Problem A. Service Diagnostics

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

You are developing a service for managing airline schedules. To optimize the service, you have been asked to develop a library for diagnosing the performance of the service.

A `span` is defined as a continuous period of time characterized by the following parameters:

- id — the identifier of the `span`.
- p_{id} — the identifier of the parent `span`. If the `span` has no parent, it is considered that $p_{id} = 0$.
- t_{start} — the start time of the `span`, when the first entry into the `span` occurred.
- t_{end} — the end time of the `span`, when the last exit from the `span` occurred.

The library must provide the following interface (set of available actions):

- `new` id — create a new `span` with identifier id. The parent `span` will be considered the last `span` that was entered but not yet exited.
- `drop` id — delete the `span` with identifier id. No further entries can be made into this `span`, but it can still be exited, and a new child `span` can be created.
- `enter` id — enter the `span` with identifier id.
- `exit` — exit the current `span`. The current `span` becomes the last `span` that was entered but not yet exited.

To test the library, you have been provided with a list of actions, each of which must be performed at a specific moment in time. Based on this data, you need to compute the characteristics for each `span` that was entered at least once: the parent identifier, start time, and end time.

Note that it is possible to exit from a `span` before exiting from its child `span`s. In this case, the exit time should be considered equal to the maximum exit time from the child `span`s. For example, the following sequence of actions is considered valid:

```
new 1 // create a new span with id = 1
enter 1 // enter the span with id = 1
new 2 // create a new span with id = 2, whose parent is the span with id = 1
exit // exit from the span with id = 1 at time t1
enter 2 // enter the span with id = 2
exit // exit from the span with id = 2 at time t2
drop 1 // delete the span with id = 1
drop 2 // delete the span with id = 2
```

In this case, the exit time from the `span` with $id = 1$ will be considered equal to the exit time `t2` from the `span` with $id = 2$.

Input

The first line contains a single integer t — the number of test cases. The following t test cases are provided.

The first line of each test case contains a single integer n — the number of events.

The next n lines contain descriptions of actions in one of the following formats:

- t_i `new` id
- t_i `drop` id
- t_i `enter` id
- t_i `exit`

It is guaranteed that the actions are strictly in increasing order of t_i. The following guarantees are also provided:

- When creating a new `span`, a previously unused id is used.
- Each `span` is deleted exactly once at any moment after its creation.
- Entry into a `span` occurs only after its creation and before its deletion.
- For each `span`, the number of entries matches the number of exits.
- Exiting from a `span` cannot occur when there is no active `span`.

$$1 \le t \le 10^5$$
$$1 \le n \le 5 \cdot 10^5$$
$$1 \le id \le 10^{18}$$
$$1 \le t_i \le 10^{18}$$

It is guaranteed that the sum of n does not exceed $5 \cdot 10^5$.

Output

For each test case, output the first line containing an integer k — the number of `span`s that were entered at least once.

In the next k lines, output four integers id, p_{id}, t_{start}, t_{end} — the identifier, parent identifier, start time, and end time of the `span`, respectively. The lines must be output in ascending order of the identifier.

Example

standard input	standard output
4 4 1 new 1 2 new 2 3 drop 1 4 drop 2 6 1 new 1 2 enter 1 3 new 2 4 exit 5 drop 1 6 drop 2 10 12 new 1 16 enter 1 22 new 2 42 enter 2 55 new 3 60 drop 1 73 drop 2 75 exit 93 exit 97 drop 3 6 1 new 1 2 enter 1 3 enter 1 4 drop 1 5 exit 6 exit	0 1 1 0 2 4 2 1 0 16 93 2 1 42 75 1 1 0 2 6

Problem B. New Runway

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

Recently, at one of the airports of a major airline, flight delays have been occurring due to high traffic. To improve the quality of passenger service, it was decided to build a new runway. For the construction of the runway, they turned to you as the best specialist.

The airport territory can be represented as a grid, where the bottom left cell has coordinates $(0, 0)$, the bottom right cell has coordinates $(42 \cdot 10^9, 0)$, the top left cell has coordinates $(0, 42 \cdot 10^9)$, and the top right cell has coordinates $(42 \cdot 10^9, 42 \cdot 10^9)$. You have been provided with n concrete slabs, where the i-th slab covers a rectangle of width w_i and height h_i cells. To successfully construct the runway, you need to place some slabs in such a way that there is a path from cell $(0, 0)$ to cell (a_x, a_y), passing through adjacent cells covered by concrete slabs. Cells are considered adjacent if they share a side (left, right, bottom, or top). The concrete slabs cannot be rotated, and they cannot overlap each other.

You need to arrange some of the provided slabs to create a runway.

Input

The first line contains a single integer t — the number of test cases. The following are t test cases.

In the first line of a test case, three integers n, a_x, and a_y are given — the number of concrete slabs and the coordinates of the cell, respectively.

In the next n lines, two integers w_i and h_i are given — the width and height of the i-th slab, respectively.

$$1 \le t \le 10^5$$
$$1 \le n \le 5 \cdot 10^5$$
$$0 \le a_x, a_y \le 10^9$$
$$(a_x, a_y) \ne (0, 0)$$
$$0 \le w_i, h_i \le 10^9$$

It is guaranteed that the sum of n does not exceed $5 \cdot 10^5$.

Output

For each test case, output "`No`" (without quotes) in the first line if it is impossible to build a runway from the provided slabs. Otherwise, output "`Yes`" (without quotes) in the first line.

In the second line, output the number k $(1 \le k \le n)$ — the number of slabs that need to be used.

In the following k lines, output three integers p_i, x_i, y_i $(1 \le p_i \le n,\ 0 \le x_i, y_i \le 10^9)$ — the slab number and the coordinates of the bottom left cell of the slab, respectively. You need to output the slabs in such an order that the i-th slab and the $(i + 1)$-th slab in the output are adjacent. Two slabs are considered adjacent if there exists a cell in the first slab and a cell in the second slab such that these cells are adjacent.

Each slab can be used no more than once.

Example

standard input	standard output
1 3 6 6 6 1 1 5 1 2	Yes 3 1 0 0 2 6 0 3 6 5

Problem C. The Dark Side of the Internet

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

The famous video blogger Ivan decided to shoot another video about the beauties of Belarus and immediately flew to Minsk. The flight lasts a whole 2 hours, so to keep the passengers entertained, the airline provided internet access right on board the plane! Ivan decided to take advantage of this feature, but he got so caught up in video games that he almost became a fool! He firmly decided that it was time to escape from the internet...

The internet is represented as a rectangular grid: its bottom left corner is at cell $(0, 0)$, and the top right corner is at cell (n, m). To escape from the internet, Ivan needs to get from cell $(0, 0)$ to cell (n, m), moving only to the right or upwards to adjacent cells. Due to exhaustion, Ivan's coordination is impaired, so he will reach the opposite corner by one of the paths with equal probability.

But it's not that simple! It turns out that k of these cells belong to the dark side of the internet! If Ivan steps on such a cell for the first time on his path, he will be added 1 year in hell! If he steps on the second such cell, he will be added **another** 2 years! And so on: for the i-th such cell he steps on, he will be added **another** i years in hell. Note that if the starting or ending cell belongs to the dark side of the internet, Ivan will step on it in any case.

Ivan is angry that the internet has confused his mind and does not remember which path he took to escape from the internet. Therefore, you, as the director of the airline, must calculate the expected number of years he will spend in hell. Otherwise, he will sue you, as before the flight, Ivan would not have spent a single day in hell!

It can be shown that the answer can be represented in the form $\frac{p}{q}$, where p and q are coprime integers ($q \not\equiv 0 \pmod{10^9 + 7}$). Let q^{-1} denote the integer such that $q \cdot q^{-1} \equiv 1 \pmod{10^9 + 7}$. Output the value of $p \cdot q^{-1}$ modulo $10^9 + 7$.

Input

The first line contains three integers: n, m, and k.

$$1 \le n, m \le 5 \cdot 10^6$$

$$0 \le k \le min(5000, (n + 1) \cdot (m + 1))$$

In each of the next k lines, two integers are given: x and y — the coordinates of the cell on the dark side of the internet (it is guaranteed that each cell in the input is unique).

$$0 \le x \le n$$

$$0 \le y \le m$$

Output

Output a single integer — the answer to the problem.

Examples

standard input	standard output
5 5 5 0 4 1 3 2 2 3 1 4 0	1
3 3 2 1 1 2 2	200000003
10 10 5 0 0 4 7 2 8 6 9 10 10	599184879

Note

In the first test case, each path will pass through exactly one cell on the dark side of the internet, so the expected number of years in hell is 1.

In the second test case, it can be calculated that 4 paths will pass through 0, 8 paths will pass through 1, and 8 paths will pass through 2 cells on the dark side of the internet, so the expected number of years in hell is $\frac{4 \cdot 0 + 8 \cdot 1 + 8 \cdot (1+2)}{4+8+8} = \frac{32}{20} = \frac{8}{5}$.

Problem D. Safe Key

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

Ivan loves to travel very much, but even more, he loves to bring various souvenirs from the countries he has visited. To bring a souvenir home safely, Ivan uses his favorite safe, which he bought during one of his trips.

The lock of the safe is a mechanical combination lock consisting of k cells, where each cell represents a cyclic mechanism for selecting exactly one digit. To open the lock, it is necessary to enter the combination that was used to close the lock.

Over a long time of traveling, Ivan has developed a special strategy for choosing combinations to close his safe. A combination s is considered valid if and only if it can be divided into a set of unique substrings that represent Thue-Morse strings for the alphabet $\{0, 1\}$.

The sequence of Thue-Morse strings for the alphabet $\{0, 1\}$ is a sequence of strings that consists only of the characters $\{0, 1\}$ and is generated by the following rule:

$$t_0 = 0,$$

$$t_k = concat(t_{k-1}, inverse(t_{k-1})),$$

where concat(x, y) is the concatenation of strings x and y, and inverse(s) is the inversion of the characters of string s over the alphabet $\{0, 1\}$, meaning that 0 changes to 1 and 1 changes to 0.

For example, the first 5 strings of the Thue-Morse sequence look as follows:

$$t_0 = 0$$

$$t_1 = 01$$

$$t_2 = 0110$$

$$t_3 = 01101001$$

$$t_4 = 0110100110010110$$

Ivan is knowledgeable about security and has a certain level of paranoia, so after each opening of the safe, he can change the current combination on the safe to any sequence of characters (not necessarily valid) so that no one can peek at the previous combination he used to close the safe.

Since Ivan is not only knowledgeable about security but also about optimization, every time he closes the safe, he wants to make the minimum number of rotations of the lock cells.

You are given a string s — the current combination of characters on the safe lock, and you need to determine the minimum number of rotations of the lock cells required to transform s into a valid combination.

Input

A single string s — the initial combination on the safe lock of length $k \leq 10^6$.

Output

A single string that contains the minimum number of operations to transform the string s into a valid combination.

Examples

standard input	standard output
8009	5
90091001	4
0110	0

Note

Explanation for the example 8009. The minimum number of operations is 5, achieved as follows:

- 8 -> 9 -> 0 (2 operations)
- 0 -> 1 (1 operation)
- 0 -> 1 (1 operation)
- 9 -> 0 (1 operation)

Problem E. Maintenance

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

Unfortunately, airplanes can break down, so they need to be constantly maintained. You are given m airplanes, each with its own size b_i. You also have n hangars, each with its own size a_i. To service an airplane of size b_i in a hangar of size a_j, it will take $\frac{b_i}{a_j}$ hours. Additionally, if the size of the airplane exceeds the size of the hangar, it will not be possible to service that airplane.

Currently, all the airplanes are lined up in front of the hangars, and since moving them is very difficult, the airplanes need to be divided into n groups, where each group will contain a certain number of consecutive airplanes. The number of airplanes in a group can be zero. Airplanes cannot be swapped. Let the group numbered i start with the l_i-th airplane and end with the r_i-th airplane. For all groups $1 \le i, j \le n$ and $i < j$, one of the following two conditions must hold:

- $r_i < l_j$.
- There are no airplanes in group i or j.

All airplanes must be assigned to a group. After the airplanes are divided into groups, the airplanes from group i will be serviced in hangar i in turn. Each hangar services airplanes simultaneously. Your task is to determine the minimum number of hours required to service the airplanes; if servicing the airplanes is not possible, output -1.

Input

The first line contains an integer n — the number of hangars.

The second line contains n integers a_i — the size of hangar number i.

The third line contains an integer m — the number of airplanes.

The fourth line contains m integers b_i — the size of airplane number i.

$$1 \le n, m \le 10^5$$
$$1 \le a_i, b_i \le 10^5$$
$$1 \le b_i \le 10^5$$

Output

Output a single number, the minimum number of hours required; if servicing the airplanes is impossible, output -1. The absolute or relative error must not exceed 10^{-6}.

Examples

standard input	standard output
3 2 6 4 4 1 2 6 4	1.33333333
2 1 3 3 1 1 4	-1

Problem F. Vacation

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

Oh no! There has been an imbalance on the airplane, and now Vadim's vacation in Bali is at risk.

The airplane, which is about to take off, holds f liters of fuel. It is known that the airplane consumes $\frac{v}{m} + c$ liters of fuel per 100 kilometers, where v is the speed of the airplane, m is the maximum possible speed of the airplane, and c is the constant minimum fuel consumption at idle. Vadim studied the map and realized that he needs to fly exactly s kilometers. Now Vadim is wondering if he can choose a flight speed such that he can reach his destination. It can be assumed that the airplane can reach any speed within $(0, m]$ instantly.

Input

The first line contains four integers f, m, c, s — the fuel tank capacity, the maximum allowable speed, the constant minimum fuel consumption, and the distance that needs to be covered.

$$1 \le f, m, c, s \le 10^6$$

Output

In a single line, output "`Yes`" (without quotes) if Vadim will be able to fly to Bali, and "`No`" (without quotes) otherwise.

Examples

standard input	standard output
450 1500 15 3000	No
451 1500 15 3000	Yes

Problem G. Correct Routes

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

Vadim recently learned about correct bracket sequences. A correct bracket sequence is a string that can be obtained using the following rules:

- `()` — a correct bracket sequence.
- `(`x`)` — a correct bracket sequence if x is also a correct bracket sequence.
- $x\ \ y$ — a correct bracket sequence if x and y are also correct bracket sequences.

Examples of correct bracket sequences: `((()(()))`, `()()()()`, `()`.

Examples of strings that are not correct bracket sequences: `((`, `)(`.

Vadim has a map consisting of n cities. The cities are connected by $n-1$ flights, each of which allows movement in either direction. It is guaranteed that it is possible to reach any city from any other city using one or more flights. Vadim decided to write down m strings consisting only of the characters "`(`" and "`)`", and assign one of these strings to each flight (the strings may repeat).

Now Vadim is interested in how many ordered pairs of cities (a, b) $(a \neq b)$ exist such that the route between these cities is correct. A route is considered correct if the concatenation of the strings assigned to the flights on the path from a to b forms a correct bracket sequence.

Input

The first line contains a single integer t — the number of test cases. The following are t test cases.

In the first line of a test case, two integers n and m are given — the number of cities and the number of strings, respectively.

In the next m lines, one string s_i is given, consisting only of the characters "`(`" and "`)`".

In the next $n-1$ lines, three integers a_i, b_i, j_i are given — the numbers of the cities connected by a flight and the index of the string, respectively.

$$1 \leq t \leq 10^5$$
$$2 \leq n \leq 2 \cdot 10^5$$
$$1 \leq m < n$$
$$|s_i| \geq 1$$
$$1 \leq a_i, b_i \leq n$$
$$a_i \neq b_i$$
$$1 \leq j_i \leq m$$

It is guaranteed that the sum of n does not exceed $2 \cdot 10^5$, and the sum of the lengths of all strings s_i does not exceed $5 \cdot 10^5$.

,

Example

standard input	standard output
3 5 3 (( )) (()) 1 2 1 2 3 2 3 4 3 4 5 3 5 2 ( ) 1 2 1 1 3 2 3 4 1 3 5 2 2 1 () 1 2 1	9 3 2

Note

In the second test case, the following paths are valid: $[2, 1, 3]$, $[4, 3, 5]$, and $[4, 3, 1]$.

In the third test case, the following paths are valid: $[1, 2]$ and $[2, 1]$.

Problem H. Experiment with Mirrors

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	256 megabytes

An optical laboratory has created a system of n rows and m columns. In some cells, k mirrors are installed at a 45-degree angle to the vertical or horizontal line: "/" and "\". A laser beam starts from the left at position $(r, 0)$ for row r and moves to the right. A row is considered safe if the beam reaches the right boundary $c = m + 1$, and dangerous if it gets stuck in a loop or exits through another boundary.

You are asked to conduct q experiments. For each experiment, you need to determine how many rows in the range $[a, b]$ are considered safe.

Input

The first line contains a single integer t — the number of test cases. The following are t test cases.

In the first line of a test case, three integers n, m, and k are given — the number of rows and columns in the optical system, as well as the number of mirrors installed.

In the next k lines, two integers and one character are given: r_i, c_i, t_i ($t \in \{\backslash, /\}$) — the row and column number where the mirror is located, as well as its direction (the character "/" or "\").

In the next line, a single integer q is given — the number of experiments.

In the following q lines, two integers a_i and b_i are given — the range of rows for the i-th experiment.

$$1 \le t \le 10^5$$

$$1 \le n, m \le 10^9$$

$$0 \le k \le 2 \cdot 10^5$$

$$1 \le q \le 2 \cdot 10^5$$

$$1 \le r_i \le n$$

$$1 \le c_i \le m$$

$$1 \le a_i \le b_i \le n$$

It is guaranteed that the sum of k and the sum of q do not exceed $2 \cdot 10^5$.

Output

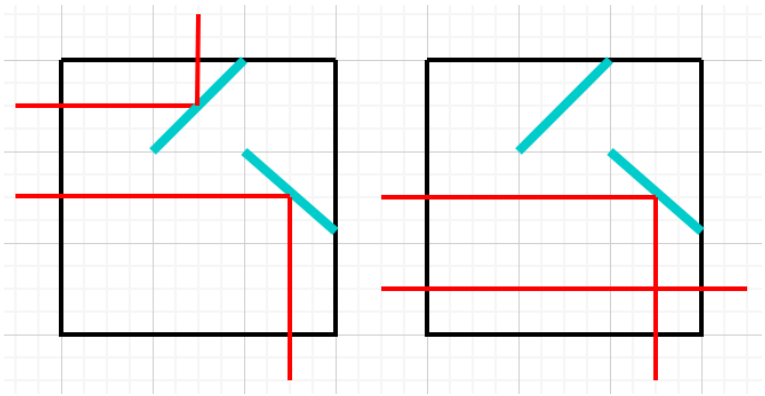
For each test case, output q lines. In the i-th line, output a single integer — the number of safe rows for the i-th experiment.

Example

standard input	standard output
2 10 10 5 1 1 \ 4 1 \ 5 3 / 3 3 / 6 6 / 3 1 1 2 4 1 6 3 3 2 1 2 / 2 3 \ 2 1 2 2 3	1 1 3 0 1

Note

Consider the second test case. For the first experiment, one beam was directed in the first row, which was reflected upwards, and the second beam was reflected downwards. Both beams did not reach the right boundary. In the second experiment, the beam in the second row was reflected downwards, while the beam in the third row reached the right boundary without obstruction.



,

Problem I. Palindrome

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

The airline "Chopik Airways" uses an unusual flight numbering system. Each flight has a unique number consisting of two numbers — *number* and *base*. The coolest part is that if we represent the number *number* in base *base*, this number will be a palindrome of even length.

A palindrome is a number that reads the same forwards and backwards. For example, 101101 is a palindrome, while 1010 is not. The length of a number refers to the number of digits it has.

Each airplane has three parameters — the number of passengers a, the payload capacity b, and the flight range c. Their product defines the number *number*, which is used in the airplane's number. However, since several airplanes may have the same values for $a \cdot b \cdot c$, there may be difficulties in finding multiple suitable *base*. As the lead programmer of "Chopik Airways", you have been tasked with finding the number of suitable *base* for given a, b, c.

It can be shown that the number of such *base* is a finite number.

Input

The first line contains three integers: a, b, and c.

$$1 \le a, b, c \le 10^6$$

Output

Output a single integer — the answer to the problem.

Examples

standard input	standard output
1 3 3	2
1 3 5	3
52 42 69	48

Note

In the first test case, the number *number* is equal to 9. It can be shown that only $base = 2$ ($9_{10} = 1001_2$) and $base = 8$ ($9_{10} = 11_8$) are suitable.

Problem J. This is apparently a bomb!

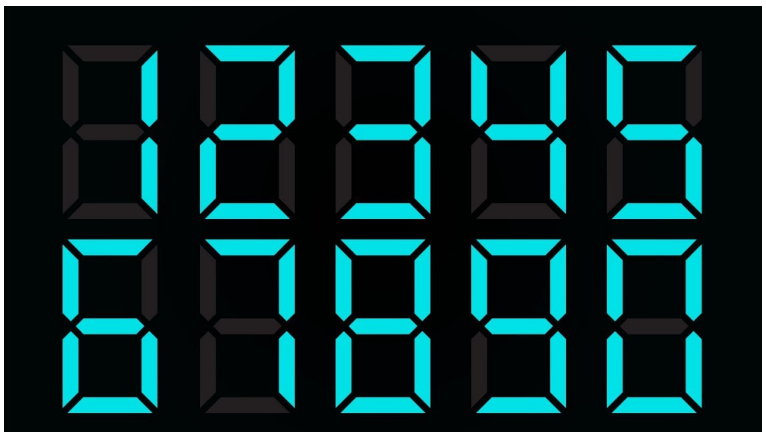
Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Is there a sapper on board? Don't worry, I just asked...

The criminal has smuggled a bomb right onto the plane! Fortunately, the crew discovered it in advance, and you happen to be the world champion in the game of Minesweeper, so you should have no trouble defusing the bomb.

But the bomb turned out to be complicated: its timer consists of n digits, and some segments on the bomb's timer are broken, meaning they never light up. After some questioning, the criminal revealed which segments do not light up in the first m digits, and for each subsequent digit, the segments that do not light up are exactly the same as those that do not light up in the digit m places to the left. Additionally, if the remaining time has fewer digits than the timer's digits, leading zeros will be displayed.

The digits on the timer are recorded as follows:



After careful analysis of the timer, you discovered that the countdown has not yet started. Furthermore, if we denote the digit in position i as $digit_i$, then $digit_i = digit_{i-1} \cdot a + b \bmod 10$ for all $i > m$.

But suddenly the timer started its countdown! After another conversation with the criminal, he confessed: to defuse the bomb, you need to find the number of timer values less than or equal to the initial one, in which no lit segment is broken.

Since the answer can be very large, it needs to be taken modulo $10^9 + 7$.

Input

The first line contains the number n — the number of digits in the countdown timer.

The second line contains the number m — the number of the first known digits of the countdown timer.

The third line contains two numbers a and b — the coefficients.

The fourth line contains the first m digits of the countdown timer without spaces.

For each i from 1 to m, in the line $4 + i$, an integer $0 \le mask_i < 2^7$ is given — the bitmask of broken segments. If the j-th bit of the number $mask_i$ is equal to 1, it means that the j-th segment in the i-th digit is not working. The numbering of the segments can be seen in the picture below.

$$1 \le n \le 10^{18}$$

$$1 \le m \le 10^5$$

,

$$1 \le a, b \le 100$$

Output

One line — the answer to the problem modulo $10^9 + 7$.

Examples

standard input	standard output
2 1 1 1 5 0	57
1 1 55 89 3 72	0

Note

In the first test case, we compute the full value of a timer of length $n = 2$. Since $m = 1$, we compute the second digit using: $digit_2 = 5 * 1 + 1 = 6$

All segments work, so we can use any digit from 0 to 9.

So, the total allowed digits are: 00 to 56, that is 57 values.

Segment numbering in digital:

Problem K. Interview

Input file:	standard input
Output file:	standard output
Time limit:	5 seconds
Memory limit:	256 megabytes

In order to get an interview with a major airline, you need to solve the following problem.

You are given an array of length $42 \cdot 10^9$, initially filled with 0s.

You are asked to perform n operations sequentially, where the i-th operation is described by four integers l_i, r_i, a_i, and b_i, which mean that for each index j in the array such that $l_i \le j \le r_i$, the value of the corresponding element of the array should be increased by $a_i + b_i \cdot (j - l_i)$.

Then you are asked to compute the remainder of each element of the array when divided by m and sum the obtained remainders. The result of the summation will be the answer to the problem.

Input

The first line contains a single integer t — the number of test cases. The following are t test cases.

In the first line of a test case, two integers n and m are given — the number of operations to be performed and the number for obtaining remainders, respectively.

In the next n lines, four integers l_i, r_i, a_i, b_i are given — the description of the operation to be performed on the array.

$$1 \le t \le 10^5$$
$$1 \le n \le 5 \cdot 10^5$$
$$2 \le m \le 10^9 + 7$$
$$1 \le l_i \le r_i \le n$$
$$0 \le a_i, b_i \le m$$

It is guaranteed that the sum of n does not exceed $5 \cdot 10^5$.

Output

For each test case, output a single integer — the answer to the problem.

Example

standard input	standard output
3 2 100 4 4 5 10 1 3 2 3 3 1000000007 1 1000 0 1 500 5000 42 128 1 2000 100 0 2 2 1 10 0 1 1 10 1 1	20 1297176542 10

Note

In the third test case, the array after the first operation looked like this:

$$[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, ...]$$

After the second operation, the array looked like this:

$$[1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 0, ...]$$