Problem A. Crazy Yesterday

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

The secret of success is to eat what you like and let the food fight it out inside.

— Mark Twain, *More Maxims of Mark*

Geopelia and *Gino* are best friends and can always read each other's minds.

One day, *Gino* went out early and left a note, which was found by *Geopelia* later.

"You have forgotten something – the yesterday...Oh, I am hungry..."

Geopelia guessed his intention at once. But she still wants to challenge you:

If you are told what day of the week it is today, answer what day of the week it was *yesterday*.

Input

The first line contains a single integer t ($1 \le t \le 2 \times 10^5$), denoting the number of test cases.

The i-th of the next t lines contains a single integer w ($1 \le w \le 7$), denoting what day of the week it is today.

Note that $w = 1$ denotes *Monday*, $w = 2$ denotes *Tuesday*, etc.

Output

For each test case, output a single integer y ($1 \le y \le 7$), denoting what day of the week it was yesterday.

Example

standard input	standard output
2 5 1	4 7

Note

In the first test case, today is *Friday*, so yesterday was *Thursday*.

In the second test case, today is *Monday*, so yesterday was *Sunday*.

This page is intentionally left blank

Problem B. Solo Leveling

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

It is not in the stars to hold our destiny but in ourselves.

— William Shakespeare, *Julius Caesar*

You are playing a computer game. To pass the current level, you need to kill all monsters solo.

As a *Player*, your power is quantified in two properties A and B.

In this level, there are n monsters. The i-th monster has two properties a_i and b_i. To kill the i-th monster, you need more power correspondingly, i.e. $A \geq a_i$ and $B \geq b_i$.

Note that different monsters can have the **same** properties.

The Designer limits your initial properties and gives you ways below to promote them.

Initially, your properties are $A = 10, B = 10$. If you killed the i-th monster, you'll get exactly c_i distributable points. Each point can promote **exactly one** property by 1 forever, i.e. $A := A + 1$ or $B := B + 1$. You can use these points at any time, but each point can be used only **once**.

Now you can kill the monsters in any order, determine whether you can kill them all.

Input

The first line contains a single integer t $(1 \leq t \leq 10^3)$, denoting the number of test cases.

The first line of each test case contains a single integer n $(1 \leq n \leq 10^3)$, denoting the number of monsters.

The i-th of the next n lines contains three integers a_i, b_i, c_i $(1 \leq a_i, b_i, c_i \leq 10^9)$, denoting the two properties of the i-th monster and the distributable points you will get after killing the i-th monster.

It's guaranteed that the sum of n over all test cases doesn't exceed 4×10^3.

Output

For each test case, print `Yes` if you can kill monsters in some order. Otherwise, print `No`.

You may print each letter in any case. For example, `YES`, `yes`, `Yes` will all be recognized as positive answer, `NO`, `no`, `nO` will all be recognized as negative answer.

Example

standard input	standard output
4	No
1	Yes
11 11 2	Yes
3	Yes
1 1 1	
2 2 2	
3 3 3	
4	
8 8 1	
11 10 1	
11 10 1	
11 12 1	
4	
1 5 10	
20 32 1	
10 22 10	
20 10 12	

Note

In the first test case, there is only one monster. If you want to kill it, you must at least promote your properties to $A = 11, B = 11$, but obviously you can't. You can not get c_1 before killing it.

In the second test case, all the monsters are weaker than your initial properties $A = 10, B = 10$, so you can kill them all in any order.

In the third test case, you can kill the monsters in the order of the input.

In the fourth test case, one possible order is:

1. Kill the 1-st monster, and use all the points you get to promote A. After that, $A = 20, B = 10$.
2. Kill the 4-th monster, and use all the points you get to promote B. After that, $A = 20, B = 22$.
3. Kill the 3-rd monster, and use all the points you get to promote B. After that, $A = 20, B = 32$.
4. Kill the 2-nd monster, and you can pass the level.

Problem C. Chain Reaction

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

Our doubts are traitors, and make us lose the good we oft might win, by fearing to attempt.

— William Shakespeare, *Measure for Measure*

Feeling a little bored, *WX* decided to play a game called "Chain Reaction".

The game is based on a classic question "Toggle Lamps". There are n lamps in total. Initially, all of them are turned off.

Also, there are n buttons. If you push the i-th button, then all lamps x such that x is a **multiple** of i will be toggled.

You have to push some button according to the following rules:

1. You need to push at least one button;
2. A button can be pushed **only once**;
3. You are given m pairs (u_i, v_i), if you push the button u_i, then you have to push the button v_i (at any moment, not necessarily after pushing the button u_i). Note that if you push the button v_i, then you don't have to push the button u_i for the given pair (u_i, v_i).

Find a way to push buttons such that **at most** $\lfloor\sqrt{n}\rfloor$ lamps[†] are on, or print -1 if it is impossible.

[†] $\lfloor x \rfloor$ denotes the round down operation.

Input

The first line contains a single integer t ($1 \le t \le 10^4$), denoting the number of test cases.

The first line of each test case contains two integers n, m ($1 \le n \le 2 \times 10^5, 0 \le m \le 2 \times 10^5$), denoting the number of lamps and the number of pairs.

The i-th of the next m lines contains two integers u_i, v_i ($1 \le u_i, v_i \le n, u_i \ne v_i$). If you push the button u_i, then you have to push the button v_i. It's guaranteed that the pairs (u_i, v_i) are distinct.

It's guaranteed that the sum of n and the sum of m over all test cases doesn't exceed 2×10^5.

Output

For each test case, output a single line:

- If it is impossible, then output -1;
- Otherwise, first output an integer k, denoting the number of buttons you push. Then output k integers b_i, denoting the buttons you push. You can output in any order. Note that all b_i must be distinct, and at last there will be at most $\lfloor\sqrt{n}\rfloor$ lamps that are on.

Example

standard input	standard output

1
4 2
1 2
4 3
	2 4 3

Note

In the first test case, there are 4 lamps and 2 pairs in total. $\lfloor\sqrt{4}\rfloor = 2$.

One possible way is:

1. Push the button 4. After that, the lamp 4 will be toggled on;
2. Push the button 3. After that, the lamp 3 will be toggled on (satisfy the 2-th pair).

Another possible way is just to push the button 2. After that, the lamp $2, 4$ will be toggled on (4 is a multiple of 2). Note that you don't need to push the button 1.

These are not the only possible ways.

Problem D. XOR Pairing

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

Excess is just as bad as deficiency.

— Confucius, *The Analects*

OP, a senior high school student, had fallen in love with the **bitwise XOR operation**.

So is there anything that can be done for OP? Don't worry, CY has a wonderful idea.

CY gives OP an array a consisting of exactly n numbers and a non-negative integer k. And then, CY throws him a question:

How many pairs (i, j) $(1 \le i < j \le n)$ satisfy $a_i \oplus a_j = k$? Here $\oplus$ denotes the **bitwise XOR operation**[†].

[†] **Bitwise XOR operation** is a binary operation that takes two bit patterns of equal length and performs the *logical exclusive OR operation* on each pair of corresponding bits. The result in each position is `1` if only one of the bits is `1`, but will be `0` if both are `0` or both are `1`. In this, we perform the comparison of two bits, being `1` if the two bits are different, and `0` if they are the same. For example:

```
    0101 (decimal 5)
XOR 0011 (decimal 3)
  = 0110 (decimal 6)
```

Input

The first line contains a single integer t $(1 \le t \le 10^4)$, denoting the number of test cases.

The first line of each test case contains two integers n, k $(1 \le n \le 2 \times 10^5, 0 \le k \le 10^9)$.

The second line contains exactly n integers $a_1, a_2, \ldots, a_n$ $(0 \le a_i \le 10^9)$.

It's guaranteed that the sum of n over all test cases doesn't exceed 2×10^5.

Output

For each test case, output a single integer, denoting the number of pairs (i, j) satisfying $a_i \oplus a_j = k$.

Example

standard input	standard output

2
6 1
1 1 4 5 1 4
7 0
1 9 1 9 8 1 0
	2
	4

Note

In the first test case, only two pairs $(3, 4)$, $(4, 6)$ satisfy the constraint: $a_3 \oplus a_4 = a_4 \oplus a_6 = 1$.

In the second test case, there are four pairs $(1, 3)$, $(1, 6)$, $(2, 4)$, $(3, 6)$ satisfy the constraint: $a_1 \oplus a_3 = a_1 \oplus a_6 = a_2 \oplus a_4 = a_3 \oplus a_6 = 0$.

This page is intentionally left blank

Problem E. Cyber Hide-and-Seek

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3.9 seconds
Memory limit:	256 megabytes

The real is the rational, and the rational is the real.
— G.W.F.Hegel, *Grundliniender Philosophiedes Rechts*

This is an interactive problem.

One day, you wake up to find *Miku* in front of you. Instead of singing, she wants to play a game with you!

This game is played on a tree with n nodes, which is rooted at node 1. A tree is a connected graph without cycles.

Miku has hidden a *fufu* at node x. You need to ask *Miku* the following two types of queries to find the location of it:

- 1 u ($1 \le u \le n$). *Miku* will tell you the distance between node u and node x. Here, the distance between two nodes denotes the number of edges in the shortest path between them.
- 2 v ($1 \le v \le n$). *Miku* will tell you the second node on the shortest path from node v to node x. However, when executing this query, you need to ensure that v **is an ancestor of** $x^\dagger$, or you will immediately receive a `Wrong Answer` verdict due to *Miku's* unhappiness.

† Node a is an ancestor of node b if and only if $a \neq b$, and the shortest path from node 1 to node b passes through a. Note that in this problem, node a is **not an ancestor of a itself.**

Can you find x within 39 queries? If you do, *Miku* will give you a *fufu* as a reward!

Input

The first line contains a single integer n ($2 \le n \le 3.9 \times 10^5$), denoting the number of nodes in the tree.

The i-th of the next $n-1$ lines contains two integers u_i and v_i ($1 \le u_i, v_i \le n$), denoting the i-th edge on the tree.

It is guaranteed that the input edges form a tree.

Interaction Protocol

To make a query, you can output one of the following two formats:

- 1 u ($1 \le u \le n$), or
- 2 v ($1 \le v \le n$).

After making a query, you should read the corresponding response, which represents the distance from u to x or the second node on the path from v to x, depending on your query.

If the response you read is -1, it means that you may have exceeded the number of allowed queries, made an invalid query, or violated the restriction for the second type of query. In this case, you should immediately exit the program and receive a `Wrong Answer` verdict; otherwise, you may receive **a random verdict** other than Accepted.

After printing a query, do not forget to output end of line and flush the output. Otherwise, you will get `Runtime Error`. To do this, use:

- `fflush(stdout)` or `cout.flush()` in C++;
- `System.out.flush()` in Java;
- `stdout.flush()` in Python.

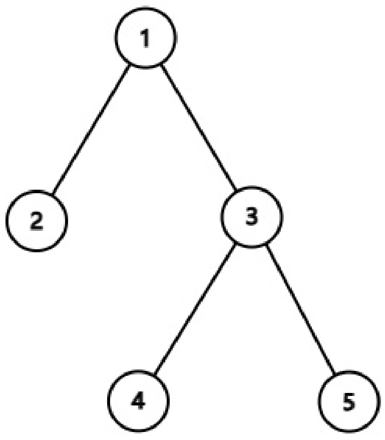
To print the answer, please output `"! x"` (Without quotes). Note that giving this answer is not counted towards the limit of 39 queries.

Example

standard input	standard output
5	
1 2	
1 3	
3 4	
3 5	
	1 2
3	
	2 3
4	
	! 4

Note

In the example, the hidden *fufu* is at node $x = 4$.



The tree in the example

You can make the following queries:

1. Ask the distance between node 2 and x, and get the reply 3. It can be inferred that $x = 4$ or 5;
2. Ask the second node from node 3 to x, and get the reply 4. Note that node 3 is the parent node of node x, so the second node is node x. Therefore, $x = 4$;

Output the answer 4, and you can get the reward.

Note that this is **not** the only way to get the answer.

Problem F. Double Holding

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

Life is a balance of holding on and letting go.

— Molana Jalaluddin Rumi, *Make Yourself Free*

RN is playing a music game called `"Mu5Ed4sH"` on her computer.

Hold is a type of note with a time range $[l_i, r_i]$, and you need to keep one finger on it within the time range.

To simplify the statement, let's assume that *Hold* is the only type of note in the game.

In this game, the notes will appear on two tracks, forming two sequences of time ranges s and t. If two *Holds* appear at the same time, then you need to use two fingers together.

However, if you use two fingers together for t seconds, then you will lose exactly t energy.

Given the initial energy E you have, determine whether the energy is enough to play the entire game. If it is enough, output the energy you will have left.

Input

The first line contains two integers n, m, E $(1 \le n+m \le 10^5, \min(n,m) \ge 1, 0 \le E \le 10^9)$, denoting the number of time ranges in sequence s and t, and the initial energy you have.

The i-th of the next n lines contains two integers l_{si}, r_{si} $(0 \le l_{si} < r_{si} \le 10^9)$, denoting the time range of the i-th *Hold* in sequence s.

The i-th of the next m lines contains two integers l_{ti}, r_{ti} $(0 \le l_{ti} < r_{ti} \le 10^9)$, denoting the time range of the i-th *Hold* in sequence t.

It's guaranteed that **no** two time ranges will **overlap** in the **same sequence**.

Output

If the energy is not enough, print -1; Otherwise, print a single integer x, denoting the energy you will have left.

Examples

standard input	standard output

3 2 616
5 7
0 4
8 9
2 6
7 10
	612

1 1 998244353
0 1000000000
0 1000000000
	-1

Note

In the first test case, there are 3 time ranges in the first track and 2 time ranges in the second track.

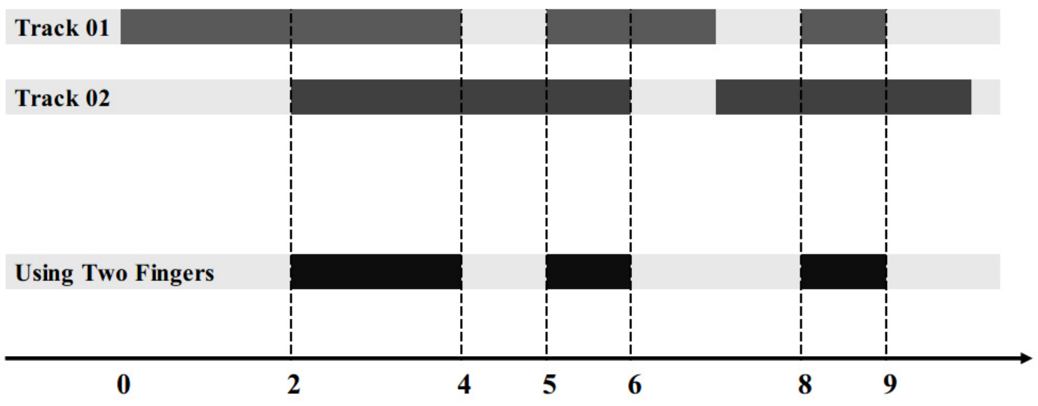


Figure of Test Case 1

As the figure shows, during $[2, 4]$, $[5, 6]$, $[8, 9]$, you need to use two fingers. And you will lose $(4 - 2) + (6 - 5) + (9 - 8) = 4$ energy.

It can be shown that the energy is enough, and the energy you will have left is $616 - 4 = 612$.

Problem G. Color Contagion

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

I love the lotus because while growing from mud, it is unstained.

— Zhou Dunyi, *On the Love of the Lotus*

TN was gifted a tree of n vertices with the root in the vertex 1. A tree is a connected undirected graph without cycles.

Initially, the root has already been colored, and he wanted to color all the vertices using the operation below.

During one operation, *TN* can choose one uncolored vertex whose parent has already been colored and color it.

It can be shown that after $n-1$ operations, all the vertices will be colored.

Let p_i be the chosen vertex of the i-th operation. If *TN* can color all vertices in this way, then the array p is called *Valid*.

Count the number of different *Valid* arrays modulo 998244353.

Input

The first line contains a single integer t ($1 \le t \le 10^4$), denoting the number of test cases.

The first line of each test case contains a single integer n ($1 \le n \le 10^5$).

The i-th of the next $n-1$ lines contains two integers u_i, v_i ($1 \le u_i, v_i \le n$), denoting the indices of vertices connected by the i-th edge.

It's guaranteed that the sum of n over all test cases doesn't exceed 2×10^5.

Output

For each test case, output a single integer, denoting the number of different *Valid* arrays modulo 998244353.

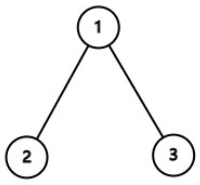
Example

standard input	standard output

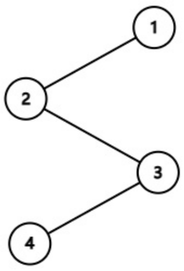
4
3
1 2
1 3
4
1 2
2 3
3 4
1
6
1 2
1 3
2 4
2 5
3 6
	2
	1
	1
	20

Note

In the first test case, there are two arrays that satisfy the condition: $p_1 = \{2, 3\}, p_2 = \{3, 2\}$, which means *TN* can color the vertex 2 at first and then color the vertex 3, or he can color the vertex 3 at first and then color the vertex 2.



Case 1

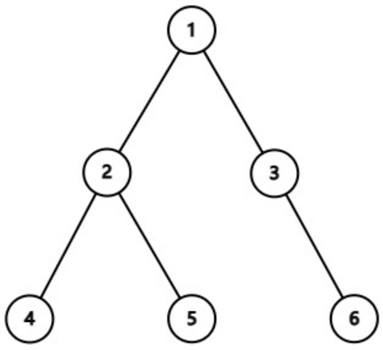


Case 2

Case 3

In the second test case, it can be shown that only one array satisfies the condition: $p = \{2, 3, 4\}$, which means *TN* can color the vertex 2 at first, then color the vertex 3, and at last, color the vertex 4.

In the third test case, note that the empty array is also *Valid*.



Case 4

Problem H. Seeking Allies

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	256 megabytes

In the middle of difficulty lies opportunity. We make alliances with the hope of avoiding bloodshed.

— Albert Einstein and Sun Tzu, *The Art of War*

A shadow falls, bringing the end to the entire world. To struggle for survival, people decided to seek allies.

There are n people standing in a row, who are initially unfamiliar with each other.

Also, there are d *conditions*, the i-th of them restricting that p_i is familiar with q_i.

Now, it's your decision time. Let's define that:

Your one *operation* is to choose two integers i, j $(1 \le i, j \le n, i \ne j)$, such that the i-th person and the j-th person were not familiar before, and make them familiar.

Note that if a is familiar with b and b is familiar with c, then a is familiar with c.

At time i $(1 \le i \le d)$, you can do at most i *operations*, but you need to satisfy the *conditions* from 1 to i (inclusive). After the last *operation* of each time i, you need to calculate the maximal number of people that one person can be familiar with.

Note that the *operations* of each i are **independent**. It means that before you start your first *operation* at each time i, you need to assume that the n people are unfamiliar with each other.

Input

The first line contains a single integer t $(1 \le t \le 10^4)$, denoting the number of test cases.

The first line of each test case contains two integers n, d $(2 \le n \le 10^5, 1 \le d \le n-1)$.

The i-th of the next d lines contains two integers p_i, q_i $(1 \le p_i, q_i \le n, p_i \ne q_i)$, denoting the i-th *condition*.

It's guaranteed that the sum of n over all test cases doesn't exceed 2×10^5.

Output

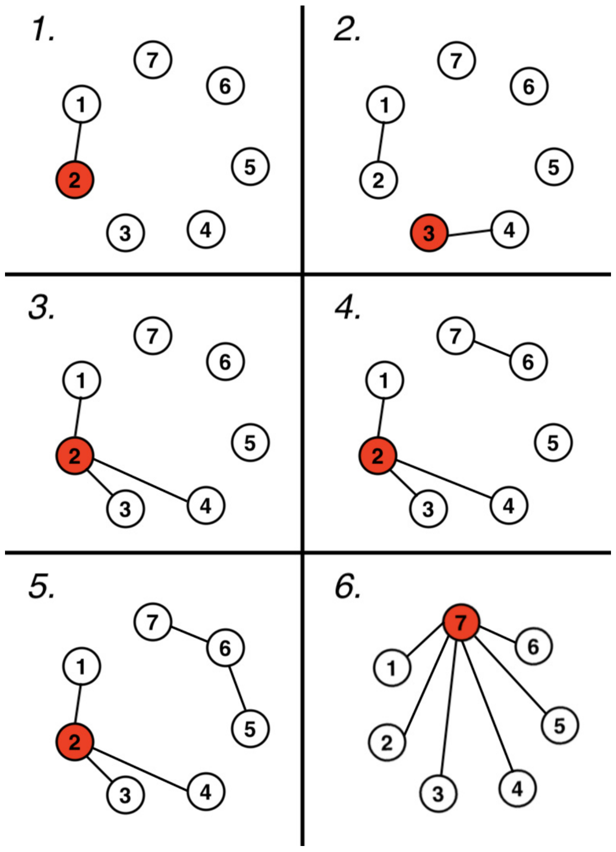
For each test case, output d lines, the i-th of them contains one integer x, denoting the maximal number of people that one person can be familiar with at time i.

Example

standard input	standard output
2	1
7 6	1
1 2	3
3 4	3
2 4	3
7 6	6
6 5	1
1 7	2
10 8	3
1 2	4
2 3	5
3 4	5
1 4	6
6 7	8
8 9	
8 10	
1 4	

Note

The explanation for the first test case:



Visual explanation

In this explanation, the circles and the numbers in them denote a person with the corresponding number. The line denotes that you decided to make two connected people familiar. The person marked with red has the maximal number of people that one person can be familiar with.

These are **not** the only correct ways.

Problem I. Aeroplane Chess

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

In every real man a child is hidden that wants to play.

— Friedrich Nietzsche, *Thus Spoke Zarathustra*

On a day that could not be more ordinary, *KP* unexpectedly received an anonymous birthday gift — an aeroplane chess!

The gift evoked memories of his childhood, but after several hours of playing, he had a doubt:

During the *Home Zone Backtrack*, if the chess is on the i-th cell, what is the expected number of rolls required to reach the end?

Formally, assume that the chess is placed on an axis, where the point `0` denotes the end.

During one operation, you will randomly choose a number y $(1 \le y \le n)$. Assume that the point of the chess is x:

- if $y < x$, then $x := x - y$;
- if $y > x$, then $x := y - x$;
- Otherwise, the game ends.

Now you need to process q queries, each query consists of a single integer x, denoting the point of the chess.

For each query, you need to output the expected number of operations needed to reach the end of the game.

It can be shown that the answer can be represented as P/Q, where P, Q are coprime integers and $Q \not\equiv 0 \pmod{998244353}$. Print the value of $P \times Q^{-1} \pmod{998244353}$.

Input

The first line contains two integers n, q $(1 \le n \le 500, 1 \le q \le 2 \times 10^5)$.

The i-th of the next q lines contains a single integer x $(1 \le x \le 10^6)$, denoting the point of the chess.

Output

For each query, output a single integer, denoting the answer to the query modulo 998244353.

Example

standard input	standard output

2 2
1
114514
	2
	64376993

Note

In the first test case, the chess is on point `1`, and the following will happen:

1. You randomly choose 1 for possibility $\frac{1}{2}$. After that, $1 = x$, and you reach the end.

2. You randomly choose 2 for possibility $\frac{1}{2}$. After that, $2 > x$, so $x := 2 - x = 1$. Then you randomly choose 1 for possibility $\frac{1}{2}$. After that, $1 = x$, and you reach the end.

3. ...

It can be shown that the expected number of operations to reach the end equals:
$E = 1 \times \dfrac{1}{2} + 2 \times \dfrac{1}{2^2} + 3 \times \dfrac{1}{2^3} + \ldots = 2.$
In the second test case, note that you need to print the answer modulo 998244353.

Problem J. Shifting Tournament

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

Destiny is spinning like the wheels of a waterwheel. Those who were high yesterday are inferior today.

— Miguel de Cervantes Saavedra, *Don Quijote de la Mancha*

2^k teams participate in a special tournament called *Shifting Tournament*.

The process of *Shifting Tournament* is the same as the standard tournament, which consists of $2^k - 1$ games. They are held as follows:

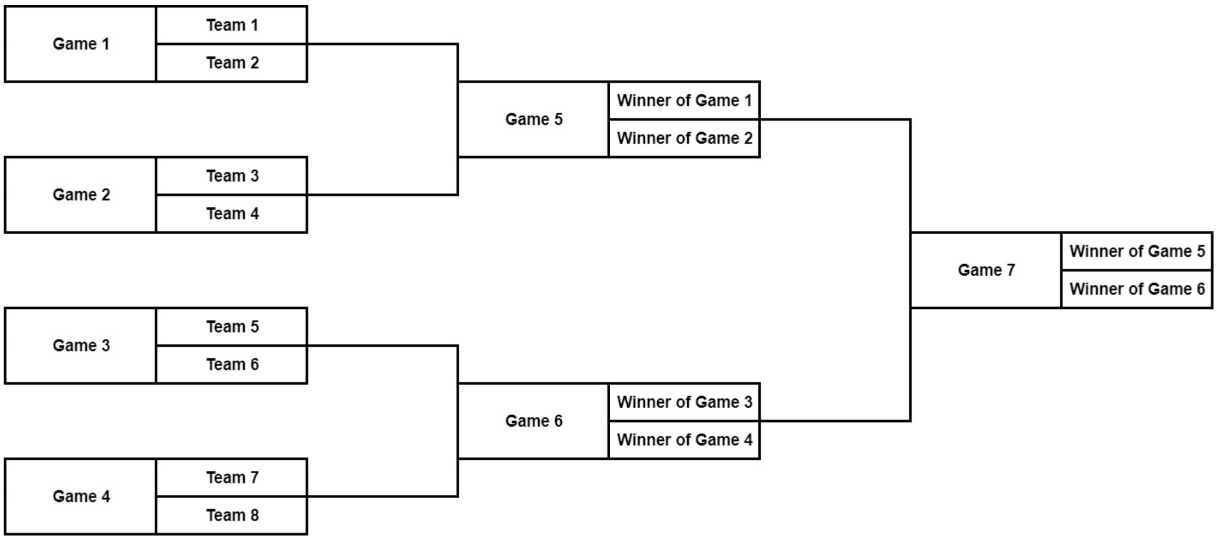
During the first round, all teams are split into pairs: team 1 against team 2, team 3 against team 4 (exactly in this order), and so on. Within the 2^{k-1} games, a team who loses the game will be eliminated, and it's guaranteed that each game results in elimination of one team (no ties). After that, 2^{k-1} teams remain.

During the second round, the remaining team continues according to the rules above. That means 2^{k-2} games will take place, and 2^{k-2} teams remain.

During the third round, ...

This process repeats until only one team remains, and the team is declared the *champion*. It's clear that there are k rounds exactly.

For example, the picture below describes the chronological order of games with $k = 3$.



visual picture of $k = 3$

The specialty of the tournament is the elimination method.

Let the string s consisting of k characters describe the elimination method in each round as follows:

- if s_i is `0`, then during the i-th round, the team with lower index will be eliminated;
- if s_i is `1`, then during the i-th round, the team with higher index will be eliminated;
- if s_i is ?, then during the i-th round, the team with lower or higher index will be eliminated.

Note that during the same round, the elimination method is **the same**.

Let $F(s)$ be the number of *possible winners* described by the string s. A team i is a *possible winner* of the tournament if it is possible to replace every ? with either 0 or 1 in such a way that team i is the *champion*.

You are given the initial state of the string s, and you need to process q queries:

- p c — replace s_p with character c, and print $F(s)$ as the result of the query.

Since the answer can be huge, print it modulo 998244353.

Input

The first line contains a single integer k $(1 \le k \le 10^5)$, denoting the number of rounds.

The second line contains a string consisting of k characters, denoting the initial state of the string s. Each character is either `0`, `1`, or ?.

The third line contains a single integer q $(1 \le q \le 10^5)$, denoting the number of queries.

The i-th of the next q lines contains an integer p and a character c $(1 \le p \le k; c$ is either `0`, `1`, or ?$)$, denoting the i-th query.

Output

For each query, print one integer, denoting $F(s) \bmod 998244353$.

Examples

standard input	standard output
3 0?1 2 2 1 3 ?	1 2
36 ???????????????????????????????????? 1 36 1	419430366

Note

In the first test case, there are exactly $2^3 = 8$ teams.

After the first query, $s =$ `011`. It's clear that there is only one *champion* — the team 2.

After the second query, $s =$ `01?`. In the third round, the remaining teams are team $2, 6$. If you replace the ? with 1, then team 2 will win. Otherwise, team 6 will win. So there are 2 *possible winners*.

In the second test case, note that the answer is huge, and you need to print it modulo 998244353.

Problem K. Uniform Dispersion

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

The universe is not required to be in perfect harmony with human fancy.

— Carl Sagan, *Broca's Brain*

OC likes the stars in the sky, which are scattered but are arranged in a certain pattern.

One day, OC encounters a map with n points. The i-th point is located at (x_i, y_i).

Just like the sky, he wants to make some partitions on it. But he is too busy, so he asks you for help.

Firstly, he chooses a lucky number k, which will be given to you.

Then, you need to draw k vertical lines and k horizontal lines, meeting with the following condition:

1. Assume that the i-th vertical line is $x = a_i$, the j-th horizontal line is $y = b_j$;
2. For all (x_i, y_i), $1 \le i \le n$, such that $x_i \notin a$ and $y_i \notin b$ (In other words, all the lines don't go through the points in the map);
3. The map is divided into $(k+1)^2$ regions by these lines. The number of points in each region is **the same**;
4. a_i and b_i can be **non-integers**.

Determine whether it is possible.

Input

The first line contains a single integer t ($1 \le t \le 10^4$), denoting the number of test cases.

The first line of each test case contains two integers n, k ($4 \le n \le 2 \times 10^5, 0 \le k \le 10^9$).

The i-th of the next n lines contains two integers x_i, y_i ($-10^9 \le x_i, y_i \le 10^9$), denoting the i-th point on the map.

It's guaranteed that the sum of n over all test cases doesn't exceed 2×10^5.

Output

For each test case, print `Yes` if it is possible. Otherwise, print `No`.

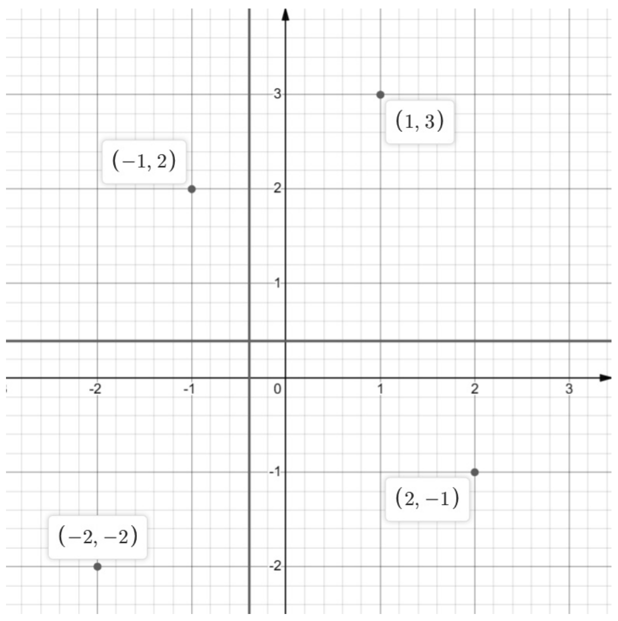
You may print each letter in any case. For example, `YES`, `yes`, `Yes` will all be recognized as positive answer, `NO`, `no`, `nO` will all be recognized as negative answer.

Example

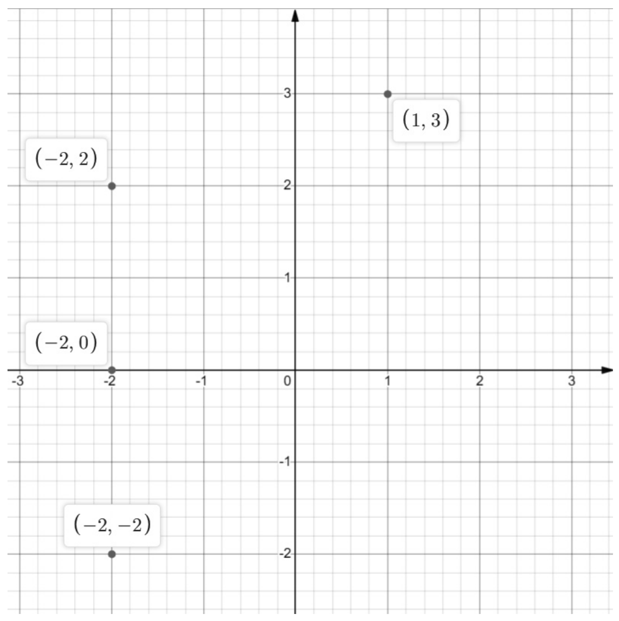
standard input	standard output
3	Yes
4 1	No
-2 -2	Yes
-1 2	
1 3	
2 -1	
4 1	
-2 -2	
-2 2	
1 3	
-2 0	
8 1	
-2 -2	
-2 -2	
-1 2	
-1 2	
1 3	
1 3	
2 -1	
2 -1	

Note

In the first test case, the points are $(-2,-2)$, $(-1,2)$, $(1,3)$, $(2,-1)$. You can draw $x = -0.39$ and $y = 0.39$ to divide the map into 4 regions. It can be shown that the number of points in each region is 1. So it is possible.



Case 1



Case 2

In the second test case, the points are $(-2,-2)$, $(-2,2)$, $(1,3)$, $(-2,0)$. It can be shown that it is impossible.

In the third test case, note that the points **can overlap**.

Problem L. Terabyte Connection

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

The secret of getting ahead is getting started. The secret of getting started is breaking your complex overwhelming tasks into small manageable tasks, and then starting on the first one.

— Mark Twain, *The Adventures of Tom Sawyer*

HC sends you a link to an urgent file. The file is too large to download it singly, so you decide to download it in chunks.

Assume that the file is divided into n chunks. For the i-th chunk, it will be successfully connected at moment p_i, then it will take exactly t_i seconds to download.

Determine the moment P when all chunks are successfully connected and the moment F when all chunks finish downloading.

Input

The first line contains a single integer t ($1 \le t \le 10^4$), denoting the number of test cases.

The first line of each test case contains a single integer n ($1 \le n \le 2 \times 10^5$).

The i-th of the next n lines contains two integers p_i, t_i ($1 \le p_i, t_i \le 10^9$).

It's guaranteed that the sum of n over all test cases doesn't exceed 2×10^5.

Output

For each test case, output two integers P, F, denoting the moment when all chunks are successfully connected and the moment when all chunks finish downloading.

Example

standard input	standard output

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

Note

In the first test case, there is only 1 chunk. Evidently, $P = 1$, $F = 2$.

In the second test case, there are 3 chunks.

1. At moment 1, chunk 2 is successfully connected;
2. At moment 2, chunk 1 is successfully connected, and chunk 2 finishes downloading;
3. At moment 3, chunk 3 is successfully connected;
4. At moment 5, chunk 1 and chunk 3 finish downloading.

Therefore, $P = 3$, $F = 5$.

This page is intentionally left blank