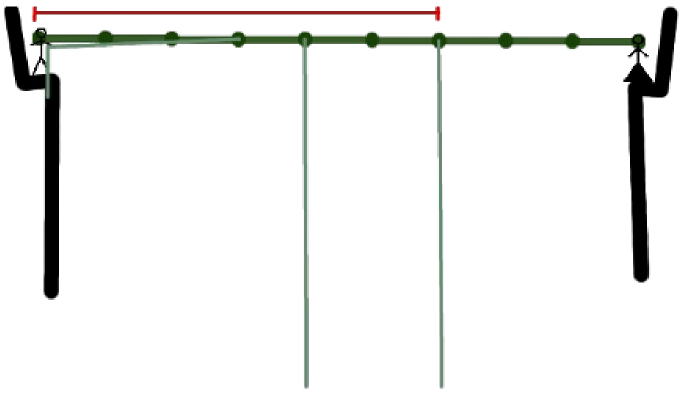
$1 \leq \mathbf{N} \leq 10000$.
There will be at most 60000 vines in all the test cases in total.

Sample

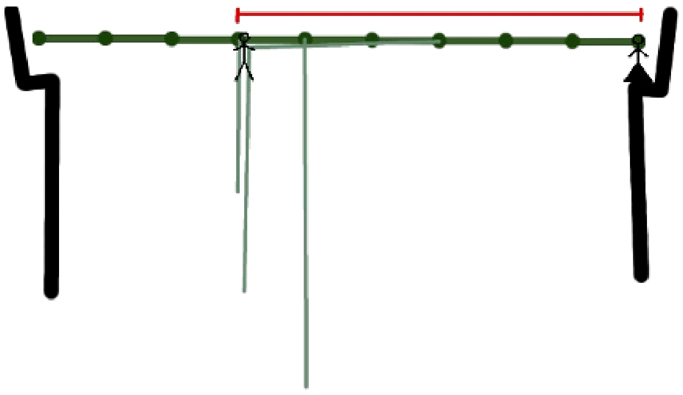
Input Output

```
4       Case #1: YES
3       Case #2: NO
3 4     Case #3: YES
4 10    Case #4: NO
6 10
9
3
3 4
4 10
7 10
9
2
6 6
10 3
13
2
6 6
10 3
14
```

In the first case, you hold the first vine 3 units away from where it is attached. You swing wildly, bypass the second vine and just barely catch the third. The picture below depicts the starting situation, and you are able to reach any vine that is rooted anywhere within the red interval:



After resting, you climb down the third one and up the first, to find yourself three units from the start, touching the canopy and holding the first and third vines. Now you let go of the first vine, swing again and again just barely reach the ledge, where your one true love awaits. The picture below depicts the situation after you caught the third vine and climbed over to the root of the first one. Again, you could reach any vine rooted within the red interval:



In the second case, you will not reach the third vine in the first swing, so your only choice is to catch the second. However, as it is attached four units from the start, you can (by going up the first vine) give yourself only one unit of swing - clearly too little to reach the third vine. Thus, you can't even reach the third vine, not to mention the other side of the swamp. Better go looking for some way around (or for a new true love).

In the third case, note that if you just swing on the first vine you hold, your path will not intersect the second vine - you have to climb up a bit while swinging (fortunately, you can) to reach the second vine. Remember, you can only climb up while swinging, you cannot climb down (because the vine going up is taut and you can put your weight on it, while the vine going down is swinging freely). In the fourth case, even though you can reach the second vine, it is too short to reach the final ledge.

Submissions

Swinging Wild

5pt Not attempted **2006/2307 users** correct (87%)
9pt Not attempted **1587/1995 users** correct (80%)

Aerobics

6pt Not attempted **1124/1509 users** correct (74%)
15pt Not attempted **741/1067 users** correct (69%)

Mountain View

13pt Not attempted **435/888 users** correct (49%)
14pt Not attempted **196/375 users** correct (52%)

Descending in the Dark

8pt Not attempted **106/170 users** correct (62%)
30pt Not attempted **0/79 users** correct (0%)

Top Scores

hos.lyric	70
LayCurse	70
eatmore	70
Gennady.Korotkevich	70
ACRushTC	70
mikhailOK	70
dolphinigle	70
Chmel.Tolstiy	70
EgorKulikov	70
Eryx	70

Problem B. Aerobics

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 6 points — Solve B-small

Large input 15 points — Solve B-large

Problem

The aerobics class begins. The trainer says, "Please position yourselves on the training mat so that each one of you has enough space to move your arms around freely, and not hit anybody else." People start milling around on the mat, trying to position themselves properly. Minutes pass, and finally the trainer is so annoyed that he asks you to write a program that will position all the people correctly, hoping it will be quicker than letting them figure it out for themselves!

You are given the dimensions (width and length) of the mat on which the class takes place. For every student, there is a circular area she has to have for herself, with radius equal to the reach of her arms. These circles can not intersect, though they can touch; and the center of each circle (where the student stands) has to be on the mat. Note that the arms **can** reach outside the mat. You know that there's plenty of space on the mat — the area of the mat is at least five times larger than the total area of the circles required by all the people in the class. It will always be possible for all the people to position themselves as required.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each test case consists of two lines. The first line contains three integers: **N**, **W** and **L**, denoting the number of students, the width of the mat, and the length of the mat, respectively. The second line contains **N** integers $\mathbf{r}_i$, denoting the reach of the arms of the ith student.

Output

For each test case, output one line containing "Case #n: y", where n is the case number (starting from 1) and y is a string containing 2**N** numbers, each of which can be an integer or a real number: $\mathbf{x}_1$, $\mathbf{y}_1$, $\mathbf{x}_2$, $\mathbf{y}_2$, etc., where the pair ($\mathbf{x}_i$, $\mathbf{y}_i$) is the position where the ith student should stand (with $0 \le \mathbf{x}_i \le \mathbf{W}$ and $0 \le \mathbf{y}_i \le L$).

As there will likely be multiple ways to position the students on the mat, you may output any correct positioning; but remember that you may not submit an output file more than 200kB in size.

Limits

$1 \le \mathbf{T} \le 50$.
$1 \le \mathbf{W}, \mathbf{L} \le 10^9$.
$1 \le \mathbf{r}_i \le 10^5$.
The area of the mat is at least 5 times larger than the total area of the circles:
$5*\pi*(\mathbf{r_1}^2 + ... + \mathbf{r_N}^2) \le \mathbf{W}*\mathbf{L}$.

Small dataset

$1 \le \mathbf{N} \le 10$.

Large dataset

$1 \le \mathbf{N} \le 10^3$.
The total number of circles in all test cases will be ≤ 6000.

Sample

```
Input          Output

2              Case #1: 0.0 0.0 6.0 6.0
2 6 6          Case #2: 0.0 0.0 7.0 0.0 12.0 0.0
1 1
3 320 2
4 3 2
```

Submissions

Swinging Wild

5pt	Not attempted **2006/2307 users** correct (87%)
9pt	Not attempted **1587/1995 users** correct (80%)

Aerobics

6pt	Not attempted **1124/1509 users** correct (74%)
15pt	Not attempted **741/1067 users** correct (69%)

Mountain View

13pt	Not attempted **435/888 users** correct (49%)
14pt	Not attempted **196/375 users** correct (52%)

Descending in the Dark

8pt	Not attempted **106/170 users** correct (62%)
30pt	Not attempted **0/79 users** correct (0%)

Top Scores

hos.lyric	70
LayCurse	70
eatmore	70
Gennady.Korotkevich	70
ACRushTC	70
mikhailOK	70
dolphinigle	70
Chmel.Tolstiy	70
EgorKulikov	70
Eryx	70

Problem C. **Mountain View**

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.

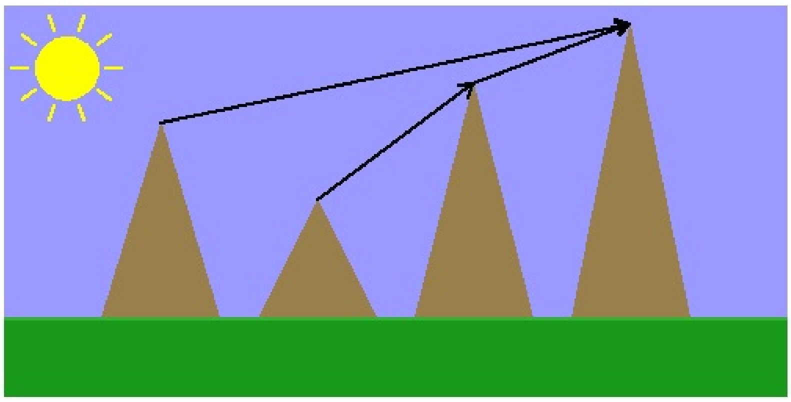
Small input 13 points	Solve C-small
Large input 14 points	Solve C-large

Problem

You are walking through the mountains. It turns out that in this mountain range there is a peak every kilometer, and there are no intermediate peaks. On every peak, you lie down for a rest, look forward, and perceive one of the peaks in front of you to be the highest one. The peak that looks like it's the highest might not **really** be the highest, for two reasons: there could be a higher peak that is obscured by another peak that's closer to you, and not as high; or you could be looking down, and a faraway peak could look higher than a nearby one.

To be precise, when we say that *peak **B** looks like it's the highest from peak **A*** we mean that **B** is further down the road than **A**; all peaks between **A** and **B** are below the line connecting the peaks **A** and **B**; and all the peaks that are further than **B** are below or on this line.

You don't know how high each peak is, but you have a very good memory; you've been on all the peaks; and you remember which peak looks like it's the highest from each of them. You would like to invent a set of heights for the peaks that is consistent with that information. Note that you were lying down when looking, so we assume you always looked from the ground level on each peak.


In this example, the fourth peak looks like it's the highest from the first and third peaks. When you're lying on the second peak, you can't see the fourth peak; the third one obscures it, and looks like it's the highest.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each test case consists of two lines. The first contains one number, **N**, the number of peaks in the range. You began your trip on peak 1 and went forward to peak **N**. The next line contains **N-1** numbers $\mathbf{x_i}$. The i-th number denotes the index of the peak that appeared to be the highest from peak *i* (note that peak **N** is the last peak, so there are no other peaks to see from there).

Output

For each test case, output one line containing "Case #n: y_1 y_2 ... $y_\mathbf{N}$", where n is the case number (starting from 1) and y_i is the height of the i-th peak. You can output any solution agreeing with the input data, except that all the heights you output have to be integers between 0 and 10^9, inclusive.

If no solution is possible, output "Case #n: Impossible" instead.

Limits

$1 \le \mathbf{T} \le 30$.
$i < \mathbf{x_i} \le \mathbf{N}$.

Small dataset

$2 \le \mathbf{N} \le 10$.

Large dataset

2 ≤ **N** ≤ 2000.

Sample

```
Input          Output

4              Case #1: 10 10 10 10 10 2
6              Case #2: 10 20 40 80
2 3 4 5 6      Case #3: Impossible
4              Case #4: 5 3 6 8
4 4 4
4
3 4 4
4
4 3 4
```

Submissions

Swinging Wild

5pt	Not attempted **2006/2307 users** correct (87%)
9pt	Not attempted **1587/1995 users** correct (80%)

Aerobics

6pt	Not attempted **1124/1509 users** correct (74%)
15pt	Not attempted **741/1067 users** correct (69%)

Mountain View

13pt	Not attempted **435/888 users** correct (49%)
14pt	Not attempted **196/375 users** correct (52%)

Descending in the Dark

8pt	Not attempted **106/170 users** correct (62%)
30pt	Not attempted **0/79 users** correct (0%)

Top Scores

hos.lyric	70
LayCurse	70
eatmore	70
Gennady.Korotkevich	70
ACRushTC	70
mikhailOK	70
dolphinigle	70
Chmel.Tolstiy	70
EgorKulikov	70
Eryx	70

Problem D. Descending in the Dark

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 8 points	Solve D-small
Large input 30 points	Solve D-large

Problem

You are on the face of Mount Everest. You need to find shelter before you freeze, and it's dark! What do you do?

The good news is you have already memorized the layout of the mountain. It is a grid with certain squares impassable and other squares containing caves where you can rest for the night. The bad news is you don't know where you are, and it's too steep to climb up. All you can do is move left, right, or down.

Here is an example layout, with '.' representing a passable square, '#' representing an impassable square, and numbers representing caves.

```
######
##...#
#..#.#
#...##
#0#..#
####1#
######
```

Since it is so dark, you will move around by following a *plan*, which is a series of instructions, each telling you to move one square left, right, or down. If an instruction would take you to a passable square or to a cave, you will follow it. If it would take you to an impassable square, you will have to ignore it. Either way, you will continue on to the next step, and so on, until you have gone through the whole plan.

To help with your descent, you want to find out two things for each cave **C**:

- What squares is it possible to reach **C** from? We will label the set of these squares by $\mathbf{S_C}$, and the number of them by $\mathbf{n_C}$.
- Is there a single plan that, if followed from any square in $\mathbf{S_C}$, will finish with you at cave **C**? If so, we say the cave is *lucky*.

Note that you might pass by several caves while following a plan. All that matters is what square you *finish* on after executing all the steps, not what caves you visit along the way.

For example, in the layout above, cave 0 is lucky. There are 9 squares that it can be reached from (including itself), and the plan "left-left-down-down-left-down" will finish with you at the cave from any of those squares.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow, beginning with a line containing integers **R** and **C**, representing the number of rows and columns in the mountain layout.

This is followed by **R** lines, each containing **C** characters, describing a mountain layout. As in the example above, a '#' character represents an impassable square, a '.' character represents a passable square, and the digits '0'-'9' represent caves (which are also passable squares).

Output

For each test case, first output one line containing "Case #x:", where x is the case number (starting from 1). For each cave **C**, starting with 0 and counting up from there, write a line "**C**: $\mathbf{n_C}$ $\mathbf{L_C}$". Here, **C** is the cave number, $\mathbf{n_C}$ is the number of squares you can reach the cave from, and $\mathbf{L_C}$ is either the string "Lucky" or the string "Unlucky", as defined above.

Limits

There will be between 1 and 10 caves inclusive.
If there are *d* caves, they will be labeled with the digits {0, 1, ..., *d* - 1}, and no two caves will have the same label.
All squares on the boundary of the mountain layout will be impassable.
$1 \le \mathbf{T} \le 20$.

Small dataset

3 ≤ **R, C** ≤ 10.

Large dataset

3 ≤ **R, C** ≤ 60.

Sample

```
Input      Output

2          Case #1:
7 5        0: 1 Lucky
#####      1: 3 Lucky
##0##      2: 4 Unlucky
##1.#      3: 7 Lucky
##2##      Case #2:
#3..#      0: 9 Lucky
#.#.#      1: 11 Unlucky
#####
7 6
######
##...#
#..#.#
#...##
#0#..#
####1#
######
```

In the first case, here are some valid plans you could use for the lucky caves:

- For cave 0, you can use the empty plan. If you can reach the cave at all, you are already in the right place!
- For cave 1, you can use the plan right-down-left.
- For cave 3, you can use the plan right-right-left-down-down-down-left.

Submissions

Perfect Game

3pt	Not attempted **395/397 users** correct (99%)
7pt	Not attempted **188/370 users** correct (51%)

Havannah

8pt	Not attempted **105/186 users** correct (56%)
12pt	Not attempted **58/87 users** correct (67%)

Quality Food

9pt	Not attempted **84/184 users** correct (46%)
18pt	Not attempted **34/60 users** correct (57%)

Lost Password

7pt	Not attempted **157/168 users** correct (93%)
36pt	Not attempted **1/3 users** correct (33%)

Top Scores

EgorKulikov	64
Eryx	64
SnapDragon	64
eatmore	64
bmerry	64
Ahyangyi	64
squark	64
andrewzta	64
Vasyl	57
misof	53

Problem A. Perfect Game

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 3 points	Solve A-small
Large input 7 points	Solve A-large

Problem

You're playing a video game, in which you will get an achievement if you complete all of the levels consecutively without dying. You can play the levels in any order, and each time you play a level you'll either complete it or die. Each level has some probability that you'll complete it, and takes some amount of time. In what order should you play the levels so that the expected time it takes you to get the achievement is minimized? Assume that it takes equally long to beat a level or to die in it, and that you will start again from the first level in your ordering as soon as you die.

Note: If you fail to complete a level, you do not personally die—only your character in the game dies. If that were not the case, only a few people would try to earn this achievement.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow, each of which consists of three lines. The first line of each test case contains a single integer **N**, the number of levels. The second line contains **N** space-separated integers $\mathbf{L_i}$. $\mathbf{L_i}$ is the number of seconds level `i` lasts, which is independent of whether you complete the level or die. The third line contains **N** space-separated integers $\mathbf{P_i}$. $\mathbf{P_i}$ is the percent chance that you will ***die*** in any given attempt to complete level `i`.

Output

For each test case, output one line containing "Case #x: ", where x is the case number (starting from 1), followed by **N** space-separated integers. The `j`th integer in the list should be the index of the `j`th level you should attempt to beat in order to minimize the amount of time you expect to spend earning the achievement.

Indices go from `0` to `N-1`. If there are multiple orderings that would give the same expected time, output the lexicographically least ordering. Out of two orderings, the lexicographically smaller one is the one with the smaller index at the first location where they differ; out of many orderings, the lexicographically least one is the one that is lexicographically smaller than every other ordering.

Limits

$1 \leq \mathbf{T} \leq 100$.
$0 \leq \mathbf{P_i} < 100$.

Small dataset

$1 \leq \mathbf{N} \leq 20$.
$\mathbf{L_i} = 1$.

Large dataset

$1 \leq \mathbf{N} \leq 1000$.
$1 \leq \mathbf{L_i} \leq 100$.

Sample

```
Input            Output

3                Case #1: 0 2 3 1
4                Case #2: 1 0 2
1 1 1 1          Case #3: 2 0 1
50 0 20 20
3
100 10 1
0 50 0
3
100 80 50
40 20 80
```

Note that the second and third samples do not satisfy the constraints for the small input.

Top Scores

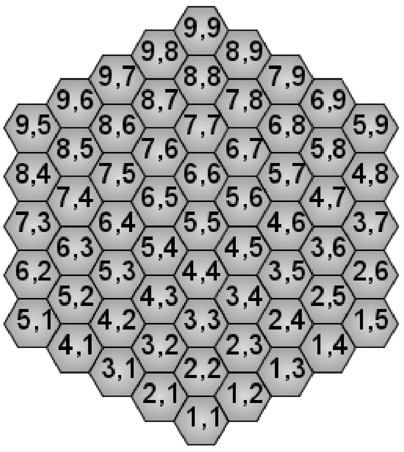
EgorKulikov	64
Eryx	64
SnapDragon	64
eatmore	64
bmerry	64
Ahyangyi	64
squark	64
andrewzta	64
Vasyl	57
misof	53

Problem B. **Havannah**

Small input 8 points	Solve B-small
Large input 12 points	Solve B-large

Problem

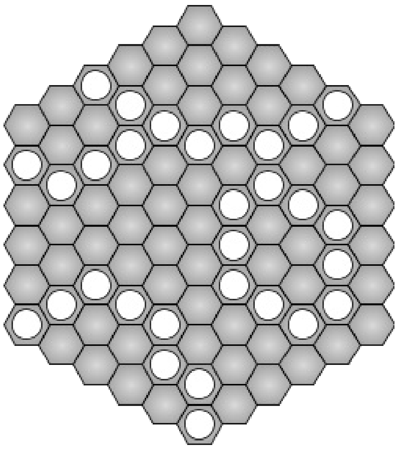
Havannah is an abstract strategy board game created by Christian Freeling. Havannah is a game played on a hexagonal board with **S** hexagons to each side. Each hexagon has two horizontal and four slanted edges. The hexagons are identified by pairs of integer values. The hexagon in the bottom corner of the board is (1, 1). The hexagon adjacent to (x, y) in the direction of a two-o'clock hand is (x, y+1). The hexagon adjacent to (x, y) in the direction of a ten-o'clock hand is (x + 1, y). Here is an example board with **S** = 5:



In the game of Havannah, each hexagon can be occupied by at most one stone. Stones once put on the board are never removed or moved. The goal of the game is to build from stones a *connected* set of stones of one of three kinds. The winning structures are:

- A **ring** that encircles one or more *empty* hexagons. That is, at least one of the inner hexagons must be empty. More specifically, there is an empty hexagon that is separated from the outermost boundary of the board by hexagons with stones. *Note that this rule is different from the official game Havannah.*
- A **bridge** that connects any two corners of the board.
- A **fork** that connects any three of the board's six edges. Corners do not count as part of either adjacent edge.

This picture shows examples of winning structures:



Your program should determine whether a sequence of moves of **a single player** builds a winning structure. If so, it should output the name of the structure and the number of the move that completed it. If a move completes multiple rings, connects more than two corners, or connects more than three edges, the structure is still considered a ring, a bridge, or a fork, respectively. But if a move completes structures of different kinds at once, your program should output the names of all of them. We are only interested in the first winning move: ignore all moves following the winning one. If there is no winning structure on the board after playing all the moves from the sequence, your program should output none.

Input

The first line of input gives the number of test cases, **T**. **T** test cases follow. The first line of each test case contains two integers **S** and **M**, the number of hexagons on each side of the board and the number of moves in the sequence, respectively. The next **M** lines provide the sequence of moves, in order, where each line contains a space-separated pair (x, y) of hexagon identifiers. All the moves in the sequence lie on the board of size **S**. In each test case, the board is initially empty and the moves do not repeat.

Output

For each test case, output one line containing "Case #**n**: " followed by one of:

- `none`
- `bridge in move` *k*
- `fork in move` *k*
- `ring in move` *k*
- `bridge-fork in move` *k*
- `bridge-ring in move` *k*
- `fork-ring in move` *k*
- `bridge-fork-ring in move` *k*

The cases are numbered starting from 1. The moves are numbered starting from 1.

Limits

Small dataset

1 ≤ **T** ≤ 200
2 ≤ **S** ≤ 50
0 ≤ **M** ≤ 100

Large dataset

1 ≤ **T** ≤ 20
2 ≤ **S** ≤ 3000
0 ≤ **M** ≤ 10000

Sample

Input

```
7
2 4
1 1
1 2
2 3
3 3
3 6
2 1
2 2
2 3
2 4
1 2
4 4
3 7
3 3
2 2
2 3
3 4
4 4
4 3
3 2
3 6
2 2
2 3
3 4
4 4
4 3
3 2
3 8
1 1
2 1
1 3
2 4
1 2
3 2
3 3
3 4
3 7
1 1

2 2
3 5
3 4
5 3
```

Output

```
Case #1: bridge in move 2
Case #2: fork in move 5
Case #3: none
Case #4: ring in move 6
Case #5: bridge-fork in move 5
Case #6: bridge in move 7
Case #7: none
```

```
4 3
3 3
3 3
1 1
1 3
3 5
```

Havannah was created by Christian Freeling and MindSports. MindSports and Christian Freeling do not endorse and have no involvement with Google Code Jam.

Top Scores

EgorKulikov	64
Eryx	64
SnapDragon	64
eatmore	64
bmerry	64
Ahyangyi	64
squark	64
andrewzta	64
Vasyl	57
misof	53

Problem C. **Quality Food**

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
9 points — Solve C-small

Large input
18 points — Solve C-large

Problem

You just moved from your hometown to a big metropolitan city! You love everything about your new environment, except for the food. Your hometown provides the best food in the region (called "quality food") and you sure will miss it.

Fortunately, the largest restaurant in your hometown provides a food delivery service. You can purchase any amount of food in one delivery. There is a constant *delivery fee* for every delivery, regardless of the amount of food purchased in the delivery.

This restaurant serves different types of food. Each type of food has two properties: a price-per-meal, and a time-to-stale. One "meal" of food will feed you for one day; once a meal has been eaten, it cannot be eaten again. The time-to-stale of a type of food is the maximum number of days for which that food can still be eaten, counting from when you received it. A time-to-stale of zero means you must eat that type of food on the day of delivery.

In a single delivery you can purchase as many different types of food, and as many meals of each of those types, as you have money for. Note that if a particular type of food has a time-to-stale of `t`, it doesn't make any sense to order more than `t+1` meals of that food in one delivery: at least one meal would go stale before you could eat it.

This restaurant has a very fast delivery service, so you will receive all the food in a delivery on the same day that you purchased it, and you may eat some of the food on the same day. Food delivery is the only way for you to receive quality food.

Given an amount of money, which you can spend on meal prices and delivery fees, what is the maximum number of days for which you can eat quality food every day?

Input

The first line of input gives the number of test cases, **T**. **T** test cases follow. Each test case will begin with three integers, **M**, **F** and **N**, denoting the amount of money you have, the delivery fee, and the number of types of food provided by the restaurant, respectively. **N** lines follow, each will consist of two integers, $\mathbf{P_i}$ and $\mathbf{S_i}$, denoting respectively the price-per-meal and time-to-stale of one type of food.

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 maximum number of days that you can keep eating at least one meal of quality food everyday.

Limits

$1 \le \mathbf{T} \le 50$.
$1 \le \mathbf{F} \le \mathbf{M}$.
$1 \le \mathbf{N} \le 200$.
$1 \le \mathbf{P_i} \le \mathbf{M}$.

Small dataset

$0 \le \mathbf{S_i} \le 2{,}000{,}000$.
$1 \le \mathbf{M} \le 2{,}000{,}000$.

Large dataset

$0 \le \mathbf{S_i} \le 10^{18}$.
$1 \le \mathbf{M} \le 10^{18}$.

Sample

```
Input        Output

3            Case #1: 3
```

```
32 5 2     Case #2: 0
5 0        Case #3: 8
10 2
10 10 1
10 10
10 1 1
1 5
```

An example scenario for the first case is by purchasing one meal of the first type and one meal of the second type during your first day in the city (costing a total of 20). Eat the first type of food that day, and eat the second type the next day. During your third day, purchase one meal of the first type and eat it on the same day. This accounts for three days.

Top Scores

EgorKulikov	64
Eryx	64
SnapDragon	64
eatmore	64
bmerry	64
Ahyangyi	64
squark	64
andrewzta	64
Vasyl	57
misof	53

Problem D. Lost Password

Small input 7 points — Solve D-small

Large input 36 points — Solve D-large

Problem

Ashish has forgotten his password. He remembers that he used the following algorithm to create his password: Ashish took up to **k** consecutive words from a passage of text, and took the first letter from each word. Then, he might have changed some of the letters to their "l33tspeak" equivalents. Specifically, he might have changed "o" to "0", "i" to "1", "e" to "3", "a" to "4", "s" to "5", "t" to "7", "b" to "8" and/or "g" to "9".

For example, if Ashish took his password from the first sentence of The Fellowship of the Ring -- *"This book is largely concerned with Hobbits, and from its pages a reader may discover much of their character and a little of their history"* -- Ashish would have reduced that to "tbilcwhafiparmdmotcaaloth". Then the password might be "tbilcwh", "7b1lcwh4f", "a", "4", or "4al07h", etc.

Ashish has a special extension installed in his browser that will prevent his computer from uploading any string that contains his password. In order to figure out which passage of text he took his password from, Ashish has created a webpage to take advantage of this extension. Every second, the webpage will tell the browser to post a "password string" for a new passage of text: a string that contains all of the possible passwords that Ashish could have chosen from that passage of text. As soon as his browser fails to post such a string, Ashish will know where he took his password from.

For example, if **k** = 2 and the passage of text contains words starting with the letters "google", then one password string for that passage is "goo0og00gle9o909l3". All substrings of length ≤ 2 from the original string, and all of their l33tspeak equivalents, are contained in the new string.

Given the first letters of the words in a passage of text, what is the minimum number of characters in the "password string" of that passage?

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each test case consists of two lines. The first line contains the integer **k**. The second line contains a string **S**, representing the first letters of the words in a passage of text. **S** contains only the characters 'a' - 'z', with no spaces.

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 minimum number of characters in the password string for **S**.

Limits

1 ≤ **T** ≤ 20.
S will contain at least 2 * **k** characters.
There will exist a password string with at most 10^{18} characters.

Small dataset

S will contain at most 1000 characters.
k = 2.

Large dataset

S will contain at most 5000 characters.
2 ≤ **k** ≤ 500.

Sample

```
Input                          Output

4                              Case #1: 6
2                              Case #2: 18
poppop                         Case #3: 53
2                              Case #4: 1136
google
2
tbilcwhafiparmdmotcaaloth
```

```
10
tbilcwhafiparmdmotcaaloth
```

In the first sample input, one possible password string is "0ppop0".
In the second sample input, one possible password string is "goo0og00gle9o909l3".

Submissions

Checkerboard Matrix

4pt	Not attempted **23/26 users** correct (88%)
9pt	Not attempted **23/23 users** correct (100%)

Power Swapper

4pt	Not attempted **25/25 users** correct (100%)
12pt	Not attempted **19/21 users** correct (90%)

Symmetric Trees

7pt	Not attempted **22/24 users** correct (92%)
18pt	Not attempted **15/22 users** correct (68%)

Paradox Sort

4pt	Not attempted **24/24 users** correct (100%)
28pt	Not attempted **11/15 users** correct (73%)

Allergy Testing

15pt	Not attempted **19/23 users** correct (83%)
35pt	Not attempted **1/6 users** correct (17%)

ARAM

22pt	Not attempted **3/5 users** correct (60%)
42pt	Not attempted **0/3 users** correct (0%)

Top Scores

Gennady.Korotkevich	136
eatmore	123
sevenkplus	101
mystic	95
mk.al13n	89
EgorKulikov	89
kcm1700	89
vepifanov	83
dzhulgakov	83
Romka	83

Problem A. Checkerboard Matrix

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 4 points	Solve A-small
Large input 9 points	Solve A-large

Problem

When she is bored, Mija sometimes likes to play a game with matrices. She tries to transform one matrix into another with the fewest moves. For Mija, one move is swapping any two rows of the matrix or any two columns of the matrix.

Today, Mija has a very special matrix **M**. **M** is a 2**N** by 2**N** matrix where every entry is either a 0 or a 1. Mija decides to try and transform **M** into a *checkerboard matrix* where the entries alternate between 0 and 1 along each row and column. Can you help Mija find the minimum number of moves to transform **M** into a *checkerboard matrix*?

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each test case starts with a line containing a single integer: **N**. The next 2**N** lines each contain 2**N** characters which are the rows of **M**; each character is a 0 or 1.

Output

For each test case, output one line containing "Case #x: y", where x is the test case number (starting from 1) and y is the minimum number of row swaps and column swaps required to turn **M** into a *checkerboard matrix*. If it is impossible to turn **M** into a checkerboard matrix, y should be "IMPOSSIBLE".

Limits

$1 \le \mathbf{T} \le 100$.

Small dataset

$1 \le \mathbf{N} \le 10$.

Large dataset

$1 \le \mathbf{N} \le 10^3$.

Sample

```
Input     Output

3         Case #1: 0
1         Case #2: 2
01        Case #3: IMPOSSIBLE
10
2
1001
0110
0110
1001
1
00
00
```

In the first sample case, **M** is already a *checkerboard matrix*.

In the second sample case, Mija can turn **M** into a *checkerboard matrix* by swapping columns 1 and 2 and then swapping rows 1 and 2.

In the third sample case, Mija can never turn **M** into a *checkerboard matrix*; it doesn't have enough 1s.

Submissions

Checkerboard Matrix

4pt	Not attempted **23/26 users** correct (88%)
9pt	Not attempted **23/23 users** correct (100%)

Power Swapper

4pt	Not attempted **25/25 users** correct (100%)
12pt	Not attempted **19/21 users** correct (90%)

Symmetric Trees

7pt	Not attempted **22/24 users** correct (92%)
18pt	Not attempted **15/22 users** correct (68%)

Paradox Sort

4pt	Not attempted **24/24 users** correct (100%)
28pt	Not attempted **11/15 users** correct (73%)

Allergy Testing

15pt	Not attempted **19/23 users** correct (83%)
35pt	Not attempted **1/6 users** correct (17%)

ARAM

22pt	Not attempted **3/5 users** correct (60%)
42pt	Not attempted **0/3 users** correct (0%)

Top Scores

Gennady.Korotkevich	136
eatmore	123
sevenkplus	101
mystic	95
mk.al13n	89
EgorKulikov	89
kcm1700	89
vepifanov	83
dzhulgakov	83
Romka	83

Problem B. **Power Swapper**

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 4 points	Solve B-small
Large input 12 points	Solve B-large

Problem

In a parallel universe, people are crazy about using numbers that are powers of two, and they have defined an exciting sorting strategy for permutations of the numbers from 1 to $2^{\mathbf{N}}$. They have defined a swapping operation in the following way:

- A range of numbers to be swapped is valid if and only if it is a range of adjacent numbers of size 2^k, and its starting position (position of the first element in the range) is a multiple of 2^k (where positions are 0-indexed).
- A valid swap operation of *size-k* is defined by swapping two distinct, valid ranges of numbers, each of size 2^k.

To sort the given permutation, you are allowed to use at most one swap operation of each size k, for k in [0, **N**). Also, note that swapping a range with itself is not allowed.

For example, given the permutation [3, 6, 1, 2, 7, 8, 5, 4] (a permutation of the numbers from 1 to 2^3), the permutation can be sorted as follows:

- [3, 6, 1, 2, 7, 8, 5, 4]: make a *size-2* swap of the ranges [3, 6, 1, 2] and [7, 8, 5, 4].
- [7, 8, 5, 4, 3, 6, 1, 2]: make a *size-0* swap of [5] and [3].
- [7, 8, 3, 4, 5, 6, 1, 2]: make a *size-1* swap of [7, 8] and [1, 2].
- [1, 2, 3, 4, 5, 6, 7, 8]: done.

The previous steps used every swap size (0, 1, and 2) at most once. Also, notice that all the swaps were valid because both ranges for each size k started at indices that were multiples of 2^k.

Count how many ways there are to sort the given permutation by using the rules above. A way is an ordered sequence of swaps, and two ways are the same only if the sequences are identical.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. The first line of each test case contains a single integer **N**. The following line contains $\mathbf{2^N}$ space-separated integers: a permutation of the numbers 1, 2, ..., $2^{\mathbf{N}}$.

Output

For each test case, output one line containing "Case #x: y", where x is the test case number (starting from 1) and y is the number of ways to sort the given permutation using the rules above.

Limits

$1 \le \mathbf{T} \le 200$.

Small dataset

$1 \le \mathbf{N} \le 4$.

Large dataset

$1 \le \mathbf{N} \le 12$.

Sample

```
Input                Output

4                    Case #1: 1
1                    Case #2: 3
2 1                  Case #3: 6
2                    Case #4: 0
1 4 3 2
3
7 8 5 6 1 2 4 3
```

```
2
4 3 2 1
```

Submissions

Checkerboard Matrix

4pt	Not attempted **23/26 users** correct (88%)
9pt	Not attempted **23/23 users** correct (100%)

Power Swapper

4pt	Not attempted **25/25 users** correct (100%)
12pt	Not attempted **19/21 users** correct (90%)

Symmetric Trees

7pt	Not attempted **22/24 users** correct (92%)
18pt	Not attempted **15/22 users** correct (68%)

Paradox Sort

4pt	Not attempted **24/24 users** correct (100%)
28pt	Not attempted **11/15 users** correct (73%)

Allergy Testing

15pt	Not attempted **19/23 users** correct (83%)
35pt	Not attempted **1/6 users** correct (17%)

ARAM

22pt	Not attempted **3/5 users** correct (60%)
42pt	Not attempted **0/3 users** correct (0%)

Top Scores

Gennady.Korotkevich	136
eatmore	123
sevenkplus	101
mystic	95
mk.al13n	89
EgorKulikov	89
kcm1700	89
vepifanov	83
dzhulgakov	83
Romka	83

Problem C. Symmetric Trees

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 7 points	Solve C-small
Large input 18 points	Solve C-large

Problem

Given a vertex-colored tree with N nodes, can it be drawn in a 2D plane with a line of symmetry?

Formally, a tree is *line-symmetric* if each vertex can be assigned a location in the 2D plane such that:

- All locations are distinct.
- If vertex $\mathbf{v_i}$ has color **C** and coordinates ($\mathbf{x_i}$, $\mathbf{y_i}$), there must also be a vertex $\mathbf{v_i}$' of color **C** located at (-$\mathbf{x_i}$, $\mathbf{y_i}$) -- Note if $\mathbf{x_i}$ is 0, $\mathbf{v_i}$ and $\mathbf{v_i}$' are the same vertex.
- For each edge ($\mathbf{v_i}$, $\mathbf{v_j}$), there must also exist an edge ($\mathbf{v_i}$', $\mathbf{v_j}$').
- If edges were represented by straight lines between their end vertices, no two edges would share any points except where adjacent edges touch at their endpoints.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow.

Each test case starts with a line containing a single integer **N**, the number of vertices in the tree.

N lines then follow, each containing a single uppercase letter. The i-th line represents the color of the i-th node.

N-1 lines then follow, each line containing two integers **i** and **j** (1 ≤ **i** < **j** ≤ **N**). This denotes that the tree has an edge from the **i**-th vertex to the **j**-th vertex. The edges will describe a connected tree.

Output

For each test case, output one line containing "Case #x: y", where x is the case number (starting from 1) and y is "SYMMETRIC" if the tree is line-symmetric by the definition above or "NOT SYMMETRIC" if it isn't.

Limits

1 ≤ **T** ≤ 100.

Small dataset

2 ≤ **N** ≤ 12.

Large dataset

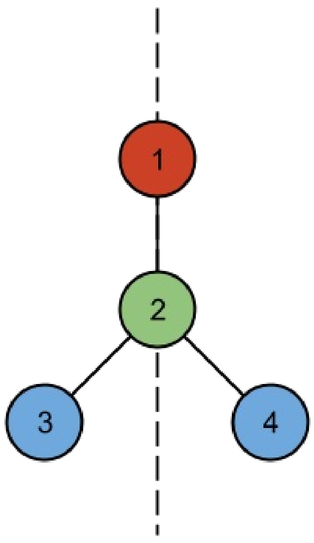
2 ≤ **N** ≤ 10000.

Sample

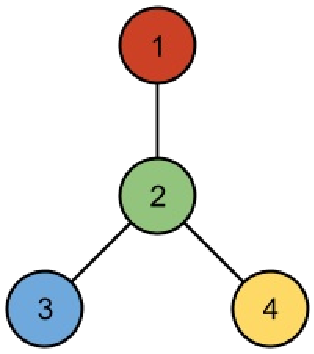
Input	Output
3 4 R G B B 1 2 2 3 2 4 4 R G B Y 1 2 2 3 2 4 12	Case #1: SYMMETRIC Case #2: NOT SYMMETRIC Case #3: SYMMETRIC

```
Y
B
Y
G
R
G
Y
Y
B
B
B
R
1 3
1 9
1 10
2 3
3 7
3 8
3 11
4 8
5 7
6 7
8 12
```

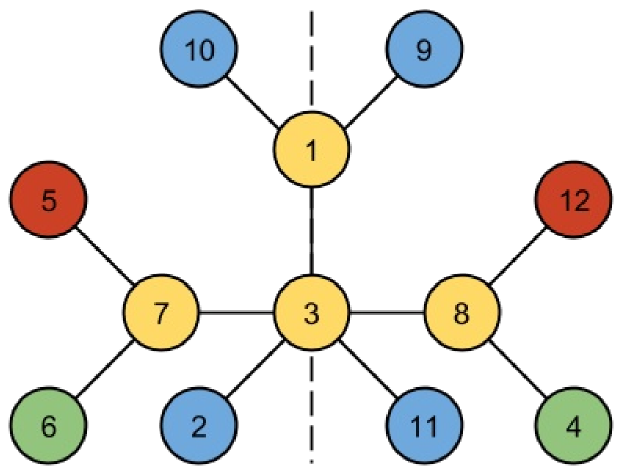
The first case can be drawn as follows:



No arrangement of the second case has a line of symmetry:



One way of drawing the third case with a symmetry line is as follows:


10
9
1
5
12
7
3
8
6
2
11
4

Submissions

Checkerboard Matrix

4pt	Not attempted **23/26 users** correct (88%)
9pt	Not attempted **23/23 users** correct (100%)

Power Swapper

4pt	Not attempted **25/25 users** correct (100%)
12pt	Not attempted **19/21 users** correct (90%)

Symmetric Trees

7pt	Not attempted **22/24 users** correct (92%)
18pt	Not attempted **15/22 users** correct (68%)

Paradox Sort

4pt	Not attempted **24/24 users** correct (100%)
28pt	Not attempted **11/15 users** correct (73%)

Allergy Testing

15pt	Not attempted **19/23 users** correct (83%)
35pt	Not attempted **1/6 users** correct (17%)

ARAM

22pt	Not attempted **3/5 users** correct (60%)
42pt	Not attempted **0/3 users** correct (0%)

Top Scores

Gennady.Korotkevich	136
eatmore	123
sevenkplus	101
mystic	95
mk.al13n	89
EgorKulikov	89
kcm1700	89
vepifanov	83
dzhulgakov	83
Romka	83

Problem D. Paradox Sort

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 4 points	Solve D-small
Large input 28 points	Solve D-large

Problem

Vlad likes candies. You have a bag of different candies, and you're going to let Vlad keep one of them. You choose an order for the candies, then give them to Vlad one at a time. For each candy Vlad receives (after the first one), he compares the candy he had to the one he was just given, keeps the one he likes more, and throws the other one away.

You would expect that for any order you choose, Vlad will always end up with his favorite candy. But this is not the case! He does not necessarily have a favorite candy. We know for any pair of candies which one he will prefer, but his choices do not necessarily correspond to a simple ranking. He may choose Orange when offered Orange and Lemon, Banana when offered Orange and Banana, and Lemon when offered Lemon and Banana!

There is a particular candy you want Vlad to end up with. Given Vlad's preferences for each pair of candies, determine if there is an ordering such that Vlad will end up with the right candy. If there is, find the lexicographically-smallest such ordering.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each test case will start with a line containing the integers **N** and **A**, separated by a space. **N** is the number of candies, and **A** is the number of the candy we want Vlad to finish with. The candies are numbered from 0 to **N**-1. The next **N** lines each contains **N** characters. Character j of line i will be 'Y' if Vlad prefers candy i to candy j, 'N' if Vlad prefers candy j to candy i, and '-' if i = j. Note that if i ≠ j, the jth character of the ith row must be different from the ith character of the jth row.

Output

For each test case output "Case #x: ", where x is the case number, followed by either "IMPOSSIBLE" or a space-separated list of the lexicographically-smallest ordering of candies that leaves Vlad with **A**.

Limits

1 ≤ **T** ≤ 100.

Small dataset

1 ≤ **N** ≤ 10.

Large dataset

1 ≤ **N** ≤ 100.

Sample

Input

```
3
2 0
-Y
N-
2 0
-N
Y-
4 3
-YNN
N-YY
YN-Y
YNN-
```

Output

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

Submissions

Checkerboard Matrix

4pt	Not attempted **23/26 users** correct (88%)
9pt	Not attempted **23/23 users** correct (100%)

Power Swapper

4pt	Not attempted **25/25 users** correct (100%)
12pt	Not attempted **19/21 users** correct (90%)

Symmetric Trees

7pt	Not attempted **22/24 users** correct (92%)
18pt	Not attempted **15/22 users** correct (68%)

Paradox Sort

4pt	Not attempted **24/24 users** correct (100%)
28pt	Not attempted **11/15 users** correct (73%)

Allergy Testing

15pt	Not attempted **19/23 users** correct (83%)
35pt	Not attempted **1/6 users** correct (17%)

ARAM

22pt	Not attempted **3/5 users** correct (60%)
42pt	Not attempted **0/3 users** correct (0%)

Top Scores

Gennady.Korotkevich	136
eatmore	123
sevenkplus	101
mystic	95
mk.al13n	89
EgorKulikov	89
kcm1700	89
vepifanov	83
dzhulgakov	83
Romka	83

Problem E. Allergy Testing

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 15 points	Solve E-small
Large input 35 points	Solve E-large

Problem

Kelly is allergic to exactly one of **N** foods, but she isn't sure which one. So she decides to undergo some experiments to find out.

In each experiment, Kelly picks several foods and eats them all. She waits **A** days to see if she gets any allergic reactions. If she doesn't, she knows she isn't allergic to any of the foods she ate. If she does get a reaction, she has to wait for it to go away: this takes a total of **B** days (measured from the moment when she ate the foods).

To simplify her experimentation, Kelly decides to wait until each experiment is finished (after **A** or **B** days) before starting the next one. At the start of each experiment, she can choose the set of foods she wants to eat based on the results of previous experiments.

Kelly chooses what foods to eat for each experiment to minimize the worst-case number of days before she knows which of the **N** foods she is allergic to. How long does it take her in the worst case?

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each test case on a single line, containing three space-separated integers: **N**, **A** and **B**.

Output

For each test case, output one line containing "Case #x: y", where x is the test case number (starting from 1) and **y** is the number of days it will take for Kelly to find out which food she is allergic to, in the worst case.

Limits

1 ≤ **T** ≤ 200.

Small dataset

$1 \le \mathbf{N} \le 10^{15}$.
$1 \le \mathbf{A} \le \mathbf{B} \le 100$.

Large dataset

$1 \le \mathbf{N} \le 10^{15}$.
$1 \le \mathbf{A} \le \mathbf{B} \le 10^{12}$.

Sample

```
Input        Output

3            Case #1: 12
4 5 7        Case #2: 3
8 1 1        Case #3: 0
1 23 32
```

In the first sample case:

- First, Kelly eats foods #1 and #2.
- If she gets no reaction after 5 days, she eats food #3. 5 days after *that*, she will know whether she is allergic to food #3 or food #4.
- If she does get a reaction to the first experiment, then 7 days after the first experiment, she eats food #1. 5 days after that, she will know whether she is allergic to food #1 or food #2.

Submissions

Checkerboard Matrix

4pt	Not attempted **23/26 users** correct (88%)
9pt	Not attempted **23/23 users** correct (100%)

Power Swapper

4pt	Not attempted **25/25 users** correct (100%)
12pt	Not attempted **19/21 users** correct (90%)

Symmetric Trees

7pt	Not attempted **22/24 users** correct (92%)
18pt	Not attempted **15/22 users** correct (68%)

Paradox Sort

4pt	Not attempted **24/24 users** correct (100%)
28pt	Not attempted **11/15 users** correct (73%)

Allergy Testing

15pt	Not attempted **19/23 users** correct (83%)
35pt	Not attempted **1/6 users** correct (17%)

ARAM

22pt	Not attempted **3/5 users** correct (60%)
42pt	Not attempted **0/3 users** correct (0%)

Top Scores

Gennady.Korotkevich	136
eatmore	123
sevenkplus	101
mystic	95
mk.al13n	89
EgorKulikov	89
kcm1700	89
vepifanov	83
dzhulgakov	83
Romka	83

Problem F. ARAM

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 22 points	Solve F-small
Large input 42 points	Solve F-large

Problem

In the game League of Legends™, you can play a type of game called "ARAM", which is short for "All Random, All Mid". This problem uses a similar idea, but doesn't require you to have played League of Legends to understand it.

Every time you start playing an ARAM game, you're assigned one of **N** "champions", uniformly at random. You're more likely to win with some champions than with others, so if you get unlucky then you might wish you'd been given a different champion. Luckily for you, the game includes a "Reroll" function.

Rerolling randomly reassigns you a champion in a way that will be described below; but you can't reroll whenever you want to. The ability to reroll works like a kind of money. Before you play your first ARAM game, you begin with **R** RD ("reroll dollars"). You can only reroll if you have at least 1 RD, and you must spend 1 RD to reroll. After every game, you gain 1/**G** RD (where **G** is an integer), but you can never have more than **R** RD: if you have **R** RD and then play a game, you'll still have **R** RD after that game.

If you have at least 1RD, and you choose to reroll, you will spend 1RD and be re-assigned one of the **N** champions, uniformly at random. There's some chance you might get the same champion you had at first. If you don't like the champion you rerolled, and you still have at least 1RD left, you can reroll again. As long as you have at least 1RD left, you can keep rerolling.

For example, if **R**=2 and **G**=2, and you use a reroll in your first game, then after your first game you will have 1.5 RD. If you play another game, this time without using a reroll, you will have 2.0 RD. If you play another game without using a reroll, you will still have 2.0 RD (because you can never have more than **R**=2). If you use two rerolls in your next game, then after that game you will have 0.5 RD.

You will be given the list of champions, and how likely you are to win a game if you play each of them. If you play 10^{100} games and choose your strategy optimally, what fraction of the games do you expect to win?

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each starts with a line containing three space-separated integers: **N**, **R** and **G**. The next line contains **N** space-separated, real-valued numbers $\mathbf{P}_i$, indicating the probability that you will win if you play champion `i`.

Output

For each test case, output one line containing "Case #x: y", where x is the test case number (starting from 1) and y is the proportion of games you will win if you play 10^{100} games.

y will be considered correct if it is within an absolute or relative error of 10^{-10} of the correct answer. See the FAQ for an explanation of what that means, and what formats of real numbers we accept.

Limits

$1 \le \mathbf{T} \le 100$.
$0.0 \le \mathbf{P}_i \le 1.0$.
$\mathbf{P}_i$ will be expressed as a single digit, followed by a decimal point, followed by 4 digits.

Small dataset

$1 \le \mathbf{N} \le 1000$.
$1 \le \mathbf{R} \le 2$.
$1 \le \mathbf{G} \le 3$.

Large dataset

$1 \le \mathbf{N} \le 1000$.
$1 \le \mathbf{R} \le 20$.
$1 \le \mathbf{G} \le 20$.

Sample

```
Input

3
2 1 1
1.0000 0.0000
3 1 1
1.0000 0.0000 0.5000
6 2 3
0.9000 0.6000 0.5000 0.1000 0.2000 0.8000
```

```
Output

Case #1: 0.750000000000
Case #2: 0.666666666667
Case #3: 0.618728522337
```

Note

League of Legends is a trademark of Riot Games. Riot Games does not endorse and has no involvement with Google Code Jam.