Problem A. Valar Morghulis

Program:	`winter.(cpp|java)`
Input:	`winter.in`
Balloon Color:	`White`

Khaleesi has set sails for Kings Landing. With three dragons and tens of thousands in her army, she will be unstoppable. The ships are now getting closer to Kings Landing and she want to start planning for her attack.

The plan for the attack is that the army will be split into two wings: right and left. There is one problem though, Khaleesi's soldiers are from very diverse backgrounds and some of them have rivalries and don't get along (like Martells and Tyrells) so they can't be in the same wing.

Khaleesi knows all the rivalries and can try to resolve some of them. Some rivalries are hard to resolve though and she will have to deal with them. She wants to check if some rivalries are not resolved, will it be possible split the soldiers into two wings where no two soldiers who don't get along are in the same wing. Can you help Khaleesi plan for her attack to sit on the iron throne?

Input

Your program will be tested on one or more test cases. The first line of the input will be a single integer **T**, the number of test cases ($1 \leq T \leq 10$)

Each test case starts with a line that contains three space separated integers:

- **N**: The number of soldiers ($1 \leq N \leq 40,000$)
- **E**: The number of rivalries ($1 \leq E \leq 40,000$)
- **Q**: The number of queries ($1 \leq Q \leq 40,000$)

Followed by **E** lines each containing 2 space separated integers **A** and **B** which means that soldier **A** and **B** can not be in the same wing. ($1 \leq A, B \leq N$)

Followed by **Q** lines each containing 2 space separated integers **C** and **D** which means that all rivalries between [C,D] can not be resolved. ($1 \leq C, D \leq E$)

Output

For each test case print **Q** lines where the i^{th} line represents the answer for the i^{th} query. The line should contain "-1 -1"(quotes for clarity) if the army can't be split into two wins or the size of the wings separated by single space otherwise. If there are multiple valid assignments print the one that maximizes the first wing.

Example

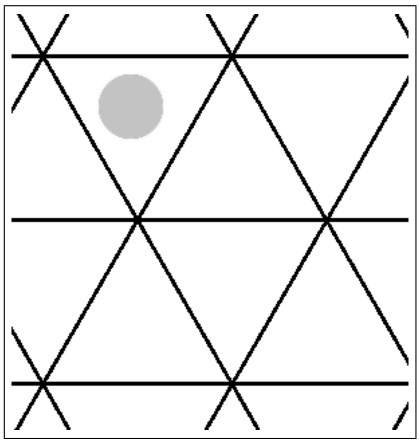
winter.in	standard output
1	-1 -1
9 8 6	-1 -1
5 4	6 3
3 4	7 2
5 6	7 2
5 3	8 1
7 8	
4 8	
5 4	
3 4	
1 8	
1 4	
2 5	
2 4	
4 7	
4 4	

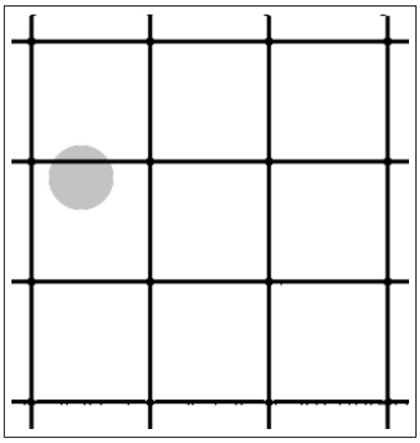
Problem B. Rains

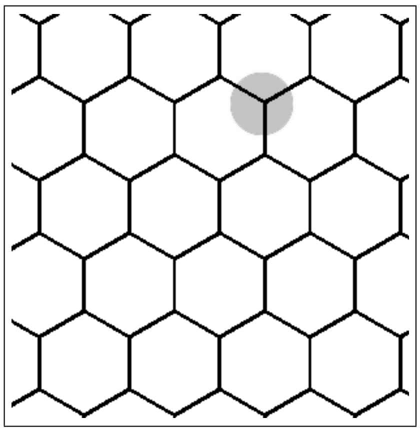
Program:	`end.(cpp|java)`
Input:	`end.in`
Balloon Color:	`Black`

It is the end of the world, the fish community rises up against years of humanity's abuse, and declares war against mankind.

After millions of years of unemployment and reliance on robots, humans became spheres of radius $\boldsymbol{R}$ consisting of only one part, the brain. Fish rebels developed special weapons named "the rains of Atlantic" which consist of large heavy sieves prepared to slice humans, a sieve is composed of extremely thin stretched strings in the form of a grid of convex regular N-sided polygons* of side length $\boldsymbol{S}$. The figure below is an example of the cases when $\boldsymbol{N} = 3, \boldsymbol{N} = 4, \boldsymbol{N} = 6$:


$\boldsymbol{N} = 3$


$\boldsymbol{N} = 4$


$\boldsymbol{N} = 6$

You are the last survivor but they have finally identified the city where you hide. They are going to throws a sieve that will cover the whole city. Your brain will be sliced if one of the threads from the sieve touches it. What is the probability that your brain will be sliced if you can be located anywhere in the city with equal probability?

Input

Your program will be tested on one or more test cases. The first line of the input contains a single integer $\boldsymbol{T}$ $(1 \leq \boldsymbol{T} \leq 100)$ the number of test cases.

Each test case consists of one line contains three integers:

- **N**: The number of sides of the polygon in the sieve $(3 \leq \text{N} \leq 100)$
- **S**: The side length of the polygon in the sieve $(1 \leq \text{S} \leq 1,000,000)$
- **R**: The radius of your brain $(1 \leq \text{R} \leq 1,000,000)$

Output

For each test case, print a single line containing the probability that your brain will be sliced, rounded to four decimal places.

Example

end.in	standard output

2
82 875072 117331
50 475634 389028
	0.0205
	0.1952

Note

A regular polygon is a polygon that is equiangular (all angles are equal in measure) and equilateral (all sides have the same length)

Problem C. Man in the middle

Program:	`code.(cpp|java)`
Input:	`code.in`
Balloon Color:	`Green`

Alice and Bob are good friends who trust each other even though they live in different countries. One day Bob had an idea for a startup that will change the world. However, he was in need of funds. Alice agreed to give him the money he needs. As Bob doesn't have a bank account, Alice told him she'll transfer the money to her friend Eve who lives in the same city with Bob, and all he needs to do is tell Eve the secret code and she'll hand him the money.

Since they were communicating over the internet Alice was worried that someone might be spying on their conversation and can hear the secret code. Bob is one of the brightest brains Alice has met, so she decided to tell Bob the code after being hashed by a certain function and she knows that if someone else was listening to their conversation, Bob will still manage to figure out the secret code first. Bob knows the following about the code:

- The code has length **L**
- The code is only formed of letters A-Z (uppercase only)
- The formula used for calculating the hash:
$$((\sum_{i=1}^{L-1} n_i * M^{L-i}) + n_L)\%10007$$
- n_i would be the numerical value that represents the letter at index i $(1 \leq i \leq L)$, where A=0, B=1, C=2, ... Z=25
- The code will be the least lexicographically string formed of letters A-Z that hash to a given value **H** using the function above.

Can you help Bob figure out the secret code quickly before someone else figures out?

Input

Your program will be tested on one or more test cases. The first line of the input will be a single integer **T** the number of test cases. $(1 \leq T \leq 100)$

Each test case consists of a line containing three space separated integers:

- **L**: The length of the secret code $(1 \leq L \leq 100,000)$
- **H**: The hash value of the code $(0 \leq H < 10,007)$
- **M**: The multiplier used in the hash formula, $(0 \leq M \leq 20)$

Output

For each test case, print a single line containing the secret code if it is possible to find such a string, or "None"if there isn't one (quotes are for clarity).

Examples

code.in	standard output

2
2 50 2
4 250 3
	NY
	ARYZ

Problem D. Lumbo Jumbo

Program:	`lumbo.(cpp|java)`
Input:	`lumbo.in`
Balloon Color:	`Cyan`

In Lala land, young Lumbo and his older brother Jumbo used to play a game that they loved so much. Initially, the game was as follows. Jumbo chooses a number N, then he draws a $3 \times N$ grid on the floor marking some cells as important. The grid cells are referenced by a pair (i, j), where i is the row number, such that the uppermost row has number 1, and j is the column number, such that the leftmost column has number 1. The grid has 3 rows and N columns. Afterwards, Jumbo asks Lumbo to try his best jumping with one leg on the grid cells such that he follows the following rules:

1. He can start from any cell he chooses.
2. Each important cell must be visited at least once.
3. He can only jump from one cell to another if they are adjacent. Two cells are adjacent if they share at least one edge.
4. Visiting non-important cells is allowed, as many times as Lumbo wants. Otherwise, the game can be impossible.

One day, Jumbo thought that the game is too easy that anyone, son of yesterday in his own words, can easily finish the game, so he decided to add new rules to it. So, in addition to the previous rules stated, Jumbo added the following rule:

5. Jumbo assigned a cost to each edge in the grid, between two adjacent cells, such that to cross this edge, Lumbo has to pay the cost assigned to this edge. To make it a bit easier, Jumpo said that whenever Lumbo pays the cost of one edge, he can use it as many times as he wants without paying again.

Then, as before, he asked Lumbo to jump with one leg on the grid cells such that he visits each important cell at least once, but this time he should minimize the overall cost he has to pay.

Lumbo tried some jumps but he is not sure whether any of them has the minimum cost. Lumbo feels that it is too hard for him to do this, so he asked you to help him compute the minimum cost. Can you help him win over his brother and prove that he is son of today?

Input

Your program will be tested on one or more test cases. The first line of the input will be a single integer **T**, the number of test cases $(1 \leq T \leq 100)$

Each test case starts with a line containing two integers:

- **N**: The number of columns of the grid $(1 \leq N \leq 10,000)$
- **P**: The number of important cells in the grid. $(1 \leq P \leq 100)$

The next 3 lines will each contain **N - 1** space separated integers describing the costs of the vertical separating edges. So, the first line describes the costs of the vertical edges separating the cells of the first row $((1, 1), (1, 2), ..., (1, N))$, and similarly for the second and third lines.

The following 2 lines will each contain **N** space separated integers describing the costs of the horizontal separating edges. So, the first line describes the costs of the edges separating cells $(1, c)$ and $(2, c)$ for each column c, and similarly, the second line describes the costs of the edges separating cells $(2, c)$ and $(3, c)$ for each column c. $(0 \leq edge_c osts \leq 100)$

This will be followed by **P** lines each containing 2 space separated integers:

- **i**: The row number of the important point $(1 \leq i \leq 3)$
- **j**: The column number of the important point $(1 \leq j \leq N)$

Output

For each test case, print a single line containing the minimum cost Lumbo needs to pay to jump on the grid cells visiting each important cell at least once.

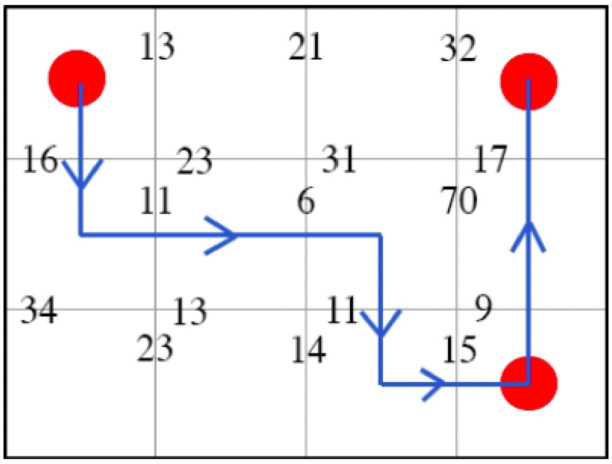
Example

lumbo.in	standard output

2
3 4
11 28
1 62
94 89
31 15 64
76 57 33
1 2
1 1
1 3
2 3
4 3
13 21 32
11 6 70
23 14 15
16 23 31 17
34 13 11 9
1 1
3 4
1 4
	103
	85

Note

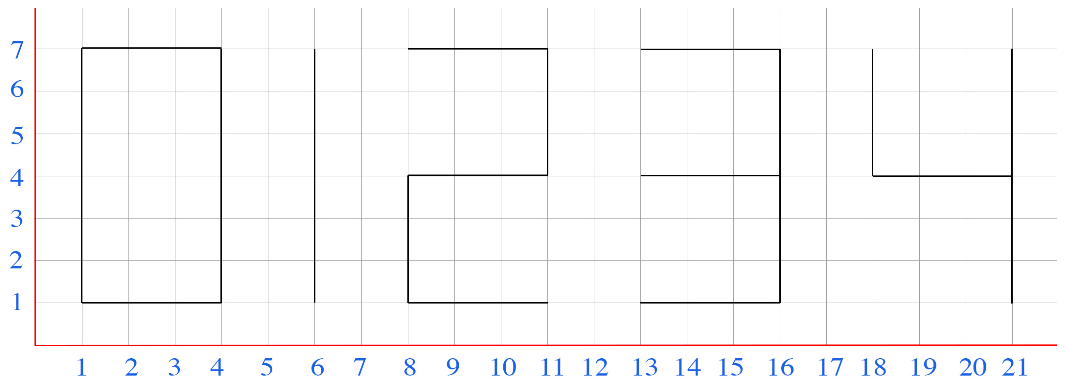
Here is an explanation for the second sample test case:



Problem E. Scrambled Digits

Program:	`digits.(cpp|java)`
Input:	`digits.in`
Balloon Color:	`Purple`

Little Petya started to learn how to write some digits, he is just learning to write the following 5 digits:



He draws each digit exactly with the same dimensions as the above image, but he might make zero or more of the following updates:

- Shrink or expand the width of the digit, keeping the width as an integer number of units.
- Shrink or expand the height of the digit, keeping the height as an integer number of units
- Rotate the digit clockwise by 90, 180 or 270 degrees.

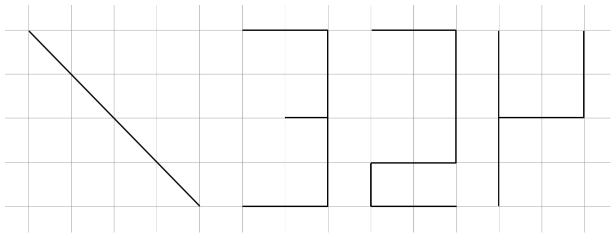
Note that shrinking and expanding a digit doesn't change the thickness of a segment, and it affects the whole digit equally. For example shrinking or expanding the digit 3 vertically (before rotating it), will always keep the middle horizontal segment exactly in the middle between the other horizontal segments.

Now he has an infinite 2D plane, where he writes a lot of digits, but he keeps the following rules valid:

- The digits won't touch each other, or overlap (the segments of each digit won't touch the segments of other digits).
- All vertical segments will be placed on a vertical line with an integer X-value.
- All horizontal segments will be placed on a horizontal line with an integer Y-value.
- All end points will be on a point with integer coordinates (an end point is a point at the end of a specific vertical or horizontal segment).

Petya will draw the digits by drawing line segments, each segment starts and ends at points with integer coordinates, and all segments are guaranteed to be vertical or horizontal with positive integer length. Note that the segments can touch or even overlap, but when this happens, these segments belong to the same instance of the same digit (check the second sample test case).

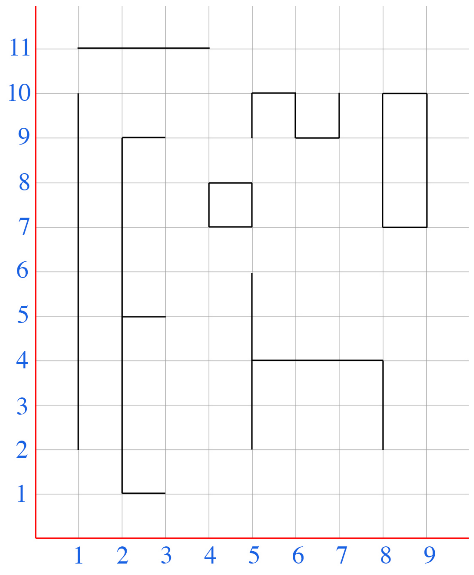
The following image represents some invalid digits which will never appear in the input:



The 1 is rotated incorrectly, the length of the middle horizontal segment in the 3 is wrong, the middle horizontal segment in the 2 isn't in the middle anymore and the 4 is mirrored (not rotated).

You will be given a list of line segments which Petya draw, and it's guaranteed that these line segments will form only valid digits as described above. Your task is to find how many times each digit was drawn.

- The first sample test case represents the first image.
- The second sample test case represents the following image (note that some segments might overlap and/or touch to form one longer segment):



Input

Your program will be tested on one or more test cases. The first line of the input will be a single integer **T**, the number of test cases ($1 \leq \mathbf{T} \leq 100$). Followed by **T** test cases.

Each case starts with a line containing an integer **N** the number of segments to draw ($1 \leq \mathbf{N} \leq 100{,}000$). Followed by **N** lines, each line contains 4 integers, **X1**, **Y1**, **X2** and **Y2**, representing a line segment between the points (**X1**, **Y1**) and (**X2**, **Y2**) ($1 \leq \mathbf{X1}, \mathbf{Y1}, \mathbf{X2}, \mathbf{Y2} \leq 1{,}000{,}000{,}000$).

Output

For each test case, print 5 integers separated by a single space, each integer represents how many times each digit was drawn, in the same order from left to right like the first image.

Example

digits.in	standard output
2	1 1 1 1 1
17	2 2 1 1 1
1 1 1 7	
1 7 4 7	
4 7 4 1	
4 1 1 1	
6 1 6 7	
11 1 8 1	
8 1 8 4	
8 4 11 4	
11 4 11 7	
11 7 8 7	
13 1 16 1	
16 1 16 7	
16 7 13 7	
16 4 13 4	
21 1 21 7	
21 4 18 4	
18 4 18 7	
24	
1 2 1 4	
1 4 1 8	
1 6 1 10	
1 11 4 11	
2 1 2 9	
2 1 3 1	
2 5 3 5	
2 9 3 9	
4 7 4 8	
4 8 5 8	
5 8 5 7	
5 7 4 7	
5 2 5 6	
5 4 8 4	
8 4 8 2	
5 9 5 10	
5 10 6 10	
6 10 6 9	
6 9 7 9	
7 9 7 10	
8 7 8 10	
8 10 9 10	
9 10 9 7	
9 7 8 7	

Problem F. Andorra

Program:	`andorra.(cpp|java)`
Input:	`andorra.in`
Balloon Color:	`Gold`

Andorra is a horizontal city that consists of a set of side-by-side blocks. Each block has a number corresponding to its type and multiple blocks can have the same type.

Andy is the contractor of Andorra. From experience, when Andy meets an investor he usually knows what type of blocks he will be interested in buying. Showing the investor only that type will be very obvious and might make the investor negotiate and reduce the price.

Andy, being the smart guy he is, makes a tour for the investor, the tour could be one block, or multiple consecutive blocks, and the tour should contain at least one block of the type the investor wants. After the tour, Andy manages to sell all blocks of that type to the investor. A tour can not include blocks that have been sold before, otherwise owners will be upset.

Andy has a series of investor coming his way and he will deal with them in the order of their arrival. He needs your help in planning. For each investor he wants to know;

- How many tours are possible to give to them.
- How many future tours are available after being done with that investor. (Remember any tour can't contain a property that was sold)

Input

Your program will be tested on one or more test cases. The first line of the input will be a single integer **T** $(1 \le T \le 100)$.

Each test case starts with a line that contains two space separated integers:

- **N**: The number of available city blocks $(0 \le N \le 100,000)$
- **Q**: The number of investors $(1 \le Q \le 100,000)$

Followed by 2 lines.The first line contains **N** space separated integers, each integer **x** represents the type of block **i**. The second line contains **Q** space separated integers, each integer **y** represents the block type that investor **j** is interested in. $(1 \le x, y \le 1,000,000,000)$.

Output

For each test case, print **Q** lines each containing 2 space separated integers.Each line j contains 2 space separated integers, the number of tours possible for investor j and the number of future tours possible being done with that investor respectively.

Example

andorra.in	standard output

1
2 2
1 2
1 2
	2 1
	1 0

Note

In the test case, Andorra has 2 block. In the beginning, they are all free so Andy can give 3 tours:

- From block 1 to block 1 (1,1)
- From block 1 to block 2 (1,2)
- From block 2 to block 2 (2,2)

1. Investor interested in blocks of type 1 arrives. Andy can make him two tours (1,1) and (1,2). After selling block 1 isn't avaliable no more so Andy can now only make the tour (2,2) so the answer is 2 1
2. Investor interested in blocks of type 2 arrives. Andy can make him only one tour (2,2). After selling, no tours are possible since they are all sold now so the answer is 1 0.

Problem G. Gotta Catch 'Em All

Program:	`catcher.(cpp|java)`
Input:	`catcher.in`
Balloon Color:	`Silver`

Ash Ketchum is a Pokémon trainer who is on a mission to catch all Pokémons in the world. Fortunately, he has caught most of them and he needs to collect K more different types of Pokémons to complete his set.

As you may already know, Ash lives in a 2D grid world. Pokémons spawn at integer coordinates on this grid. In this world there can be multiple Pokémons of the same type. Remember, Ash needs to catch K more different types of Pokémons not any K Pokémons.

Ash decided to catch those Pokémons by throwing a big squared net from the sky. A Pokémon is considered to be caught if the Pokémon lies within the boundaries of the net or even on one of its edges. Ash wants to find a square that contains those Pokémons where each side of the square is either parallel to the x-axis or to the y-axis.

Can you help Ash find the minimum side of the square that contains K different types of Pokémons so that he can buy an appropriate sized net to catch'em all? Since nets need to always have a positive area, the side of the square needs to be positive.

Input

Your program will be tested on one or more test cases. The first line of the input will be a single integer **T**, the number of test cases ($1 \leq T \leq 100$).

Each test case starts with a line that contains two space separated integers:

- **N**: Number of Pokémons in the world ($2 \leq N \leq 1000$)
- **K**: Number of types of Pokémons ($2 \leq K \leq 100$)

Followed by **N** lines each containing 3 space separated integers:

- **X**: The x-coordinate of the Pokémon($-1{,}00{,}000{,}000 \leq X \leq 1{,}00{,}000{,}000$)
- **Y**: The y-coordinate of the Pokémon ($-1{,}00{,}000{,}000 \leq Y \leq 1{,}00{,}000{,}000$)
- **Z**: The type of the Pokémon ($1 \leq Z \leq K$)

Output

For each test case, print a single line containing the minimum side of a square that contains **K** different types of Pokémons.

Example

catcher.in	standard output

2
5 2
0 0 1
0 1 1
0 2 1
2 0 2
2 1 2
3 3
0 0 1
1 1 2
2 2 3
	2
	2

Problem H. Military Service

Program:	`soldiers.(cpp|java)`
Input:	`soldiers.in`
Balloon Color:	`Red`

You have just started your military service with the border guards. Since you are a Computer Science graduate, they asked you to implement a schedule for the soldiers. Initially no soldier is on duty and you need the schedule to satisfy 2 requirements:

- There are **N** soldiers, each soldier can be on duty for at most **K** continuous months and then he must take one month of vacation.
- The schedule needs to make sure that the guaranteed minimum number of soldiers on duty at any given time is maximized.

Given **N** and **K**, the system will calculate the maximum guaranteed number of soldiers to be on duty at any given time.

Input

Your program will be tested on one or more test cases. The first line of the input will be a single integer **T**, the number of test cases ($1 \le T \le 100$).

Each test case consists of a line containing 2 space separated integers:

- **N**: The number of soldiers ($0 \le N \le 10,000,000$)
- **K**: The number of continuous months a soldier can be on duty before they have to take a month of vacation ($0 \le K \le 10,000,000$).

Output

For each test case, print a single line containing the maximum guaranteed number of soldiers on duty at any given time.

Examples

soldiers.in	stdout

3
4 1
9 3
21 3
	2
	6
	15

Problem I. Baklawa

Program:	`baklawa.(cpp|java)`
Input:	`baklawa.in`
Balloon Color:	`Pink`

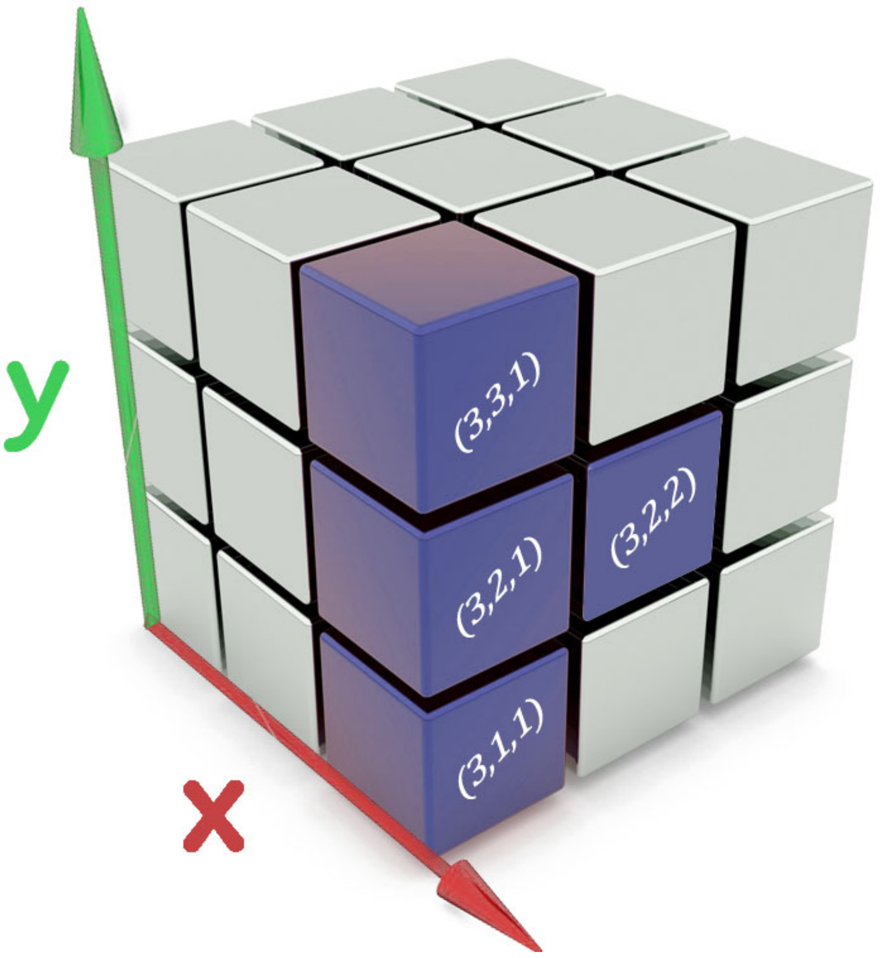
Baklawa or baklava, is a sweet middle eastern dessert, mainly made from phyllo dough sheets, walnuts, and sugar syrup cut into small cubic pieces and served in cuboid boxes containing multiple layers.

Alice and Bob love to play what they call the last baklawa game, The rules are as follows:

- They choose a cuboid containing X by Y by Z cells of baklawa to play with.
- There are N poisonous cells both players know them.
- Alice plays first, and the two players alternate.
- In his/her turn the player has to cut the cuboid into two cuboids at least one of which is safe (doesn't contain a poisonous cube) and eats the safe part. The game continues with the other part.
- A cut is made by slicing along one of the axis X, Y or Z.
- The person who cannot make a move in his/her turn losses the game.

Assuming both players play optimally, you are asked the following question: who wins the game?

The figure below is an example of the first test case:



Input

Your program will be tested on one or more test cases. The first line of the input will be a single integer **T**, the number of test cases ($1 \leq T \leq 100$).

Each test case starts with a line containing four space separated integers:

- **X**: The length of the cuboid ($1 \leq X \leq 1,000,000,000$)
- **Y**: The width of the cuboid ($1 \leq Y \leq 1,000,000,000$)
- **Z**: The height of the cuboid ($1 \leq Z \leq 1,000,000,000$)
- **N**: The number of poisonous cells. ($0 \leq N \leq 100$)

Followed by **N** lines each containing three space separated integers:

- x_i: The x-coordinate of the i^{th} cube. ($1 \leq x_i \leq X$)
- y_i: The y-coordinate of the i^{th} cube. ($1 \leq y_i \leq Y$)
- z_i: The z-coordinate of the i^{th} cube. ($1 \leq z_i \leq Z$)

Output

For each test case, print a single line containing "Alice" if Alice wins the game, or "Bob" otherwise.

Example

baklawa.in	standard output

2
3 3 3 4
3 2 1
3 1 1
3 3 1
3 2 2
5 5 5 1
3 4 2
	Alice
	Bob

Problem J. Football Hooliganism

Program:	`football.(cpp|java)`
Input:	`football.in`
Balloon Color:	`Orange`

Football hooliganism is a conflict between gangs of association football clubs supporters formed for the specific purpose of intimidating and physically attacking supporters of other teams.

Unfortunately after this year Byteland Clasico between team zero and team one, rival fans clashed with each other and riot police, they set fire to recycle bins and smashed windows, while police tried to disperse them by firing tear gas, many people were arrested, but in order to prevent more fightings in the jail, authorities asked you to redesign the jail taking in consideration the following rules:

- The jail is a grid of N by 2 identical cells where each prisoner is assigned to a single cell.
- You can join adjacent cells to build larger but only rectangular cells.
- The cells can not overlap and prisoners can not be moved.
- Each prisoner supports either team zero or team one.
- All prisoners in the same cell must support the same team.
- Byteland people are very social, you have to minimize the number of solitary confinements (cells that have a single prisoner) where prisoners may feel lonely and commit suicide.

You are asked the following question: what is the minimum number of solitary confinements assuming you are joining the cells optimally.

Input

Your program will be tested on one or more test cases.The first line contains the number of test cases **T** ($1 \leq T \leq 100$). Each test cases starts with a line with integer **N** ($1 \leq N \leq 10^4$) the length of the grid. Following this **N** lines each contains a string of length 2 indicating the team prisoners in cell 1 and 2 of the i^{th} row support respectively.The teams can take values 0 or 1.

Output

For each test case, print a single line containing the minimum number of solitary confinements.

Example

football.in	standard output
2 6 01 01 01 01 10 00 4 00 10 10 10	2 0

Problem K. Pokémon Buddy

Program:	`pokemon.(cpp|java)`
Input:	`pokemon.in`
Balloon Color:	`Blue`

Pokémon Go just released the Buddy update. It lets you select a Pokémon to appear alongside your trainer's avatar on your profile screen. As you walk with your buddy, it will find candy that can be used to evolve the Pokémon.

The Buddy system divides the Pokémons into 3 groups. Each group gives one candy upon walking for 1, 3, and 5 kilometers respectively.

In this problem you will be given the Pokémon group **G**, the number of candies **C** you initially have, and the number of candies **E** required to evolve the Pokémon. You should calculate the number of Kilometers required to walk in order to evolve the Pokémon.

Input

Your program will be tested on one or more test cases. The first line of the input will be a single integer **T**, the number of test cases ($1 \leq T \leq 100$).

Each test case consists of a line containing three space separated integers:

- **G**: The group of the Pokémon ($1 \leq G \leq 3$)
- **C**: The initial candies you have ($0 \leq C \leq 100$)
- **E**: The candies required to evolve the Pokémon ($1 \leq E \leq 100$)

Output

For each test case, print a single line containing the number of Kilometers of walking required to Evolve the Pokémon.

Example

pokemon.in	standard output

2
1 15 51
1 18 21
	36
	3

Problem L. Shika Bika

Program:	`shikabika.(cpp|java)`
Input:	`shikabika.in`
Balloon Color:	`Yellow`

Shika and Bika are two cute little sisters. They like to play together all the time. These days, they are learning about integers. They developed the following game. Shika chooses a certain range, without telling Bika about it, then the game goes in rounds. In each round, Shika calls out one integer from the range she chose, then Bika replies with another integer (it needn't be in the range, just any integer). Shika ends the game only after making sure that she called out each integer in the range at least once. This means Shika can call out an integer more than once and Bika can reply with different integer each time. For example, Shika might call out 3 in a round and Bika replies with 4, then later in another round, Shika calls out 3 again but Bika replies with 3 this time. After every round, they write down a pair consisting of the integer that Shika called out in the round and the integer Bika replied with. The write up has the following problems:

- They do not write the pair in a certain order. So, if Shika says x, and Bika replies with y, they sometimes write this pair as (x, y) and other times as (y, x).
- In some rounds, they forget to write the pair at all.

Next day after the game, Shika asks Bika questions like: "Did I call out integer q yesterday?". Bika has the write up and she is allowed to answer with one of the following answers. "YES", "NO", or "NOT SURE". Given the written pairs, can you help Bika answer Shika's questions?

Input

Your program will be tested on one or more test cases. The first line of the input will be a single integer **T** $(1 \leq T \leq 100)$.

Each test case represents a game and starts with a line that contains two space separated integers:

- **N**: The number of written pairs $(1 \leq N \leq 1000)$
- **Q**: The number of queries $(1 \leq Q \leq 1000)$

Followed by **N** lines each containing 2 space separated integers **a** and **b** representing the i^{th} pair $(-1,000,000 \leq a, b \leq 1,000,000)$

Followed by **Q** lines each containing a single integer **A** representing the j^{th} query

$(-1,000,000 \leq A \leq 1,000,000)$

Output

For each test case, print Q lines, where the i_{th} line answers the i_{th} query with "YES", "NO", or "NOT SURE".

Examples

shikabika.in	stdout

1
2 3
10 7
20 14
3
8
11
	NOT SURE
	NOT SURE
	YES

Note

In the test case, we can see that we have the following possibilities:

1. – Round 1: Shika called out 10, Bika replied with 7
   – Round 2: Shika called out 20, Bika replied with 14
2. – Round 1: Shika called out 10, Bika replied with 7
   – Round 2: Shika called out 14, Bika replied with 20
3. – Round 1: Shika called out 7, Bika replied with 10
   – Round 2: Shika called out 20, Bika replied with 14
4. – Round 1: Shika called out 7, Bika replied with 10
   – Round 2: Shika called out 14, Bika replied with 20

For the first query, Bika would never be able to decide if 3 was called out by Shika or not, since some rounds' pairs may be missing. For the second query, in possibilities 1 and 2, Bika can't guarantee that Shika called out 8. For the third query, 11 was for sure called out by Shika, as it is between the two integers called out by Shika in all possibilities.