A. Wizards and Trolleybuses

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

In some country live wizards. They love to ride trolleybuses.

A city in this country has a trolleybus depot with n trolleybuses. Every day the trolleybuses leave the depot, one by one and go to the final station. The final station is at a distance of d meters from the depot. We know for the i-th trolleybus that it leaves at the moment of time t_i seconds, can go at a speed of no greater than v_i meters per second, and accelerate with an acceleration no greater than a meters per second squared. A trolleybus can decelerate as quickly as you want (magic!). It can change its acceleration as fast as you want, as well. Note that the maximum acceleration is the same for all trolleys.

Despite the magic the trolleys are still powered by an electric circuit and cannot overtake each other (the wires are to blame, of course). If a trolleybus catches up with another one, they go together one right after the other until they arrive at the final station. Also, the drivers are driving so as to arrive at the final station as quickly as possible.

You, as head of the trolleybuses' fans' club, are to determine for each trolley the minimum time by which it can reach the final station. At the time of arrival at the destination station the trolleybus does not necessarily have zero speed. When a trolley is leaving the depot, its speed is considered equal to zero. From the point of view of physics, the trolleybuses can be considered as material points, and also we should ignore the impact on the speed of a trolley bus by everything, except for the acceleration and deceleration provided by the engine.

Input

The first input line contains three space-separated integers n, a, d ($1 \le n \le 10^5$, $1 \le a, d \le 10^6$) — the number of trolleybuses, their maximum acceleration and the distance from the depot to the final station, correspondingly.

Next n lines contain pairs of integers t_i v_i ($0 \le t_1 < t_2... < t_{n-1} < t_n \le 10^6$, $1 \le v_i \le 10^6$) — the time when the i-th trolleybus leaves the depot and its maximum speed, correspondingly. The numbers in the lines are separated by spaces.

Output

For each trolleybus print a single line the time it arrives to the final station. Print the times for the trolleybuses in the order in which the trolleybuses are given in the input. The answer will be accepted if the absolute or relative error doesn't exceed 10^{-4}.

Examples

input

```
3 10 10000
0 10
5 11
1000 1
```

output

```
1000.5000000000
1000.5000000000
11000.0500000000
```

input

```
1 2 26
28 29
```

output

```
33.0990195136
```

Note

In the first sample the second trolleybus will catch up with the first one, that will happen at distance 510.5 meters from the depot. The trolleybuses will go the remaining 9489.5 meters together at speed 10 meters per second. As a result, both trolleybuses will arrive to the final station by the moment of time 1000.5 seconds. The third trolleybus will not catch up with them. It will arrive to the final station by the moment of time 11000.05 seconds.

B. Wizards and Huge Prize

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

One must train much to do well on wizardry contests. So, there are numerous wizardry schools and magic fees.

One of such magic schools consists of n tours. A winner of each tour gets a huge prize. The school is organised quite far away, so one will have to take all the prizes home in one go. And the bags that you've brought with you have space for no more than k huge prizes.

Besides the fact that you want to take all the prizes home, you also want to perform well. You will consider your performance good if you win at least l tours.

In fact, years of organizing contests proved to the organizers that transporting huge prizes is an issue for the participants. Alas, no one has ever invented a spell that would shrink the prizes... So, here's the solution: for some tours the winner gets a bag instead of a huge prize. Each bag is characterized by number a_i — the number of huge prizes that will fit into it.

You already know the subject of all tours, so you can estimate the probability p_i of winning the i-th tour. You cannot skip the tour under any circumstances.

Find the probability that you will perform well on the contest and will be able to take all won prizes home (that is, that you will be able to fit all the huge prizes that you won into the bags that you either won or brought from home).

Input

The first line contains three integers n, l, k ($1 \le n \le 200, 0 \le l, k \le 200$) — the number of tours, the minimum number of tours to win, and the number of prizes that you can fit in the bags brought from home, correspondingly.

The second line contains n space-separated integers, p_i ($0 \le p_i \le 100$) — the probability to win the i-th tour, in percents.

The third line contains n space-separated integers, a_i ($1 \le a_i \le 200$) — the capacity of the bag that will be awarded to you for winning the i-th tour, or else -1, if the prize for the i-th tour is a huge prize and not a bag.

Output

Print a single real number — the answer to the problem. The answer will be accepted if the absolute or relative error does not exceed 10^{-6}.

Examples

input

```
3 1 0
10 20 30
-1 -1 2
```

output

```
0.300000000000
```

input

```
1 1 1
100
123
```

output

```
1.000000000000
```

Note

In the first sample we need either win no tour or win the third one. If we win nothing we wouldn't perform well. So, we must to win the third tour. Other conditions will be satisfied in this case. Probability of wining the third tour is 0.3.

In the second sample we win the only tour with probability 1.0, and go back home with bag for it.

C. Wizards and Numbers

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

In some country live wizards. They love playing with numbers.

The blackboard has two numbers written on it — a and b. The order of the numbers is not important. Let's consider $a \le b$ for the sake of definiteness. The players can cast one of the two spells in turns:

- Replace b with $b - a^k$. Number k can be chosen by the player, considering the limitations that $k > 0$ and $b - a^k \ge 0$. Number k is chosen independently each time an active player casts a spell.
- Replace b with $b \ mod \ a$.

If $a > b$, similar moves are possible.

If at least one of the numbers equals zero, a player can't make a move, because taking a remainder modulo zero is considered somewhat uncivilized, and it is far too boring to subtract a zero. The player who cannot make a move, loses.

To perform well in the magic totalizator, you need to learn to quickly determine which player wins, if both wizards play optimally: the one that moves first or the one that moves second.

Input

The first line contains a single integer t — the number of input data sets ($1 \le t \le 10^4$). Each of the next t lines contains two integers a, b ($0 \le a, b \le 10^{18}$). The numbers are separated by a space.

Please do not use the `%lld` specificator to read or write 64-bit integers in C++. It is preferred to use the `cin`, `cout` streams or the `%I64d` specificator.

Output

For any of the t input sets print "`First`" (without the quotes) if the player who moves first wins. Print "`Second`" (without the quotes) if the player who moves second wins. Print the answers to different data sets on different lines in the order in which they are given in the input.

Examples

input

```
4
10 21
31 10
0 1
10 30
```

output

```
First
Second
Second
First
```

Note

In the first sample, the first player should go to (11,10). Then, after a single move of the second player to (1,10), he will take 10 modulo 1 and win.

In the second sample the first player has two moves to (1,10) and (21,10). After both moves the second player can win.

In the third sample, the first player has no moves.

In the fourth sample, the first player wins in one move, taking 30 modulo 10.

D. Wizards and Roads

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

In some country live wizards. They love to build cities and roads.

The country used to have k cities, the j-th city ($1 \le j \le k$) was located at a point (x_j, y_j). It was decided to create another $n - k$ cities. And the i-th one ($k < i \le n$) was created at a point with coordinates (x_i, y_i):

- $x_i = (a \cdot x_{i-1} + b)\ mod\ (10^9 + 9)$
- $y_i = (c \cdot y_{i-1} + d)\ mod\ (10^9 + 9)$

Here a, b, c, d are primes. Also, $a \ne c, b \ne d$.

After the construction of all n cities, the wizards have noticed something surprising. It turned out that for every two different cities i and j, $x_i \ne x_j$ and $y_i \ne y_j$ holds.

The cities are built, it's time to build roads! It was decided to use the most difficult (and, of course, the most powerful) spell for the construction of roads. Using this spell creates a road between the towns of u, v ($y_u > y_v$) if and only if for any city w which lies strictly inside the corner at the point u, v (see below), there is a city s that does not lie in the corner, which is located along the x-coordinate strictly between w and u and simultaneously $y_s > y_v$.

A *corner* on the points $p_2(x_2, y_2)$, $p_1(x_1, y_1)$ ($y_1 < y_2$) is the set of points (x, y), for which at least one of the two conditions is fulfilled:

- $min(x_1, x_2) \le x \le max(x_1, x_2)$ and $y \ge y_1$
- $y_1 \le y \le y_2$ and $(x - x_2) \cdot (x_1 - x_2) \ge 0$

The pictures showing two different corners

In order to test the spell, the wizards will apply it to all the cities that lie on the x-coordinate in the interval $[L, R]$. After the construction of roads the national government wants to choose the maximum number of pairs of cities connected by the road, so that no city occurs in two or more pairs. Your task is for each m offered variants of values L, R to calculate the maximum number of such pairs after the construction of the roads. Please note that the cities that do not lie in the interval $[L, R]$ on the x-coordinate, do not affect the construction of roads in any way.

Input

The first line contains two space-separated integers n, k ($1 \le k \le n \le 10^5$, $k \le 30$). Next k lines contain coordinates of the cities' location points from the first to the k-th one. The j-th line contains space-separated pair of integers x_j, y_j ($0 \le x_j, y_j < 10^9 + 9$) — coordinates of the j-th city.

The next line contains space-separated integers a, b, c, d ($2 \le a, b, c, d < 10^9 + 9$). It is guaranteed that those numbers are prime and also that $a \ne c, b \ne d$.

It's guaranteed, that for every two different cities i and j, $x_i \ne x_j$ and $y_i \ne y_j$ holds.

The next line contains integer m ($1 \le m \le 10^5$) — the number of variants to build the roads. Next m lines contain pairs of space-separated integers L_i, R_i ($0 \le L_i \le R_i < 10^9 + 9$) — the variants of choosing the cities to build the roads.

Output

For any pair of numbers L_i, R_i print the answer to the problem on a single line. Print the answers for the pairs in the order, in which the pairs are given in the input data.

Examples

input

```
6 6
0 0
1 1
2 2
3 3
4 4
5 5
2 3 3 2
4
0 5
1 4
2 3
3 3
```

output

```
3
2
1
0
```

input

```
6 1
0 0
3 5 23917 11
4
0 1000000008
0 10
100 150
200 10000
```

output

```
2
1
0
1
```

Note

In the first sample the roads connect the cities in a chain in the order of increasing of x.

In the second sample the remaining 5 cities will be located at points $(5, 11); (20, 263098); (65, 292514823); (200, 76958738); (605, 622120197)$.

E. Wizards and Bets

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

In some country live wizards. They like to make weird bets.

Two wizards draw an acyclic directed graph with n vertices and m edges (the graph's vertices are numbered from 1 to n). A *source* is a vertex with no incoming edges, and a *sink* is the vertex with no outgoing edges. Note that a vertex could be the sink and the source simultaneously. In the wizards' graph the number of the sinks and the sources is the same.

Wizards numbered the sources in the order of increasing numbers of the vertices from 1 to k. The sinks are numbered from 1 to k in the similar way.

To make a bet, they, as are real wizards, cast a spell, which selects a set of k paths from all sources to the sinks in such a way that no two paths intersect at the vertices. In this case, each sink has exactly one path going to it from exactly one source. Let's suppose that the i-th sink has a path going to it from the a_i's source. Then let's call pair (i, j) an *inversion* if $i < j$ and $a_i > a_j$. If the number of inversions among all possible pairs (i, j), such that $(1 \le i < j \le k)$, is even, then the first wizard wins (the second one gives him one magic coin). Otherwise, the second wizard wins (he gets one magic coin from the first one).

Our wizards are captured with feverish excitement, so they kept choosing new paths again and again for so long that eventually they have chosen every possible set of paths for exactly once. The two sets of non-intersecting pathes are considered to be different, if and only if there is an edge, which lies at some path in one set and doesn't lie at any path of another set. To check their notes, they asked you to count the total winnings of the first player for all possible sets of paths modulo a prime number p.

Input

The first line contains three space-separated integers n, m, p ($1 \le n \le 600$, $0 \le m \le 10^5$, $2 \le p \le 10^9 + 7$). It is guaranteed that p is prime number.

Next m lines contain edges of the graph. Each line contains a pair of space-separated integers, a_i b_i — an edge from vertex a_i to vertex b_i. It is guaranteed that the graph is acyclic and that the graph contains the same number of sources and sinks. Please note that the graph can have multiple edges.

Output

Print the answer to the problem — the total winnings of the first player modulo a prime number p. Please note that the winnings may be negative, but the modulo residue must be non-negative (see the sample).

Examples

input

```
4 2 1000003
1 3
2 4
```

output

```
1
```

input

```
4 2 1000003
4 1
3 2
```

output

```
1000002
```

input

```
4 4 1000003
2 1
2 4
3 1
3 4
```

output

```
0
```

input

```
6 5 1000003
1 4
1 5
1 6
2 6
3 6
```

output

```
0
```

input

```
5 2 1000003
5 1
3 4
```

output

```
1
```

Note

In the first sample, there is exactly one set of paths — $(1 \to 3), (2 \to 4)$. The number of inversions is 0, which is an even number. Therefore, the first wizard gets 1 coin.

In the second sample there is exactly one set of paths — $(4 \to 1), (3 \to 2)$. There is exactly one inversion. Therefore, the first wizard gets -1 coin. $-1 \bmod (10^6 + 3) = 10^6 + 2$.

In the third sample, there are two sets of paths, which are counted with opposite signs.

In the fourth sample there are no set of paths at all.

In the fifth sample, there are three sources — the vertices with the numbers (2, 3, 5) and three sinks — the vertices with numbers (1, 2, 4). For a single set of paths $(5 \to 1), (3 \to 4), (2 \to 2)$ are 2 inversions, that is, their number is **even**.