Problem A. A + B = C

Input file: standard input
Output file: standard output

Hi everyone! Welcome to the Stage 7 of this series of training contests. Cuber QQ is the problem settler of this contest; he has prepared 11 problems for you and wishes you to enjoy it. Good luck!

As the first problem of this contest, Cuber QQ thought that it's reasonable to start with an easy one, so he modified the famous A + B problem by a little bit, so that it's easy enough but not that trivial.

Given a, b, c , find an arbitrary set of x, y, z such that $a \cdot 10^x + b \cdot 10^y = c \cdot 10^z$ and $0 \leq x, y, z \leq 10^6$.

Input

The input consists of multiple test cases, starting with an integer t ($1 \leq t \leq 100$), denoting the number of test cases.

For each test case, there are three space-separated integers, a, b, c respectively ($1 \leq a, b, c \leq 10^{100000}$).

Output

For each test case, please output three space-separated integers x, y, z. If there are multiple solutions, print any of them.

In case there are no solutions, please output -1 as a single line.

Example

standard input	standard output
3 23 39 62 2 31 51 1 1 1	0 0 0 1 0 0 -1

Note

HDOJ may give ambiguous feedback due to the compatibility issue of our checker. If you see "System Error", please think of it as "Wrong Answer".

Problem B. Bracket Sequences on Tree

Input file: standard input
Output file: standard output

Cuber QQ knows about DFS on an undirected tree, he's sure that you are familiar with it too. In case you are not, Cuber QQ is delighted to share with you a snippet of pseudo code:

```
function dfs(int cur, int parent):
  print(‘(‘)
  for all nxt that cur is adjacent to:
    dfs(nxt, cur)
  print(‘)‘)
```

You might notice that Cuber QQ print a "`(`" when entering a node, and a "`)`" when leaving a node. So when he finishes this DFS, in his console, he will see a bracket sequence of length $2n$, where n is the number of vertices in the tree.

Obviously, if the tree is undirected and the nodes are unlabelled (meaning that all the nodes are treated equally), you can get a lot of different bracket sequences when you do the DFS. There are two reasons accounting for this. Firstly, when you are at `cur`, you can follow any permutation of the nodes that `cur` is adjacent to when you visit `nxt`. Secondly, the entrance to the tree, that is the root, is undeterministic when you start your DFS.

So Cuber QQ couldn't help wondering how many distinct bracket sequences he can get possibly. As the answer can be very large, output it modulo 998 244 353.

Input

The first line of the input consists of one integer t ($1 \le t \le 10^5$), which is the number of the test cases.

For each test case, the tree is given in a standard format, which you might be very familiar with: first line n ($1 \le n \le 10^5$), the size of tree; then $n - 1$ lines, each consisting of two space-separated integers u, v ($1 \le u, v \le n$, $u \ne v$), denoting an edge.

The sum of n from all test cases does not exceed 3.2×10^6.

Output

For each test case, output the answer in one line.

Example

standard input	standard output
3 4 1 3 2 3 4 3 5 1 2 2 3 3 4 4 5 5 1 2 2 3 3 4 3 5	2 4 8

Problem C. Cuber Occurrence

Input file: `standard input`
Output file: `standard output`

Recently, Cuber QQ has received a letter from his old friend Little Fang, which is a strangely long paper note. Cuber QQ thought it will be interesting to count the occurrences of some words in this paper note, thus he brought in a dictionary. If you are curious about what occurrences mean, here are some examples: `nan` appears once in `banana`; `a` appears 3 times in `banana`, `ana` appears twice in `banana`.

When Cuber QQ got up the second day, he forgot all about his counting yesterday, and he figured he had to do that again. So he used a highlight pen and highlighted those occurrences. Note that a letter can't be highlighted more than once. For example, finding `ana` in `banana` will result in highlighting altogether 5 letters.

On the third day, Cuber QQ again noticed a problem he could have noticed on the second day. He can't tell the number of occurrences from his highlighting because of some touching or overlapping area. If you are counting `a` in `banana`, you will get 3; that's fine. Counting `an` or `ana` will all give you the wrong count, which is 1. Cuber QQ has finally given up and named his incorrect counting as Cuber Occurrence.

Let's do a more complicated example to make sure everything is clear. Cuber Occurrence of `cuc` in `cucucuberqqcucuber` is 2, and that of `q` is 1 (since two `q` are touching each other), and that of `u` is 5.

Cuber QQ couldn't help wondering the opposite problem. If he knows the Cuber Occurrence of a word in a string, could you find him this word?

Input

The first line of the input contains an integer t ($1 \leq t \leq 100$), indicating there are t tests.

The follows t test cases. Each test consists of two lines. The first line contains a lowercase ascii string s ($1 \leq |s| \leq 10^5$, $\sum |s| \leq 10^6$), and the second line is an integer k ($1 \leq k \leq |s|$).

Output

For each test case, output a word in one line, which has the Cuber Occurrence of k in s. In case there are more than one solutions, output the lexicographically smallest one. If there is no such string, output -1 instead.

Example

standard input	standard output
4 cucucuberqqcucuber 2 cucucuberqqcucuber 5 cucucuberqqcucuber 1 cucucuberqqcucuber 10	b c berq -1

Note

Special note if you are doing this problem on HDOJ: The memory limit can be a little tight as 512MB is the best we can do.

Problem D. Data Structure Problem

Input file: standard input
Output file: standard output

```
using namespace std;
const unsigned mul = 20190812;

class Magic {
public:
  Magic(unsigned state): state(state), ans(0) {}

  unsigned long long retrieve() {
    unsigned modulo = 0x7fffffff;
    state = ((unsigned long long) state * mul + ans) % modulo;
    unsigned high = state;
    state = ((unsigned long long) state * mul + ans) % modulo;
    return high * modulo + state;
  }

  int retrieve(int a, int b) {
    assert (a <= b);
    return (int) (retrieve() % (b - a + 1)) + a;
  }

  void submit(unsigned k) {
    ans = ans * mul + k;
  }

  unsigned retrieveAns() {
    return ans;
  }

private:
  unsigned state, ans;
};

class DataStructure {
public:
  DataStructure() {
    // The data structure is initially empty, until it's not.
    // Implement your initialization here.
  }
  void add(int x, int y) {
    // Add a 2D point (x, y) to the DS.
    // Implement your add here.
  }
  void erase(int r) {
    // Erase the r-th added point, of all the points that
    // have still not been erased.
    // Implement your erase here.
  }
  int size() {
    // Return how many points are still in the DS
```

```
  }
  pair<int, int> query() {
    // find two points p_i, p_j in the DS (not necessarily distinct),
    // such that the dot product of these two <p_i, p_j> (i <= j)
    // the smallest among all. Return (i, j).
    // If the DS is empty for now, return (0, 0).
    // Implement your query here.
  }
};

int main() {
  const int lim = 1E9;
  int q; cin >> q;
  for (int k = 0; k < q; ++k) {
    unsigned state;
    cin >> state;
    string s;
    cin >> s;
    DataStructure ds;
    Magic magic(state);
    for (char c: s) {
      if (c == 'a') {
        // add one point
        int x = magic.retrieve(-lim, lim);
        int y = magic.retrieve(-lim, lim);
        ds.add(x, y);
      } else if (c == 'e') {
        // select the lucky point
        unsigned pos = magic.retrieve(1, ds.size());
        ds.erase(pos);
      } else if (c == 'q') {
        // query global minimum
        auto best = ds.query();
        magic.submit(best.first);
        magic.submit(best.second);
      }
    }
    cout << magic.retrieveAns() << endl;
  }
}
```

Input

The input format is a little different from traditional. Instead of giving a large file of input, the input is produced by the program above in a protocol. Thus reading the code and getting what it's doing is a fundamental step to solve this problem. The code is provided only in C++, but it should be easily converted into other languages.

As you probably have seen, the program implements a protocol to generate the input data, based on some seeds (or input of input, if you like). The program reads from standard input: in the first line an integer t ($1 \le t \le 20$), which is the number of test cases; then t cases, each in two lines: first the integer state, which is between 10 and 10^7, inclusive; then follows a non-empty string of `aeq` of length no greater than 10^5. `a` means add, `e` means erase and `q` means query. You would always find something to erase, i.e., we guarantee we will not try to erase from an empty data structure.

Output

Follow the practice of the given code for the output.

Example

standard input	standard output
3 42 qaq 84 aqaqaeqaaaqaeqeq 9102 aaaaaaqeqeqeqeqeqeq	20190813 2668638611 549902096

Note

Dot product of two points p and q, is here, defined as the dot product from 0 to p and 0 to q. Specifically, if $p = (a, b)$, $q = (c, d)$, dot product of p and q is $ac + bd$.

For sample input one, the queries are:

- Query: return $(0, 0)$.
- Add 1st point: $(518204527, -960426430)$.
- Query: return $(1, 1)$.

The answer is $20190812 \cdot 1 + 1 = 20190813$.

For sample input two, the modifications and queries are:

- Add 1st point: $(-258558241, -773369213)$.
- Query: return $(1, 1)$.
- Add 2nd point: $(150662440, -65439273)$.
- Query: return $(1, 2)$.
- Add 3rd point: $(-888171126, -337111254)$.
- Erase 3rd point added among all still alive, so 3rd was erased.
- Query: return $(1, 2)$.
- Add 4th point: $(-470261300, 657949534)$.
- Add 5th point: $(-592507383, 366158932)$.
- Add 6th point: $(635425367, 329355360)$.
- Query: return $(1, 6)$.
- Add 7th point: $(900114299, 534416578)$.
- Erase 5th point added among all still alive, so 6th was erased.
- Query: return $(1, 7)$.
- Erase 2nd point added among all still alive, so 2nd was erased.

- Query: return $(1, 7)$.

Sample input three decodes to be as follows:

- Add 1st point: $(-457246811, -252726229)$.
- Add 2nd point: $(-815357226, 987160210)$.
- Add 3rd point: $(689370621, -861950583)$.
- Add 4th point: $(-850699215, -38670848)$.
- Add 5th point: $(187488045, -980321149)$.
- Add 6th point: $(217752313, -264046720)$.
- Query: return $(2, 3)$.
- Erase 2nd point added among all still alive, so 2nd was erased.
- Query: return $(3, 4)$.
- Erase 3rd point added among all still alive, so 4th was erased.
- Query: return $(1, 3)$.
- Erase 1st point added among all still alive, so 1st was erased.
- Query: return $(6, 6)$.
- Erase 3rd point added among all still alive, so 6th was erased.
- Query: return $(3, 5)$.
- Erase 1st point added among all still alive, so 3rd was erased.
- Query: return $(5, 5)$.
- Erase 1st point added among all still alive, so 5th was erased.
- Query: return $(0, 0)$.

Problem E. Equation

Input file: standard input
Output file: standard output

Cuber QQ has encountered a math problem during his research of "nothing related to math", he thought the problem too boring for him, and decided to leave it to you. You might be curious about more background about why this problem appeared in his research, but to save your time, let's say "the background isn't helpful for you to solve this problem".

Given n, m, count the number of sequence $x_1, x_2, \ldots, x_n$ such that $x_1^2 + x_2^2 + \cdots + x_{n-1}^2 \equiv x_n^2 \pmod{m}$ and $0 \le x_i < m$ for $1 \le i \le n$.

Input

The first line of the input is an integer t ($1 \le t \le 2500$), where t is the number of test cases.

Then follows t test cases, each of which is a line with two space-separated integers n and m ($3 \le n \le 100, 3 \le m \le 999\,999\,999$ and m is odd).

Output

For each test case, output the answer modulo $10^9 + 7$.

Example

standard input	standard output
3 3 5 5 3 9 101	25 81 980480839

Problem F. Final Exam

Input file: `standard input`
Output file: `standard output`

Final Exam is coming! Cuber QQ has now one night to prepare for tomorrow's exam.

The exam will be a exam of problems sharing altogether m points. Cuber QQ doesn't know about the exact distribution. Of course, different problems might have different points; in some extreme cases, some problems might worth 0 points, or all m points. Points must be integers; a problem cannot have 0.5 points.

What he knows, is that, these n problems will be about n totally different topics. For example, one could be testing your understanding of Dynamic Programming, another might be about history of China in 19th century. So he has to divide your night to prepare each of these topics separately. Also, if one problem is worth x points in tomorrow's exam, it takes at least $x + 1$ hours to prepare everything you need for examination. If he spends less than $x + 1$ hours preparing, he shall fail at this problem.

Cuber QQ's goal, strangely, is not to take as much points as possible, but to solve at least k problems no matter how the examination paper looks like, to get away from his parents' scoldings. So he wonders how many hours at least he needs to achieve this goal.

Input

The first line of the input is an integer t ($1 \le t \le 20\ 000$), denoting the number of test cases.

Each test case are three space-separated integers n, m, k ($0 \le m \le 10^9$, $1 \le k \le n \le 10^9$).

Output

For each test case, output the number of hours Cuber QQ needs.

Example

standard input	standard output
2 1 10 1 10 109 10	11 1100

Note

Cuber QQ should solve one problem in sample 1, so he needs at least 11 hours when the problem 1 is worth 10 points.

Cuber QQ should solve all the ten problems in sample 2, so he needs at least 110 hours to prepare for each problem because they may be one problem worth 109 points.

Problem G. Getting Your Money Back

Input file: standard input
Output file: standard output

For some weird security reason, Cuber Bank now shuts down the interface where you can check your balance on your bank account. From now on, when you click the "check" button on your interface, they will show you a range, saying that the money in your account has to be one of the integers between x and y, inclusive. Yes, less informative, but more secure.

Perhaps you are not convinced of their explanation, and you get annoyed as much as I do. Moreover, you are the kind of person who won't get a good sleep until you see from your screen that your beloved money is lying right in your bank account, safe and sound. Since Cuber Bank can't provide this service any more, it's time to withdraw all your money and go for another bank.

However, when you went to the bank the other day, the manager Cuber QQ told you that, there have been so many people trying to close their account recently, so that they have to do something: to make the rule that you have to pay an extra fee each time you attempt to withdraw money from your account.

There is a machine you can interact with. After you insert your card, the proceses goes in an interactive way, just like playing a game. The machine first tells you a range, $[x, y]$, which your balance is currently between. Then at each attempt, you type in the amount of money (also an integer) you want to withdraw; the machine then tell you whether your operation is successful or not (successful if and only if the amount fits your balance). In order to prevent excessive requests that cause great pressure on the system, if successful you have to pay a dollars for this attempt; if fail you have to pay b dollars. You pay these fees with cash, brought by yourself. You can do more attempts until you are very convinced that there is no more money on your account. The machine will not disclose any further information about any range after the first attempt.

Devise a strategy that will cost you the least dollars of extra fees in the worst case, in order to retrieve all the money from your account.

Input

The first line of the input is an integer t ($1 \le t \le 10^4$), which means t test cases will follow.

The follows t test cases, each is one line of space-separated integers x, y, a, b ($0 \le x \le y \le 2 \cdot 10^5$, $0 \le a, b \le 10^9$).

The sum of y in all test cases does not exceed $8 \cdot 10^6$.

Output

For each test case, output the extra dollars you have to prepare in one line.

Example

standard input	standard output
2 0 2 2 3 0 2 3 2	5 5

Note

In sample 1, the first attempt should be a retrieval of 1 dollar. If this works, there might be one more dollar (or not), for which you need to pay $2 + \max(2, 3) = 5$ altogether; if it fails, then you are done, for which you need to pay 3 dollars.

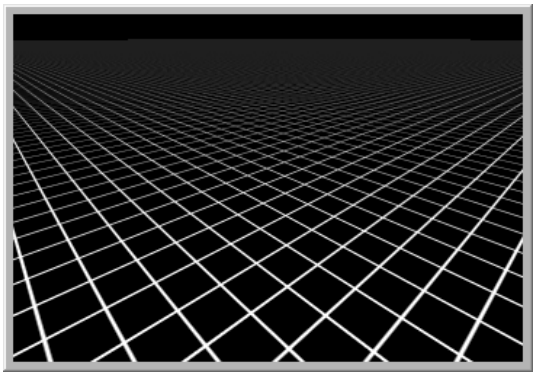
In sample 2, try 2 dollars first, pay 3 dollars if succeeded; otherwise, the balance could be 0 or 1 dollars, for which you will need to pay $2 + \max(2, 3) = 5$ dollars altogether.

Problem H. Halt Hater

Input file: standard input
Output file: standard output

Cuber QQ has recently won a million dollars and bought a new car. He is now driving in the Infinite City and he doesn't want his car to stop, ever!

The Infinite City, looks like an infinite grid, with infinite streets built at all possible x's and y's when they are integers. The following picture, looks like part of this city. (The picture is downloaded from a webpage, which is certainly not describing the Infinite City.)



There are traffic lights installed on every crossings. Just like what Chinese people usually do, the traffic lights only show signals for left and straight and you can always turn right regardless of the traffic light. The estimate time of waiting for a left turn is a, at all crossings; and the time of waiting for a straight is b.

Cuber QQ is now driving from $(0, -1)$ to $(0, 0)$, about to entering the crossing at $(0, 0)$. At a crossing, he can choose to go straight, turn left or turn right. He wishes to go to (x, y), but here is a weird request: he wants to have the least estimated traffic light waiting time. In other words, he doesn't care whether he has to drive a long way, or he has to burn a lot of gas, all he wants is to wait as little as possible.

Note that Cuber QQ doesn't have to wait for the traffic light at (x, y); he also doesn't care from which direction he enters (x, y).

Input

The first line of the input contains an integer t ($1 \le t \le 100\,000$).

Then follows t test cases, each containing a, b, x, y ($1 \le a, b \le 10^9$, $-10^9 \le x, y \le 10^9$, $|x| + |y| > 0$), space separated.

Output

For each test case, output one integer as the answer.

Example

standard input	standard output
4 9 1 0 2 2 1 8 0 2 3 3 3 1 1 -1 -1	2 7 6 1

Problem I. Intersection of Prisms

Input file: `standard input`
Output file: `standard output`

Cuber QQ used to encounter such a problem: find the intersection of two infinitely long convex right prisms A and B. Actually this problem is also on HDU Online Judge, you can solve it afterwards if you are interested.

Cuber QQ quickly solved it, and thought it too easy because his program can handle more than that. So he modify it a little bit, by removing the constraints of "convex", that is, the cross-section of these two prisms are not necessarily convex, but they are still simple.

Formally, you have to find the volume of intersection of two prisms, whose cross-sections are both simple polygons. All joining edges of A are parallel to the z-axis, and all joining edges of B are parallel to the x-axis. If A and B do not intersect, the answer is 0 of course.

Input

The first line is an integer t, denoting the number of test cases follows.

For each test case, there are two polygons, the base polygon of A and B respectively.

An integer n ($3 \le n \le 10^5$) followed by n lines, each containing two space-separated integers (x, y) ($-10^6 \le x, y \le 10^6$), is the representation of the base polygon of A. The following $m + 1$ ($3 \le m \le 10^5$) gives a similar representation for B, except that the coordinates given are (y, z) ($-10^6 \le x, y \le 10^6$).

Each polygon are given in either clockwise or counterclockwise order. Following the convention, polygon P is simple, if no two edges have any points in common, with the obvious exception of two consecutive segments having one common point.

The sum of $m + n$ from all test cases does not exceed $2 \cdot 10^6$.

Output

For each test case, output the answer modulo $10^9 + 7$, that is, if the answer is $\frac{P}{Q}$, you should output $P \cdot Q^{-1}$ modulo $10^9 + 7$, where Q^{-1} denotes the multiplicative inverse of Q modulo $10^9 + 7$.

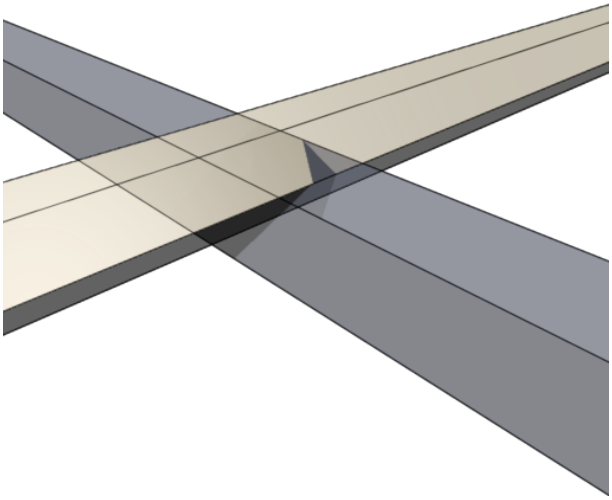
Example

standard input	standard output
1 4 10 0 0 10 -10 0 0 -10 3 5 0 -15 -10 -15 10	666667713

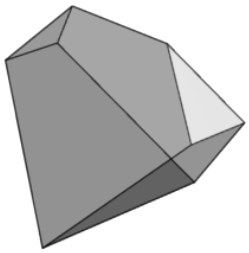
Note

The intersection of the sample input is $\frac{3125}{3}$.

Prisms before intersection:



The intersection part looks like this:



The input size is quite large. Please mind the efficiency of your input buffer.

Problem J. Just Repeat

Input file: standard input
Output file: standard output

When Cuber QQ was chatting happily in a QQ group one day, he accidentally noticed that there was a counterfeit of him, who stole his avatar and mimicked his tone, and more excessively, had a nickname "Quber CC", which was sarcastic to him. So Cuber QQ decided to play a little game with this Quber CC, and their bet was, whoever lost the game would have to do something really humiliating in front of everyone in the QQ group.

The game is like this. It's a traditional card game. Cuber QQ will play first. Cuber QQ, and his opponent (Quber CC of course), will each possess a hand of cards. There is no number (or rank if you prefer) on the card, but only color (or suit if you prefer). The players play cards alternatively, each player can only play one card in each turn. An additional rule is that, a player must **not** play a card with the same color as any card which has been played by his/her opponent, but coincidence with a card played by himself/herself is acceptable.

The player who can't play any card loses. This might due to the fact that he/she has cards but he cannot play any due to the game rules, or he doesn't have any cards any more. As a game played between civilized people, the game will be played in a completely transparent manner, where Cuber QQ and Quber CC know exactly what's left in their opponent's hand.

It's now a game attracting thousands of eyes, and you decided to invent a predictor whose job is to predict "who will win if both players play smart" to excite the audience.

Input

The first line of the input is a positive integer t, denoting the number of test cases.

Then follows t test cases, each test case starts with space-separated integers n, m, p ($1 \le n, m \le 10^5$, $p \in \{1, 2\}$). Generally speaking, this should be followed by two lines of integers $a_1, a_2, \ldots, a_n$ and $b_1, b_2, \ldots, b_m$, denoting the colors of Cuber QQ's hand and Quber CC's hand, respectively. Unfortunately, as it turns out, the input will be the bottleneck in that case. So we introduce p to deal with this problem.

For $p = 1$, there follows two lines, where the first line is $a_1, a_2, \ldots, a_n$ and the second line is $b_1, b_2, \ldots, b_m$, all space separated and $0 \le a_i, b_i < 10^9$.

For $p = 2$, there follows two lines, which are k_1, k_2, mod ($0 \le k_1, k_2 < 2^{64}$, $1 \le mod \le 10^9$) to generate $\{a_i\}$ and $\{b_i\}$, respectively.

Here are some instructions on how to generate $\{a_i\}$, $\{b_i\}$ with k_1, k_2, mod, which you've probably seen before, somehow:

```
unsigned long long k1, k2;
unsigned long long rng() {
    unsigned long long k3 = k1, k4 = k2;
    k1 = k4;
    k3 ^= k3 << 23;
    k2 = k3 ^ k4 ^ (k3 >> 17) ^ (k4 >> 26);
    return k2 + k4;
}

// generate
read(k1, k2, mod);
for (int i = 0; i < n; ++i)
    a[i] = rng() % mod;

read(k1, k2, mod);
for (int i = 0; i < m; ++i)
```

```
b[i] = rng() % mod;
```

Also, the sum of $n + m$ for $p = 1$ does not exceed $5 \cdot 10^5$; the sum of $n + m$ from all test cases does not exceed 10^7.

Output

For each test case, predict the winner: `Cuber QQ` or `Quber CC`.

Example

standard input	standard output
2 6 7 1 1 1 4 5 1 4 1 9 1 9 8 1 0 10 20 2 1 2 10 1 2 10	Cuber QQ Quber CC

Problem K. Kejin Player

Input file: standard input
Output file: standard output

Cuber QQ has always envied those Kejin players, who are used to paying a lot of RMB to get a higher level in the game. So he worked so hard that you are now the game designer of this game. Finally he can propose a new rule in this game, to teach those Kejin players that, RMB can't solve every problem in this world.

Technically, this change hasn't been approved by his boss, so it has to be sneaky and not noticeable. There's a fundamental rule in this game: when you are level i, you have to pay (ke) a_i RMB to get to level $i+1$. Cuber QQ now changed it to: the player shall get to level $i+1$ with probability p_i when he/she pays a_i RMB; with the other $1-p_i$ probability, the player will go into level x_i, where $x_i \leq i$.

Cuber QQ realized that he needs to choose $\{p_i\}$ smart. If these numbers are too small, it's a breaking change and definitely not acceptable; if they are too large, the lesson won't take effect. So he wants to check all q players and their dreaming level, respectively, to see how much money expectedly they have to pay from them to reach that level.

Input

The first line of the input is an integer t, denoting the number of test cases.

For each test case, there is two space-separated integers n ($1 \leq n \leq 500\,000$) and q ($1 \leq q \leq 500\,000$) in the first line, meaning the total number of levels and the number of queries.

Then follows n lines, each containing integers r_i, s_i, x_i, a_i ($1 \leq r_i \leq s_i \leq 10^9$, $1 \leq x_i \leq i$, $0 \leq a_i \leq 10^9$), space separated. Note that p_i is given in the fractional form of $\frac{r_i}{s_i}$.

The next q lines are q players. Each of these queries are two space-separated integers l and r ($1 \leq l < r \leq n+1$), where l is their current level, r is their dreaming level.

The sum of n and sum of q from all t test cases both does not exceed 10^6.

Output

For each player, output answer in the fraction form modulo 10^9+7, that is, if the answer is $\frac{P}{Q}$, you should output $P \cdot Q^{-1}$ modulo 10^9+7, where Q^{-1} denotes the multiplicative inverse of Q modulo 10^9+7.

Example

standard input	standard output
1 3 2 1 1 1 2 1 2 1 3 1 3 3 4 1 4 3 4	22 12

Note

Huge IO (Over 40MB)! IO optimization is preferred.