The 3rd Universal Cup

Stage 19: Shenyang

November 30 - December 1, 2024

This problem set should contain 13 problems on 24 numbered pages.

Based on

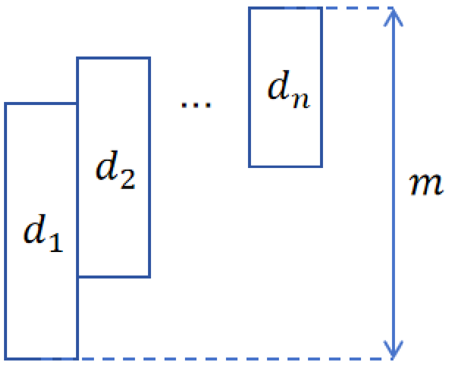
International Collegiate Programming Contest (ICPC)

Prepared & Hosted by

Problem A. Safety First

Time limit:	3 seconds
Memory limit:	1024 megabytes

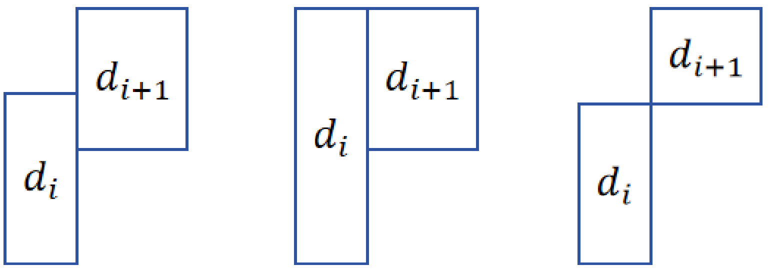
A ladder has n sections, each of which has a positive integer length, and the lengths are sequentially non-increasing. That is, the lengths of these n sections from left to right $d_1, d_2, \ldots, d_n$ should satisfy $d_1 \geq d_2 \geq \cdots \geq d_n$.



Ladder

Little Q needs a ladder that, when stood upright, has a height exactly equal to m, where the height of a ladder is defined as the altitude difference between its highest end and its lowest end. For safety reasons, this ladder needs to be stable, which means it satisfies the following requirements:

- The lower end of the first section of the ladder must touch the ground;
- For each $i = 1, 2, \ldots, n-1$, the upper end of the i-th section must be locked into either the upper end or the lower end of the $(i + 1)$-th section.



Not stable Stable Stable

You need to compute the number of different stable ladders with n sections that can be constructed if their height when stood upright is exactly m. Two ladders are considered different if and only if there exists some k such that the lengths of the k-th section of the two ladders are different, or if the k-th section of one ladder is locked into the upper end of the $(k + 1)$-th section while the k-th section of the other ladder is locked into the lower end of the $(k + 1)$-th section.

Since the answer may be large, output it modulo 998 244 353.

Input

The first line of the input contains an integer T ($1 \le T \le 10^5$), indicating the number of test cases. For each test case:

The only line contains two integers n and m ($1 \le n, m \le 2\,000$), indicating the number of sections and the required height of the stable ladder.

Output

For each test case, output a line containing an integer, indicating the number of different stable ladders modulo 998 244 353.

Examples

standard input	standard output

3
1 3
2 3
3 3
	1
	4
	10

1 2000 2000	204576309

Note

In the following explanation, we use **bold text** to indicate that the upper end of the corresponding section is locked into the lower bound of the next section and keep the same in other scenarios.

In the first sample case:

- for the first test case, the only possible ladder has a single section of length 1;
- for the second test case, the section lengths of all 4 possible ladders are $[\mathbf{2}, 1]$, $[3, 1]$, $[3, 2]$ and $[3, 3]$ respectively;
- for the third test case, the section lengths of all 10 possible ladders are $[\mathbf{1}, \mathbf{1}, 1]$, $[\mathbf{2}, 1, 1]$, $[2, \mathbf{1}, 1]$, $[2, \mathbf{2}, 1]$, $[3, 1, 1]$, $[3, 2, 1]$, $[3, 2, 2]$, $[3, 3, 1]$, $[3, 3, 2]$ and $[3, 3, 3]$ respectively.

Problem B. Magical Palette

Time limit:	1 second
Memory limit:	1024 megabytes

The little white rabbit has a magical palette with a grid of n rows and m columns. Before starting to mix the colors, the little white rabbit will squeeze a kind of pigment to the left of each row, denoted by $a_1, a_2, \ldots, a_n$, and also squeeze a kind of pigment above each column, denoted by $b_1, b_2, \ldots, b_m$.

There are a total of $n \times m$ kinds of selectable pigments, represented by integers $0, 1, 2, \ldots, nm - 1$ for different pigments. Then, in the cell of the i-th row and the j-th column, the little white rabbit will mix a color $c_{i,j} = a_i b_j \bmod nm$ using the pigment a_i to the left of the i-th row and the pigment b_j above the j-th column.

The little white rabbit hopes that each of the $n \times m$ cells has a different color, and you need to find out whether it can be achieved.

Input

The first line of the input contains an integer T ($1 \le T \le 10^4$), indicating the number of test cases. For each test case:

The only line contains two integers n and m ($1 \le n, m \le 10^6$, $1 \le n \times m \le 10^6$), indicating the number of rows and the number of columns.

It is guaranteed that the sum of $n \times m$ over all test cases does not exceed 10^6.

Output

For each test case, if no solution exists, output "`No`" (without quotes) in one line. Otherwise, output three lines:

- The first line contains one string "`Yes`" (without quotes).
- The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i < nm$).
- The third line contains m integers $b_1, b_2, \ldots, b_m$ ($0 \le b_i < nm$).

Example

standard input	standard output

2
2 3
2 2
	Yes
	1 2
	1 3 5
	No

Note

For the first sample case, $[c_{1,1}, c_{1,2}, c_{1,3}, c_{2,1}, c_{2,2}, c_{2,3}] = [1, 3, 5, 2, 0, 4]$, which are pairwise different.

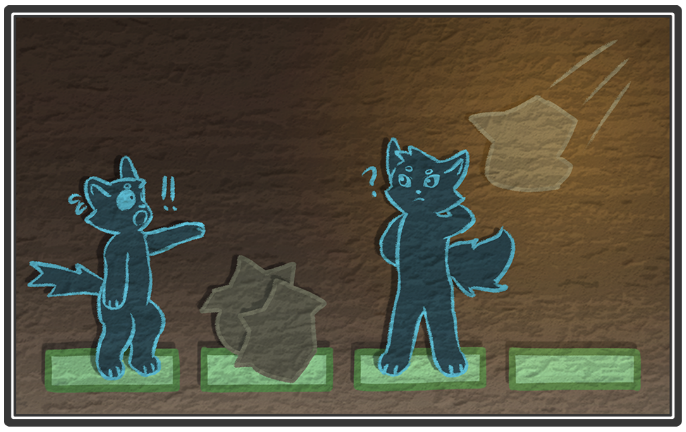
Problem C. Crisis Event: Meteorite

Time limit:	2 seconds
Memory limit:	1024 megabytes

Sulfox the fennec fox has recently become obsessed with *Anomaly Collapse*, a roguelite turn-based strategy game built around its innovative one-dimensional battlefield grid system. As he controls his squad of warriors to fight against enemies on the special battlefield, where positioning mechanisms and proper allocation of ability points are crucial for victory, he encounters another crisis event (a thoughtful "gift" from the game designers) as the factor introduced to escalate the challenge of the new chapter — Meteorite.

For this problem, let's push this crisis event to a more extreme version than the original game. Consider a linear battlefield consisting of n grids numbered from 1 to n sequentially. Initially, the battlefield is clear of any meteorites, and several characters are positioned on certain grids. Sulfox's objective is to ensure the survival of all characters through m rounds of the meteorite disaster.

At the beginning of the i-th round ($i = 1, 2, \ldots, m$), the fatal meteorite hazard unfolds: For each $j = 1, 2, \ldots, n$, exactly $a_{i,j}$ (possibly zero) meteorites descend upon the j-th grid, accumulating with any existing meteorites. Movement of characters is prohibited during this phase, which means any character unfortunate enough to be standing on a grid where meteorites descend will instantly be squashed into a pancake.


Do not let character stand on the grid where meteorites descend

The action phase occurs after all meteorites for the i-th round have landed, during which Sulfox can move any surviving character to an adjacent grid, and this movement may be performed arbitrarily many times (possibly zero) for each character before this round ends. However, existing meteorites block characters from entering their occupied grids. Therefore, before a character can enter such a grid, Sulfox must spend 1 point per meteorite on that grid to destroy them.

Please help Sulfox calculate the minimum number of points required to ensure the survival of all characters throughout the meteorite disaster, or report that it is an impossible mission to achieve the objective.

Input

The first line of the input contains an integer T ($1 \le T \le 10^4$), indicating the number of test cases. For each test case:

The first line contains two integers n and m ($1 \le n, m \le 10^6$, $1 \le n \times m \le 10^6$), indicating the number of grids and the number of rounds respectively.

The second line contains n integers $c_1, c_2, \ldots, c_n$ ($c_i \in \{0, 1\}$). There is a character positioned on the i-th grid initially if and only if $c_i = 1$. It is guaranteed that there is at least one character in the game.

Then m lines follow, the i-th of which contains n integers $a_{i,1}, a_{i,2}, \ldots, a_{i,n}$ ($0 \le a_{i,j} \le 1\,000$), indicating that $a_{i,j}$ meteorites descend upon the j-th grid at the beginning of the i-th round.

It is guaranteed that the sum of $n \times m$ over all test cases does not exceed 10^6.

Output

For each test case, if it is impossible to ensure the survival of all characters, output -1 in one line. Otherwise, output an integer in one line, indicating the minimum number of points required.

Example

standard input	standard output

2
3 4
1 0 1
0 1 0
2 0 0
0 0 3
4 5 0
1 1
1
1000
	4
	-1

Problem D. Dot Product Game

Time limit:	1 second
Memory limit:	1024 megabytes

A permutation of 1 to n in this problem is a 1-based sequence of n integers in which every integer from 1 to n appears exactly once. Alice and Bob are playing a game on $A = [a_1, a_2, \ldots, a_n]$ and $B = [b_1, b_2, \ldots, b_n]$, two permutations of 1 to n. They take turns to perform operations, with Alice going first, and the one with no possible operation loses.

In each turn, Alice can only operate on the permutation A, while Bob can only operate on the permutation B. A valid operation consists of choosing two indices i and j on the operable permutation ($1 \le i, j \le n$) and swapping their corresponding elements, under the constraint that $\sum_{i=1}^{n} a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$, the dot product of these two permutations, must strictly increase after the swap. Namely, a player loses if he or she cannot perform any swap to increase the dot product, and the other player wins the game.

As Alice and Bob are both clever enough to adopt the best strategy, the winner of such a game can be determined from the start. Therefore, they decide to play the game n times with modifications to make it less boring. Please help them to predict the winners of these n games.

More specifically, two initial permutations $A = [a_1, a_2, \ldots, a_n]$, $B = [b_1, b_2, \ldots, b_n]$, and $(n-1)$ modifications $[(t_1, l_1, r_1, d_1), (t_2, l_2, r_2, d_2), \ldots, (t_{n-1}, l_{n-1}, r_{n-1}, d_{n-1})]$ will be given. The first game will start from the given permutations A and B, and for $k = 1, 2, \ldots, n-1$, the $(k+1)$-th game will start from the permutations for the k-th game after shifting the interval between indices l_k and r_k of the permutation t_k left d_k times.

Please note that shifting an interval $[p_l, p_{l+1}, \ldots, p_r]$ left once will give $[p_{l+1}, p_{l+2}, \ldots, p_r, p_l]$. For instance, shifting the interval $[2, 4]$ of $[1, 2, 3, 4, 5]$ left once will give $[1, 3, 4, 2, 5]$, and twice will give $[1, 4, 2, 3, 5]$. Besides, shifting the interval $[1, 5]$ of $[3, 1, 4, 5, 2]$ left 4 times will give $[2, 3, 1, 4, 5]$.

Input

The first line of the input contains an integer T ($1 \le T \le 10^5$), indicating the number of test cases. For each test case:

The first line contains an integer n ($1 \le n \le 5 \times 10^5$), indicating the length of the permutations.

The second line contains n distinct integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$), indicating the permutation A.

The third line contains n distinct integers $b_1, b_2, \ldots, b_n$ ($1 \le b_i \le n$), indicating the permutation B.

In the next $(n-1)$ lines, the i-th line contains a character t_i ($t_i \in \{\texttt{A}, \texttt{B}\}$), and three integers l_i, r_i ($1 \le l_i \le r_i \le n$), and d_i ($1 \le d_i \le n$).

It is guaranteed that the sum of n over all test cases does not exceed 5×10^5.

Output

For each test case, output a string of length n in one line, where the i-th character of the string is `A` if Alice wins the i-th game, or `B` otherwise (i.e., Bob wins the i-th game).

Example

standard input	standard output
5	BBB
3	ABB
1 2 3	AAB
1 2 3	AAB
A 1 1 1	AABBBBAAAA
B 1 1 1	
3	
1 2 3	
2 1 3	
A 1 2 1	
B 2 2 1	
3	
1 2 3	
2 1 3	
A 1 3 1	
B 1 2 1	
3	
1 2 3	
3 2 1	
A 2 2 1	
B 2 3 1	
10	
1 2 3 4 5 6 7 8 9 10	
4 2 3 9 6 1 5 8 7 10	
A 2 9 10	
B 2 7 9	
A 1 10 8	
B 4 6 7	
B 3 10 6	
A 2 5 5	
A 8 9 4	
B 3 9 3	
A 2 7 2	

Note

For the third test case of the sample case:

- The first game starts with $A = [1, 2, 3]$, $B = [2, 1, 3]$;
- The second game starts with $A = [2, 3, 1]$, $B = [2, 1, 3]$;
- The third game starts with $A = [2, 3, 1]$, $B = [1, 2, 3]$.

Problem E. Light Up the Grid

Time limit: 1 second
Memory limit: 1024 megabytes

Little Q is testing a self-designed puzzle game involving a 2×2 grid where each cell can either be on or off. When a cell is toggled, its state changes from off to on or from on to off. The following four operations can be performed at a certain cost:

- Toggle a single cell with cost a_0;
- Toggle a row of cells with cost a_1;
- Toggle a column of cells with cost a_2;
- Toggle all cells with cost a_3.

Little Q has realized that the game has started, but the screen somehow malfunctions, blocking him from seeing the current state of any cell. The only feedback he can receive is a special prompt sound triggered if, after an operation, all cells are on.

Knowing all m possible initial grids of the game in advance, Little Q wants to find a sequence of operations, ensuring that regardless of the initial grid when he starts playing, he will always hear the prompt sound after some operations in the order of the sequence.

Please calculate the minimum total cost of such sequence of operations.

Input

The first line of the input contains five integers T ($1 \le T \le 10^4$), a_0, a_1, a_2, and a_3 ($1 \le a_0, a_1, a_2, a_3 \le 10^6$), indicating the number of games and the costs of each operation. For each game:

The first line contains an integer m ($1 \le m \le 16$), indicating the number of possible initial grids of the game.

Then m grids follow. For each grid:

- There will first be an empty line if it is not the first grid in the game.
- For the following two lines, the i-th line contains a string $s_{i,1}s_{i,2}$ of length 2 consisting of characters `0` and `1`, describing a 2×2 grid as a possible initial grid. Let (i, j) be the cell on the i-th row and the j-th column. If $s_{i,j}$ is `0`, then the cell (i, j) is off, or otherwise the cell (i, j) is on.

It is guaranteed that the given grids in a game are pairwise different.

Output

For each game, output a line containing an integer, indicating the minimum total cost.

Example

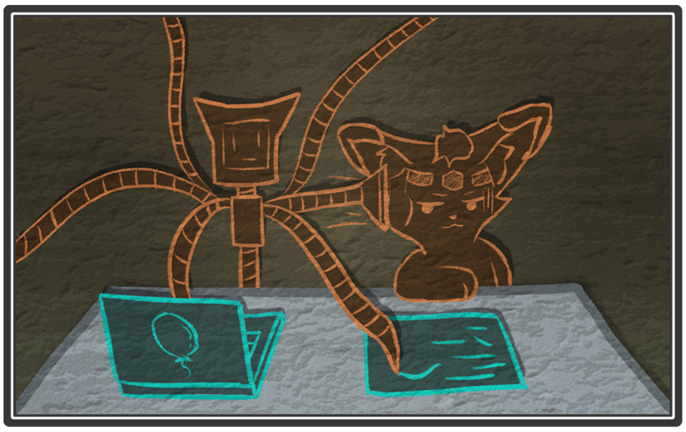
standard input	standard output
2 1000 100 10 1	1121
4	2
10	
00	
01	
00	
00	
10	
00	
01	
1	
11	
11	

Problem F. Light Up the Hypercube

Time limit: 1 second
Memory limit: 1024 megabytes

Do androids dream of electric sheep? This topic, posed decades ago, stung with renewed urgency in future world, serving as a bitter reminder where biological minds had almost surrendered their thinking and creativity to artificial intelligence.

One afternoon in 2040, as Sulfox the fennec fox immersed himself in the pleasure of a simple two-dimensional puzzle game designed by Little Q (introduced in **Problem E. Light Up The Grid**), his solutions caught the attention of his custom AI system. The AI, despite having achieved nearly omnipotent capabilities in these years, failed to empathize with Sulfox's recreational purpose, misinterpreting his joy as a desire for greater challenges. In a display of its superior grasp of higher dimensions, the AI spontaneously provided an n-dimensional puzzle game that only it could truly comprehend.


Self-Important Custom AI System

The enhanced puzzle game involves an n-dimensional hypercube, where each of its edges extends along only one of the n dimensions, and each of its 2^n vertices is equipped with a light that can either be on or off. When a light is toggled, its state changes from off to on or from on to off.

Different from the simple puzzle game, there are 2^n kinds of toggle operations which can be performed at a certain cost, each numbered by an integer from 0 to $(2^n - 1)$. Specifically, for operation i $(i = 0, 1, \ldots, 2^n - 1)$:

- The cost of performing such operation equals a_i;
- Let $\overline{b_n b_{n-1} \ldots b_1}$ be the binary representation of i, where each $b_j \in \{0, 1\}$ $(j = 1, 2, \ldots, n)$ denotes the j-th bit of the binary representation from low to high, such operation allows selecting any valid k-face (subcube of dimensionality k) of the hypercube, where k is the number of $b_j = 1$ in the binary representation, each $b_j = 1$ constrains that the k-face selected must extend along the j-th dimension, and each $b_j = 0$ constrains that the k-face selected must have zero thickness in the j-th dimension. Then, toggle the lights at all vertices of the k-face selected.

For instance, operation 0 toggles the light at a single vertex (0-face) with cost a_0, operations $2^0, 2^1, \ldots, 2^{n-1}$ toggle lights at both endpoints of an edge (1-face) along their corresponding dimensions with cost $a_{2^0}, a_{2^1}, \ldots, a_{2^{n-1}}$ respectively, and operation $(2^n - 1)$ toggles all lights with cost a_{2^n-1}.

Sulfox had realized that the game had started, but the screen had been forbidden by the AI, blocking him from seeing the current state of any light. The only feedback he could receive is a special prompt sound triggered if, after an operation, all lights were on.

Therefore, he needed to find a sequence of operations, ensuring that regardless of the initial state of the lights (among all 2^{2^n} possibilities) when he started playing, he would always hear the prompt sound after some operations in the order of the sequence.

As a three-dimensional being, Sulfox felt impossible to visualize higher dimensions, yet believing the minimum total cost of such sequence of operations was still computable. “OK, AI, I want full manual control now,” he said and set about solving, determined to swim free in the pool of cognition rather than be tethered by a lifejacket until forgetting how to stay afloat.

Please calculate the minimum total cost modulo 998 244 353 since it could be extremely large.

Input

The first line of the input contains an integer T $(1 \le T \le 10^4)$, indicating the number of test cases. For each test case:

The first line contains an integer n $(2 \le n \le 20)$, indicating the dimensionality of the hypercube.

The second line contains 2^n integers $a_0, a_1, \ldots, a_{2^n-1}$ $(1 \le a_i \le 10^6)$, indicating the costs of each operation.

It is guaranteed that the sum of 2^n over all test cases does not exceed 2^{20}.

Output

For each test case, output a line containing an integer, indicating the minimum total cost modulo 998 244 353.

Example

standard input	standard output

2
2
4 2 2 1
4
3 3 2 3 1 3 2 2 2 3 2 1 1 2 3 2
	25
	65666

Problem G. Guess the Polygon

Time limit:	1 second
Memory limit:	1024 megabytes

This is an interactive problem.

You are given a simple polygon, but before giving you the n vertices of the polygon, Little Q has shuffled them.

Now you can ask Little Q no more than $(n-2)$ queries, each of which consists of two integers p and q $(0 \le p/q \le 1\,000, 1 \le q \le 1\,000)$. He will tell you the total length of the points on the line $x = p/q$ that lie inside or on the boundary of the polygon, which can be represented as r/s with two integers r and s $(r \ge 0, s \ge 1, \gcd(r, s) = 1)$. Here $\gcd(r, s)$ is the greatest common divisor of r and s.

You need to find the area of the polygon, which can also be represented as u/v with two integers u and v $(u \ge 0, v \ge 1, \gcd(u, v) = 1)$.

The polygon and the order of the vertices are determined before the start of the interaction and do not depend on your queries. In other words, the interactor is not adaptive.

Input

The first line of the input contains an integer T $(1 \le T \le 1\,000)$, indicating the number of test cases. For each test case:

The first line contains an integer n $(3 \le n \le 1\,000)$, indicating the number of vertices of the polygon.

Then n lines follow, each containing two integers x and y $(0 \le x, y \le 1\,000)$ that give the coordinates (x, y) of the vertices of the polygon in shuffled order.

The polygon is simple, i.e., its vertices are distinct and no two edges of the polygon intersect or touch, except that consecutive edges touch at their common vertex. In addition, no two consecutive edges are collinear.

It is guaranteed that the sum of n over all test cases does not exceed $1\,000$.

Interaction Protocol

If you want to ask a query, output one line. First output `?` followed by a space, then output two integers p and q $(0 \le p/q \le 1\,000, 1 \le q \le 1\,000)$. After flushing your output, your program should read two integers r and s $(r \ge 0, s \ge 1, \gcd(r, s) = 1)$, indicating the answer to your query.

If you want to guess the area of the polygon, output one line. First output `!` followed by a space, then output two integers u and v $(u \ge 0, v \ge 1, \gcd(u, v) = 1)$, indicating that the area of the polygon is u/v. After flushing your output, your program should continue processing the next test case, or exit immediately if there are no more test cases. Note that your guess does not count as a query.

To flush your output, you can use:

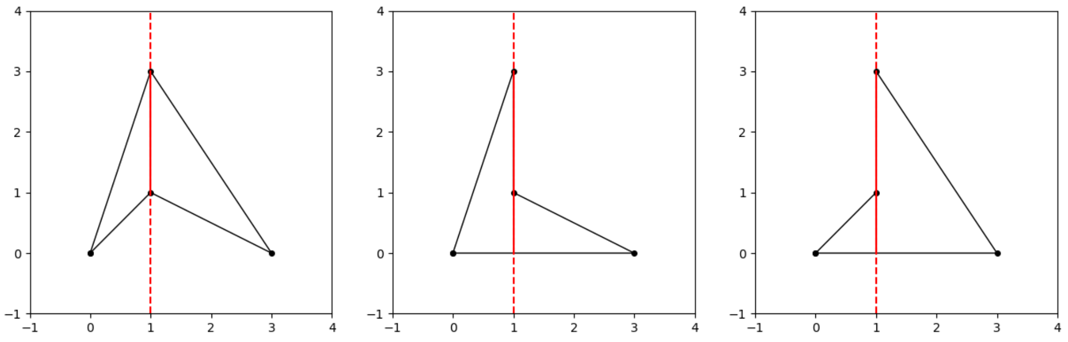
- `fflush(stdout)` (if you use `printf`) or `cout.flush()` (if you use `cout`) in C and C++.
- `System.out.flush()` in Java and Kotlin.
- `sys.stdout.flush()` in Python.

Example

standard input	standard output
2	
4	
0 0	
1 3	
1 1	
3 0	
	? 1 1
2 1	
	! 3 1
3	
0 0	
999 1000	
1000 999	
	? 1 1
1999 999000	
	! 1999 2

Note

For the first sample case, there are three candidate polygons, shown in the figure below. Only the leftmost one meets the answer 2 to the query $x = 1$, and thus the area is 3. The other two have the answer 3 to the query $x = 1$.



For the second sample case, there is only one candidate polygon, and thus the area is 1999/2.

Note that the order of the vertices shown in the sample case may not be consistent with the actual provided.

The blank lines in the sample case are added for readability. In your output, extra spaces or blank lines will be ignored.

Problem H. Guide Map

Time limit:	2 seconds
Memory limit:	1024 megabytes

Byteland has n cities, and there is a bidirectional road between any two cities. There are exactly $(n-2)$ roads with scenery, and if properly built scenery on one more road, one can always travel from one city to the others only through roads with scenery.

Little Q is initially located in the city 1 with a guide map in his hand, showing some of the roads in Byteland, and starts his journey in Byteland. Assuming he is currently in the city u:

- If he can walk along a road on the guide map to a city v that he has not yet visited, he will choose the smallest such v and walk along the road to city v. Note that city 1 is visited initially;
- Otherwise, he will return along the road he took to reach city u for the first time, unless he is already in city 1, in which case the journey ends immediately.

Little Q wants to visit all the sceneries while traveling on no more than one different road without sceneries. You need to help him calculate how many different guide maps can satisfy his requirements. Two guide maps are considered different if and only if there is a road on one map that does not appear on the other.

Since the answer may be large, output it modulo 998 244 353.

Input

The first line contains an integer n ($2 \le n \le 2 \times 10^5$), indicating the number of cities.

Then $(n-2)$ lines follow, the i-th of which contains two integers u and v ($1 \le u, v \le n$), indicating that the i-th scenery is on the road between the u-th and the v-th cities. It is guaranteed that if properly built scenery on one more road, one can always travel from one city to the others only through roads with scenery.

Output

Output a line containing an integer, indicating the number of different guide maps modulo 998 244 353.

Examples

standard input	standard output

4
1 4
2 3
	6

2	2

Note

For the first sample case, one of the feasible guide maps shows 4 roads $(1,4)$, $(2,3)$, $(1,2)$, and $(1,3)$. Little Q will visit the cities $1 \to 2 \to 3 \to 2 \to 1 \to 4 \to 1$ in order following this guide map and visit the sceneries on roads $(1,4)$ and $(2,3)$ while traveling on only one road $(1,2)$ without sceneries.

For the second sample case, there is no road with scenery, and thus a feasible guide map can not only include but also exclude the only existing road $(1,2)$.

Problem I. Growing Tree

Time limit:	2 seconds
Memory limit:	1024 megabytes

Little Q has planted a tree, which initially has only one root node 1, which is also a leaf node. Every morning, each leaf node u will sprout two branches:

- One branch connects node u and node $2u$, with a length of a_{2u};
- The other branch connects node u and node $2u+1$, with a length of a_{2u+1}.

After that, node u is no longer a leaf node, and nodes $2u$ and $2u+1$ become new leaf nodes. This tree will grow for n days, ultimately forming a tree with $\left(2^{n+1}-1\right)$ nodes, of which 2^n nodes are leaf nodes.

During these n days, every afternoon, Little Q can choose one of the branches that have already grown to "prune", modifying its length to any positive integer (possibly less than or greater than the original one), or he can choose not to prune any branches at all.

Little Q hopes that after the tree has fully grown, the sum of the lengths of the branches on the simple path from the root node to each leaf node will be pairwise different. You need to find the minimum number of branches that need to be pruned, or indicate that it is impossible to achieve this.

Input

The first line of the input contains an integer T $(1 \le T \le 1\,000)$ indicating the number of test cases. For each test case:

The first line contains an integer n $(1 \le n \le 10)$, indicating the number of days the tree grows.

The second line contains $\left(2^{n+1}-2\right)$ integers $a_2, a_3, \ldots, a_{2^{n+1}-1}$ $(1 \le a_i \le 10^8)$, indicating the length of the branches.

It is guaranteed that the number of test cases with n greater than 7 does not exceed 10.

Output

For each test case, output a line containing an integer, indicating the minimum number of branches that need to be pruned if it is possible to meet the requirement, or -1 otherwise.

Example

standard input	standard output

3
2
1 2 4 3 2 1
2
1 2 3 3 2 1
2
1 2 3 3 1 1
	1
	2
	-1

Problem J. Make Them Believe

Time limit:	1 second
Memory limit:	1024 megabytes

The 2024 League of Legends World Championship culminated in an epic clash between two titans of professional League of Legends on November 2, 2024. Bilibili Gaming, after dominating the LPL with two domestic titles in 2024, undoubtedly carried the hopes of China's premier league that had endured three years without lifting the Summoner's Cup. In their way stood T1, who had qualified as LCK's fourth seed but proved their championship mettle through their tournament run. And there he was — Faker, the man revered as "the highest mountain and the longest river" — still standing at the center of LoL's greatest stage as if time itself had learned to bow before him. With four crowns already weighing heavy in his legacy, Faker somehow made the hunt for a fifth feel less like ambition and more like destiny. Now, beneath the towering dome of London's O2 Arena, the stakes were immense for both teams: BLG aimed to end the LPL's drought and cement their remarkable 2024 campaign, while T1 sought to defend their title and further establish their dynasty.

In an electrifying best-of-five series that pushed both teams to their limits, T1 emerged victorious over BLG with a hard-fought 3-2 triumph, clinching their fifth world championship. True to the tournament's official slogan "Make Them Believe", T1 mounted an incredible comeback from a 1-2 deficit, reaffirming once again why they remain a seemingly insurmountable obstacle for LPL teams in the international arena.

As we trace back through the journey of these two finalist teams, it all started in that fateful knockout stage, where eight teams competed in a single-elimination tournament: LNG Esports, Weibo Gaming, Hanwha Life Esports, Bilibili Gaming, Top Esports, T1, Gen.G, and FlyQuest. In an eight-team single-elimination tournament format, teams were seeded into the bracket based on their performance in the Swiss stage to face off in head-to-head matches where the loser is immediately eliminated. The tournament progresses through three rounds: Quarterfinals with eight teams, Semifinals with four teams, and the Finals with two teams, ultimately crowning a single champion.

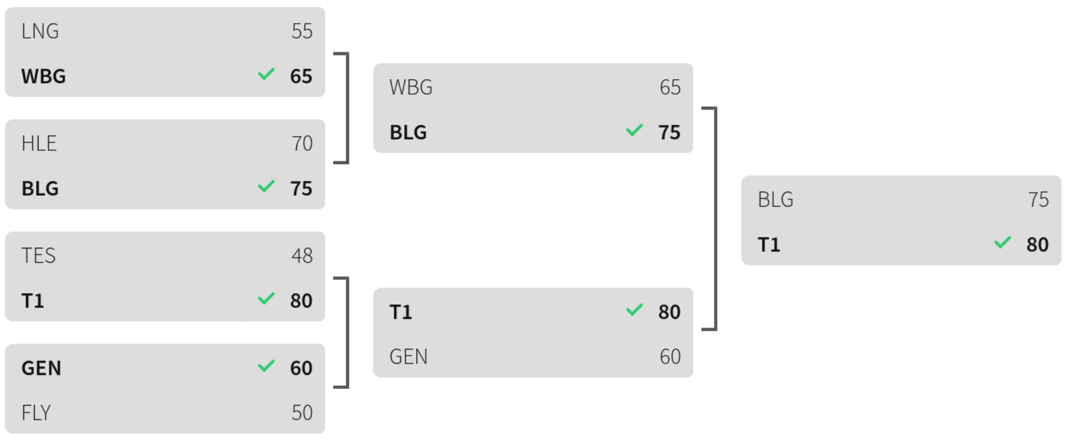


Figure: Eight-Team Single-Elimination Tournament Bracket

Although Worlds 2024 has irretrievably passed into history, we can still craft our own version of the story. For this problem, you will simulate a hypothetical knockout stage using the same eight-team single-elimination format. You will be given the names and the strength values of the eight teams in order from top to bottom of the Quarterfinals bracket (shown in the leftmost column in Figure), and both the names and the strength values are guaranteed to be pairwise different. When two teams compete, the team with the higher strength value always wins, while the losing team is eliminated. For two teams competing in the Finals, the team that wins is designated as the champion, while the other team is designated as the runner-up.

Your task is to find the champion team and the runner-up team in the hypothetical knockout stage.

Input

The input contains eight lines, the i-th of which contains a string S_i ($1 \le |S_i| \le 3$) and an integer t_i ($1 \le t_i \le 100$), indicating the name and the strength value of the i-th team from top to bottom of the Quarterfinals bracket.

It is guaranteed that all team names contain only uppercase English letters or decimal digits, and both the names and the strength values are pairwise different.

Output

Output "`A beats B`" (without quotes) in one line, where `A` is the name of the champion team and `B` is the name of the runner-up team.

Examples

standard input	standard output
LNG 55 WBG 65 HLE 70 BLG 75 TES 48 T1 80 GEN 60 FLY 50	T1 beats BLG
LNG 55 WBG 65 HLE 70 BLG 81 TES 48 T1 80 GEN 60 FLY 50	BLG beats T1

Note

In the second sample case, since we intentionally increase BLG's strength value in the hypothetical knockout stage, please note that the outcome of the Finals, sadly, differs from the actual Worlds 2024.

Problem K. Fragile Pinball

Time limit:	3 seconds
Memory limit:	1024 megabytes

Given a convex polygon with n edges, there is a very small pinball, which can be seen as a point, that starts from a certain position inside or on the polygon and moves straight at a positive constant speed.

You can activate an edge of the convex polygon at any moment for an instant. If the pinball is exactly on an activated edge (including the endpoints) at that time, it will be reflected by this edge. That is, the angle of incidence equals the angle of reflection with respect to the line that the edge lies on. Since the edges are fragile, each edge can be activated at most once.

It is important to note that if the pinball is simultaneously on two edges, you cannot activate both edges at the same time. However, you can activate the two edges in quick succession. In this case, the pinball will be reflected twice, and the moving direction of the pinball also changes twice while its position and speed remain unchanged. You can also choose to activate only one of the edges or not to activate any.

Since the pinball is also fragile, it can only bear at most k reflections. You need to find the maximum distance that the pinball can travel within the convex polygon for each $k = 0, 1, 2, \ldots, n$.

Input

The first line contains an integer n ($3 \le n \le 6$), indicating the number of vertices of the convex polygon.

For the following n lines, the i-th line contains two integers x_i and y_i ($-100 \le x_i, y_i \le 100$), indicating the coordinates of the i-th vertex of the convex polygon.

It is guaranteed that the n vertices are given in counter-clockwise order, and any three of them are not collinear.

Output

Output $(n+1)$ lines, the i-th of which contains a real number, indicating the length of the longest journey of the pinball with at most $i-1$ reflections.

Your answer will be considered correct if its absolute or relative error does not exceed 10^{-6}. Formally speaking, suppose that your output is a and the jury's answer is b, your output is accepted if and only if $\frac{|a-b|}{\max(1,|b|)} \le 10^{-6}$.

Examples

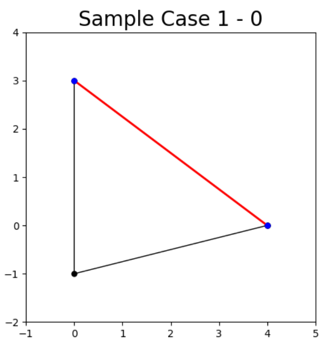
standard input	standard output

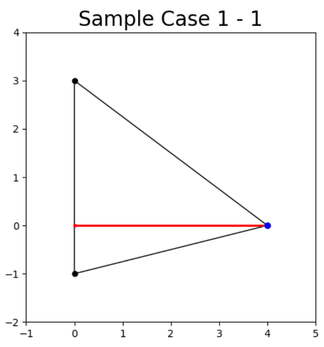
3
4 0
0 3
0 -1
	5.000000000000000000
	8.000000000000000000
	8.868185038797563409
	12.210024810881955830

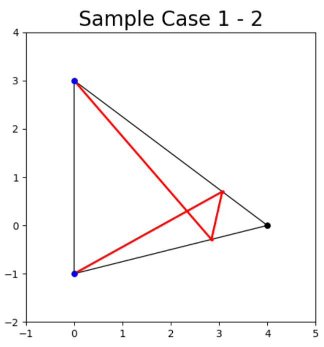
3
4 0
0 3
0 2
	5.000000000000000000
	5.366563145999495272
	6.111919138499425171
	6.782203304416628317

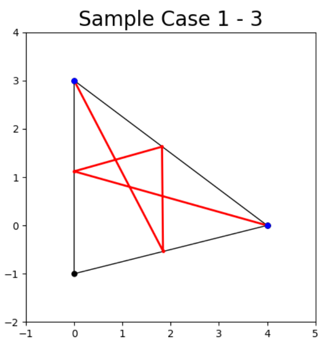
Note

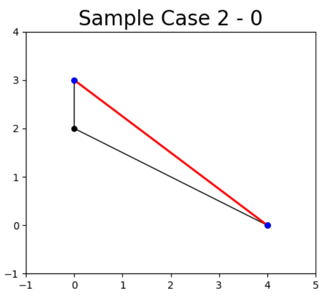
The sample cases are shown in the figures below.

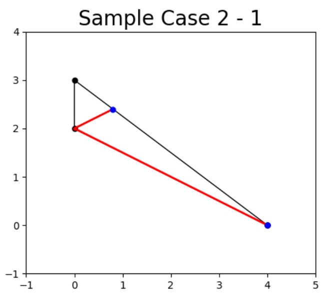

Sample Case 1 - 0

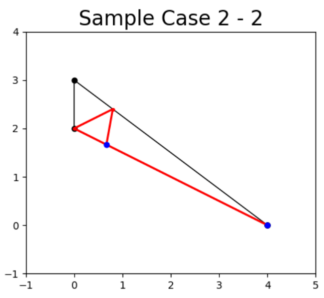

Sample Case 1 - 1

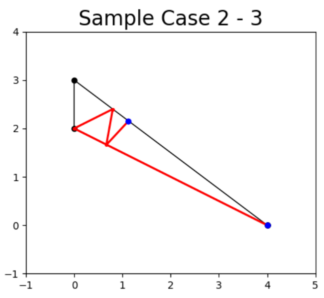

Sample Case 1 - 2


Sample Case 1 - 3


Sample Case 2 - 0


Sample Case 2 - 1


Sample Case 2 - 2

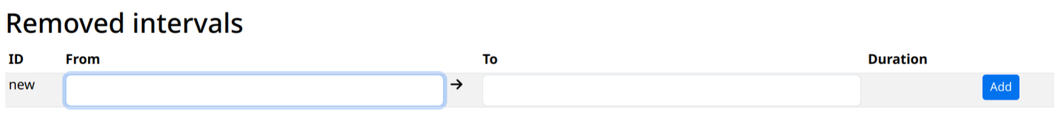

Sample Case 2 - 3

Problem L. The Grand Contest

Time limit:	3 seconds
Memory limit:	1024 megabytes

In the Grand Contest, there are many problems and a very long duration. Team 1 and team 2 participated in this contest, which had a total of n submissions. Each submission can either get a correct or incorrect verdict. A problem is considered solved by a team when the submission from that team is correct.

After the contest ends, teams are ranked by the most problems solved. Teams that solve the same number of problems are ranked by the least total time. The total time is the sum of the time consumed for each solved problem. The time consumed for a solved problem is the time elapsed from the beginning of the contest to the submission of the first correct submission, plus p penalty minutes for each previously incorrect submission for that problem. There is no time consumed for a problem that is not solved. In case of a tie, team 1 ranks before team 2.



The judges of the Grand Contest will test a feature named “remove interval”, which allows selecting two integers L and R ($0 \le L < R$) and remove the time interval $[L, R]$ in minutes from the contest such that:

- All submissions with submission time less than L minutes are unaffected;
- All submissions with submission time in $[L, R]$ minutes have their submission time changed to L minutes from the beginning;
- All submissions with submission time more than R minutes have their submission time reduced by $(R - L)$ minutes.

It is important to note that the above changes will only affect the submission times of certain submissions but will not affect the relative order of the submissions.

You need to find the shortest time interval $[L, R]$ in minutes such that after removing this interval from the contest, the rankings of the two teams will change, or indicate that such an interval does not exist. If there are multiple such intervals, you need to find the one with the smallest L.

Input

The first line of the input contains an integer T ($1 \le T \le 4 \times 10^5$), indicating the number of test cases. For each test case:

The first line contains two integers n ($1 \le n \le 4 \times 10^5$) and p ($1 \le p \le 10^{12}$), indicating the number of submissions and the penalty minutes.

Then n lines follow, describing the submissions during the contest in order. Each line contains four integers a ($a \in \{1, 2\}$), b ($1 \le b \le 10^9$), c ($1 \le c \le 10^{12}$), and d ($d \in \{0, 1\}$), indicating team a submitted on problem b at c minutes from the beginning of the contest and got verdict d. If $d = 1$, the submission is correct and solves the problem; otherwise, the submission is incorrect. It is guaranteed that the submission time is non-decreasing.

It is guaranteed that the sum of n over all test cases does not exceed 4×10^5.

Output

For each test case, output a line. If there exists a feasible time interval in minutes, output two integers L and R, indicating the shortest interval with the smallest L. Otherwise, output -1.

Example

standard input	standard output

2
6 20
1 1 60 0
2 2 60 0
2 2 120 1
1 2 180 1
1 2 180 0
2 2 300 1
2 20
1 1 300 1
2 2 300 1
	120 160
	-1

Problem M. Obliviate, Then Reincarnate

Time limit:	2 seconds
Memory limit:	1024 megabytes

Leafing through his journal of musings, Mu discovered a note forgotten between the yellowed pages. The note, inspired by the faraway constellation Vulpecula, resembled a mantra of self-encouragement which read, "*Dreams are like the stars, distant and sometimes insignificant, yet enough to spark up the darkest night.*"

In time, Mu had eventually glimpsed the veiled face of dreams. Through years of growth, he had emerged more seasoned, perhaps more scarred, only to accept that stars and dreams alike, constrained by the fatalism that equally governs everything, are destined to fade into oblivion. Pursuit of dreams had crumbled, doubt about reality had shadowed, yet in his refusal to succumb to pessimism, Mu had waited, and was waiting still, harboring the belief that broken dreams, like stellar ashes in the pageantry of samsara, would be reconstructed and kindled anew.

He continued reviewing the journal, with the boundless cosmos reigning overhead. Indeed, the infinite had always held more promise and mystery than the finite — be it in the realm of spacetime, in the nebula of dreams, or even in the world of mathematics. As if on cue, his fingers paused at a page where Hilbert's Hotel Paradox, the classic paradigm of infinite sets, had once captured his contemplation. Amid the jottings, Mu had envisioned an upgraded version of the infinite hotel — the Celestial Hilbert's Hotel — and cast himself as its manager, leaving an intriguing problem for programmers like you to solve.

Hilbert's Infinite Hotel

As a marvel of deep space, the Celestial Hilbert's Hotel features infinitely many rooms. Each room is assigned a unique integer (can be positive, negative, or zero) as its number, and all the room numbers constitute $\mathbb{Z}$, the set of all integers.

There are n floors in the hotel, where rooms are distributed based on the non-negative remainder of their number when divided by n. Specifically, room x and room y are on the same floor if and only if the room numbers have the common remainder when divided by n, expressed as a congruence equation $x \equiv y \pmod{n}$.

Like any typical Hilbert's Hotel, the version in question also operates with certain relocation instructions to vacate occupied rooms. Mu has designed m relocation instructions. Each instruction, denoted as a pair (a, b), directs all guests on the same floor as room a to simultaneously move to new rooms by adding b to their previous room numbers. When necessary, Mu can select one of these m instructions and broadcast it to the corresponding floor. You might wonder about the case where guests are asked to move into already occupied rooms. In this scenario, the relocating guests will still move in, accompanying the original occupants forever.

However, due to the infinity of the Celestial Hilbert's Hotel, it is hard for Mu to track a guest's potential locations after multiple instructions, resulting in management inefficiency. Thus, Mu proposes q queries, each concerning a guest who initially occupies room x. Supposing that the relocation instructions can be applied any number of times in any order, with both the exact number of times and order being unspecified, for each query, your task is to determine whether the set of all room numbers possibly reachable by the guest concerned is an infinite set. More formally, let S_k be the set of room numbers possibly reachable by the guest concerned after at most k instructions, then you need to determine whether for all integers t, such k exists that $|S_k| > t$ can be achieved, where $|S_k|$ denotes the size of S_k.

Input

The first line contains three integers n, m, and q ($1 \le n, m, q \le 5 \times 10^5$), indicating the number of floors, relocation instructions, and queries, respectively.

Then m lines follow, each of which contains two integers a and b ($-10^9 \le a, b \le 10^9$), describing a relocation instruction (a, b).

Then q lines follow, each of which contains an integer x ($-10^9 \le x \le 10^9$), indicating a query concerning the guest who initially occupies room x.

Output

For each query, output "`Yes`" (without quotes) in one line if the set of all room numbers possibly reachable by the guest concerned is an infinite set, or otherwise output "`No`" (without quotes) in one line.

Examples

standard input	standard output

```
3 2 3
1 1
-1 3
1
2
3
```
	```
	Yes
	Yes
	No
	```

```
3 2 3
1 1
-1 0
1
2
3
```
	```
	No
	No
	No
	```