Problem 1. Passage

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

> *Once upon a time a farmer went to a market and purchased a fox, a goose, and a bag of beans. On his way home, the farmer came to the bank of a river and rented a boat. But in crossing the river by boat, the farmer could carry only himself and a single one of his purchases - the fox, the goose, or the bag of beans. If left together, the fox would eat the goose, or the goose would eat the beans. The farmer's challenge was to carry himself and his purchases to the far bank of the river, leaving each purchase intact. How did he do it?*
>
> (An old puzzle)

Afanasiy and Anatoliy work as the zoo caretakers at the famous Novosibirsk Zoo, and they have to solve problems much worse than the rescue of beans and a goose. For example, once two of them had to carry across the Ob river all zoo animals in two boats! You are invited to feel yourself in their skins.

There are N animals in the zoo. If you leave some of the animals together unattended, it is very likely that one will eat another. It is known for each animal what other animals it can eat. Each zoo caretaker is a great expert in his work and can deal even with a tiger! When there is at least one caretaker near the animal, it definitely cannot eat anyone.

Zoo caretakers have two boats. According to the terms of insurance, it is forbidden to put two or more animals into the same boat at the same time. In any boat at any given time, there may be no more than one caretaker and no more than one animal. Initially, all the animals, caretakers, and boats are located on the left bank of the river. All of them must be transfered to the right bank of the river. Of course, there are no other banks of the river, except for the left and right ones.

Caretakers can float on boats between the two banks as they please. Both boats are rowing, so they cannot move without a caretaker. If at any point, there is such a pair of the animals on the same bank that one of them can eat the other one, and there is no caretaker on this bank, then something irretrievable happens. Of course, this must be avoided: all the animals should get to the right bank of the river safe and sound.

You need to determine whether it is possible to carry out the crossing.

Input

In the first line of the input file there is one integer N, which is the number of animals in the zoo ($2 \le N \le 200$). Each of the animals has an inventory number ranging from 1 to N inclusively.

The following N lines contain information about gastronomic passions of the animals. i-th of them describes which animals can eat the animal with the inventory number i ($1 \le i \le N$). The line begins with a non-negative integer k_i, being the number of animals which can eat the animal i, followed by k_i positive integers, which are the inventory numbers of those ravenous animals. These

numbers are different and lie in the range from 1 to N. None of these numbers can be equal to i because no animal can eat itself.

The total number of "predator-prey" pairs, which is equal to the sum of all numbers k_i, does not exceed 1500.

Output

If it is possible to carry across all animals safe and sound, print "`Hurrah!`". If not, print "`Fired.`".

Examples

input.txt	output.txt
5 1 2 1 3 1 4 0 0	Hurrah!
5 4 2 3 4 5 4 3 4 5 1 4 4 5 1 2 4 5 1 2 3 4 1 2 3 4	Fired.

Problem 2. Files list

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

You are given a set of files, and you are to output the number of files with each extension. The file extension is a sequence of characters in the name of the file after a dot character.

The file system is case-sensitive: even if the file names differ only by a characters case, they are still considered to be different.

Input

In the first line of the input file there is an integer N, which is number of file names ($1 \le N \le 10^3$). In each of the following N lines there is a file name that is no more than 200 characters in length. The file name consists of only lower and upper Latin letters, numbers and a dot character '.'. It is guaranteed that a dot character can be found in the file name exactly once. Also, there is at least one symbol before and after the dot. It is guaranteed that each file name is present only once in the input file.

Output

For each of the extensions that are present in the input file, output the number of files with this extension in the form of <extension>: <number>. Output extensions in order of the first mention in the input file.

Example

input.txt	output.txt
6 218052.pdf Movie00.mkv Invoice.xls book.pdf book.epub Movie01.mkv	pdf: 2 mkv: 2 xls: 1 epub: 1

Example explanation

Two files with extension pdf: 218052.pdf, book.pdf.

Two files with extension mkv: Movie00.mkv, Movie01.mkv.

One file with extension xls: Invoice.xls.

One file with extension epub: book.epub.

Problem 3. Graph optimization

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	3 seconds
	6 seconds (for Java)
Memory limit:	256 megabytes

Answer to the Ultimate Question of Life, The Universe, and Everything: ***32***

Kondratiy loves to optimize code. He enjoys this activity so much that he could even be charged for the opportunity to optimize! Unfortunately, he does not notice that 90% of his code would run hundreds of times faster, if he used more efficient algorithms instead of writing inline assembly.

Kondratiy recently met a girl Vsemila. Vsemila adores solving algorithmic problems and can give up on anything for the opportunity to solve more. Of course, Kondratiy generally does not like her code, yet he does not complain, because for the first time in his life he has met a girl who programs so well!

One day a holy war about the true beauty happened between the friends. As a result, they bet who could solve more efficiently the first problem the bump into: either Vsemila with her suffix automata and min-cost flows, or Kondratiy with his hatred towards branches and his love of AVX-512. Unfortunately for Kondratiy, they bumped into a problem related to graph construction. Raw computing power cannot help him here, some thought is needed to solve it. So he asks you to help him solve the problem, because he can not lose the primacy in such an important thing as programming to a girl!

Here is the problem Kondratiy and Vsemila are solving. It is necessary to construct a directed graph with n vertices, satisfying a given set of requirements.

There are two types of requirements:

1. From vertex a it is possible to get to vertex b.
2. From vertex a it is not possible to get to vertex b.

If there are no such graphs, it must be determined and reported.

Input

The first line of the input file contains an integer n, which is the number of vertices in the desired graph ($1 \le n \le 10^5$).

The second line contains an integer k, being the number of reachability requirements ($0 \le k \le 10^5$). In each of the following k lines there are two integers a_i and b_i, meaning that from the vertex a_i it should be possible to get to the vertex b_i by edges of the graph ($1 \le a_i, b_i \le n$).

In the following line there is an integer p — the number of unreachability requirements ($0 \le p \le 10^5$). In each of the following p lines there are two integers c_j and d_j, meaning that from the vertex c_j it should be impossible to get to the vertex d_j ($1 \le c_j, d_j \le n$).

Output

If there is no such graph, the output file should contain only the word `NO`.

Otherwise, the first line of the output file should contain the word YES, and the second line must contain the number of edges m ($0 \le m \le 3 \cdot 10^5$). The following m lines should contain descriptions of the edges, one per line. Each edge is described in format "x_i y_i", where x_i is the head vertex of the edge and y_i is its tail vertex.

Multiple edges and self-loops are allowed. It is guaranteed that if there exists a graph satisfying all the requirements given in the input file, then there is a graph with no more than $3 \cdot 10^5$ edges among them.

Examples

input.txt	output.txt
2 1 1 2 1 1 2	NO
2 1 1 2 0	YES 1 1 2

Problem 4. Housing payments

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

Being a decent citizen, Ippolit Matveevich pays his monthly housing payments the same day he receives his receipt. When he got to the bank recently, he found out that a new tariff system was implemented.

For now on he has to pay extra X_i rubles, if he pays during the i-th month, as commission to the bank, regardless of the actual sum being payed. At first, it seems like if he pays rarely, then he will save money on commission. But the system works smarter: after each month the total amount of debt increases by p_i percent. Therefore, it is the choice: either to pay rarely with smaller commissions but bigger fine, or to pay more regularly with smaller fines but bigger commission payments.

Help Ippolit Matveevich to find the way to minimize the total sum of his payments.

Input

In the first line of the input file there is one integer N, being number of months for which he has to make payments ($1 \le N \le 10^5$).

In each of the following N lines there are three integers S_i, x_i, p_i, where S_i is the principal amount of the payment received in a receipt on the i-th month, x_i is the commission for payments on i-th month, and p_i is the percentage by which the sum of debt will increase at the end of the month ($10^3 \le S_i \le 10^4, 10 \le x_i \le 500, 1 \le p_i \le 10$).

Output

The output file should contain one real number, being the minimum possible amount of money Ippolit Matveevich has to pay for the entire period. The answer must be printed with an absolute or relative error not greater than 10^{-8}.

Examples

input.txt	output.txt
3 5000 20 1 5000 20 1 5000 20 1	15060
3 5000 500 1 5000 200 1 5000 100 3	15250.5

Example explanation

In the first case, it is more favorable to pay each month. In the second one paying once at the end of three months is cheaper.

Problem 5. Arithmetic expressions

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	5 seconds
Memory limit:	256 megabytes

Pupil Peter composes arithmetic expressions from the symbols `'0'-'9'`, parentheses `'('` and `')'`, and symbols `'+'` and `'-'` (plus and minus signs). Each composed expression can be either a string containing the decimal representation of an integer (without leading zeros) lying in the range from 0 to $M-1$ inclusive, or a string of the form (E), where E is also an expression, or a string of the form $E_1 \sigma E_2$, where E_1 and E_2 are also expressions and σ is a symbol `'+'` or `'-'`.

Peter wants to count the number of different expressions of the fixed length N, for which the remainder after dividing the value by M is equal to P. The value of an arithmetic expression is calculated by the school rules of arithmetic. Please note: the remainder of the division is always within the range from 0 to $M-1$, even if the value is negative.

Since the answer can be large, you should print its remainder modulo 10^9+7.

Input

The first line of the input file consists of three integers N, M and P, where N is length of expressions, M is modulus used for division, P is required remainder of the division ($1 \le N \le 50$, $0 \le P < M \le 200$).

Output

The output file should contain one integer — the number of all expressions that have given length and given value of the remainder of division. You should output the answer modulo 10^9+7.

Examples

input.txt	output.txt
3 100 2	12
3 100 98	8
2 100 98	1

Example explanation

All sought-for expressions for the case $N = 3$, $M = 100$, $P = 2$:

```
(2)   0+2   1+1   2+0   2-0   3-1   4-2   5-3   6-4   7-5   8-6   9-7
```

All sought-for expressions for the case $N = 3$, $M = 100$, $P = 98$:

```
0-2   1-3   2-4   3-5   4-6   5-7   6-8   7-9
```

All sought-for expressions for the case $N = 2$, $M = 100$, $P = 98$:

```
98
```

Problem 6. Sputnik

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

The "New Zenith" probe is on a circular orbit around a newly discovered dwarf planet in the Kuiper belt. The panoramic camera of the probe registered an anomaly on the planet surface, presumably an active cryovolcano, at the location with the latitude ϕ and longitude λ. Scientists need a second image of this area taken with a high-resolution camera.

The fuel reserves of the probe are limited, so before planning the orbit correction, the mission control center needs to define when (and whether) the probe will pass above the required area staying on its current orbit. Luckily, to get satisfactory images it's not necessary for the probe to pass exactly above the point of interest. It is enough that the probe passes over any point within the circle with the center at (ϕ, λ) and angular radius of δ degrees.

The probe moves along the orbit with constant angular velocity and a rotation period of t seconds. The orbit inclination (the angle between the orbit plane and the planet's equator plane) is α degrees. The longitude of the ascending node (the point where the probe orbit intersects with the equator when moving in South-North direction) at the starting moment is zero.

The plane of the probe orbit passes through the center of the planet and is stationary relative to it. The planet rotates around the axis perpendicular to the equator plane at constant angular velocity and with a rotation period of T. The rotations of the planet and of the probe are prograde (i.e. both the satellite and the points on the planet surface move from West to East). The angular speed of the planet is constant in the equator plane, the angular speed of the probe is constant in the plane of the orbit.

At the initial moment the probe is located over a point with the latitude ϕ and longitude λ. The probe can perform the command to take a high-resolution photograph not earlier than after time $2t$ from the initial moment.

If it's impossible to take a photograph earlier than in T_{max} seconds from the initial moment (i.e. the probe doesn't fly through the vicinity of the point of interest), the orbit must be corrected.

Input

The first line of the input file contains seven real numbers: t, T, T_{max}, α, ϕ, λ, and δ ($60 \le t \le 1\,000$, $3\,600 \le T \le 100\,000$, $10\,000 \le T_{max} \le 1\,000\,000$, $0 \le \alpha \le 60$, $-60 \le \phi \le 60$, $0 \le \lambda < 360$, $0.1 \le \delta \le 1$).

All values are given with no more than 15 decimal digits after the decimal point. The time parameters (t, T, T_{max}) are set in seconds. The angular parameters $(\alpha, \phi, \lambda, \delta)$ are set in degrees.

Output

If during the time interval from $2t$ to T_{max} from the initial moment the probe is located at an angular distance of δ or less from the point (ϕ, λ), then the output file must contain the earliest moment of time τ when this happens. The answer must be precise up to relative or absolute error 10^{-7}. In the other case, print `-1`.

It is guaranteed that a change of δ by 10^{-3} degrees in any direction would result in a change of the answer by less than a second.

Example

input.txt	output.txt
120 4500 100000 60 30 19.471220634490699 0.5	8999.83111211

Example explanation

All the numbers in the input file are written on a single line (space-separated). In the current text the numbers are shown on two separate lines, because they didn't fit into a single one.

Problem 7. Voting

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

In Berland presidential elections occurs by a complex multilevel hierarchical system. The opposition believes that this electoral system was invented to confuse the public and to conceal the unfairness of voting. This year one candidate from the opposition is running for president. If he wins, there would be a real opportunity to change the political situation.

The simplest representation of the voting system is a rooted tree. Leaves of this tree correspond to individual voters. Each internal node is an official social group of some kind (family, home, neighborhood, district, etc.). Children of a node correspond to the entities that directly belong to this official social group: individual voters and/or other groups. The root node of the tree is the group including all the citizens of Berland.

Each of the voters may either vote for a particular candidate or refrain from voting, for example, by not coming to the election.

After every voter has voted one way or another, for each group the local election results are determined, in order from smaller to larger groups. To do this, the votes of all children entities of this group are considered. A candidate wins in the local election if he receives strictly more votes than any other candidate. If there is no such winning candidate, then the local election has failed. Further, the results of voting in the group are taken into account in the local voting on the next level as a single vote from the whole group. If there was no winner in local election, the group is considered "abstained", and its opinion is not taken into account any further.

A candidate who has won in the local election of the highest group (i.e. in the tree's root) becomes Berland's President. If there is no election winner in this group, then the position of the president in Berland is abolished until the next election.

For each individual voter it is known for whom he is going to vote. The opposition can change the preference for any voter in any way desired, but it requires some effort. It is necessary to determine the minimum number of voters whose opinions are to be changed in order for the opposition candidate to become the president.

Input

The first line of the input file contains integers N, M, K, where N is number of individual voters, M is number of social groups and K is number of candidates ($2 \le N \le 5\,000$, $1 \le M \le 5\,000$, $2 \le K \le 3$). Voters, groups and candidates are individually numbered starting from one. The number of the opposition candidate is 1.

The second line contains N integers corresponding to initial voters' opinions. Opinion is either equal to the number of the candidate for whom the voter intends to vote, or to zero, if he abstains from voting.

The following M lines describe social groups. The t-th of the lines describes entities that belong to the social group t. It starts with a positive integer S_t, which is the number of these entities, followed by a space-separated list of S_t integers which define these entities ($2 \le S_t \le 20$). If an entity is an individual voter, his number is given. If it is a group, then its number with a minus sign is given. The group at the tree's root, which contains all the citizens of Berland, always has

number 1.

Output

The first line should contain a single integer R equal to the minimum number of voters whose plans need to be changed for the victory.

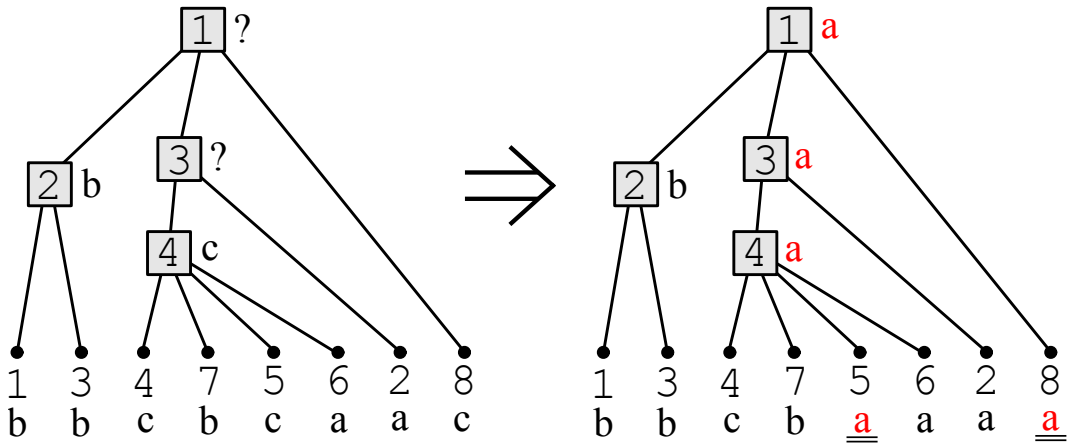
In each of the following R lines, two integers a and v must be written, where a is the number of the voter, whose plans need to be changed, and v is either the number of the candidate for whom this voter should vote, or 0 if the voter should not come to the election. If the opposition candidate wins without any changes, the output file should contain a single number 0.

Example

input.txt	output.txt
8 4 3 2 0 2 1 1 3 2 1 3 -2 -3 8 2 1 3 2 -4 2 4 4 7 5 6	0
8 4 3 2 1 2 3 3 1 2 3 3 -2 -3 8 2 1 3 2 -4 2 4 4 7 5 6	2 8 1 5 1

Example explanation

The voting tree and the proposed changes for the second example are shown on the picture below. Letters a, b, c denote votes for candidates 1, 2, 3 respectively. The question mark denotes an "abstained" vote.



Problem 8. Novice urbanist

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	1.5 seconds
Memory limit:	256 megabytes

Lately New State University (NSU) is being constructed rapidly. New buildings, dormitories and housings keep appearing here and there along the legendary Student Avenue. Vasya enjoys walking along his favorite avenue, however, the problem is that different buildings of NSU are located on opposite sides of the street. In order to get from one building to another it is sometimes necessary to have a long walk to the nearest crosswalk. So he decided to write a computer program to understand how to move the crosswalks along the Student Avenue in the most beneficial way for pedestrians. He wants as many crosswalks in front of NSU buildings as possible, at the same time the move of crosswalks should be minimal. Help Vasya to write a program, as he is also preparing an appeal to the mayor of the city and needs trustworthy calculations.

Here's how Vasya drew the plan of the avenue and NSU buildings before writing the program. The Student Avenue is represented by a straight line. Crosswalks are regarded as points on this line. All the buildings of NSU stand parallel to the avenue, so you can consider them to be segments on the line. Each of the segments has its left and right boundaries. A crosswalk is located in front of a building if the corresponding point on the Avenue is located between the left and the right borders of the building (including the boundary points). Since the crosswalks have been established in compliance with certain standards, Vasya decided to keep the distances between them, so he wants to move all the crosswalks by the same distance.

Input

In the first line of the input file there are two integers n, m, where n is the number of crossings on the Student Avenue, m is the number of NSU buildings ($1 \le n \le 10^4$, $1 \le m \le 10^3$).

In the second line there is a list of integers a_i ($1 \le i \le n$), specifying the coordinates of crosswalks on the avenue ($0 \le a_i \le 10^6$).

The third line contains m pairs of integers l_i, r_i ($1 \le i \le m$), each pair describes the coordinates of the boundaries of a building ($0 \le l_i < r_i \le 10^6$).

Output

In the output file print two nonnegative integers: the distance by which you should shift the crosswalks, and the number of crosswalks that will be in front of any building as a result of the shift. The second number (number of crosswalks) must be the maximum possible. If the answer is not uniquely determined, you have to additionally minimize the first number (the shift). Take into account that it is possible to move the crosswalks in any of the two directions.

Examples

input.txt	output.txt
3 1 1 2 4 5 6	4 2
4 2 1 6 6 1 4 5 3 5	1 2

Problem 9. Rangefinder

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	5 seconds
Memory limit:	256 megabytes

At the World Championships of Robotics in 3175, participants are creating a robot, which performs various tasks in a maze. Exclusively for the MCLXXVI Open All-Siberian Pottosin Olympiad, the jury has decided to make a virtual version of the competition allowing ordinary programmers, not skilled enough to assemble robots, to participate.

The maze is located on a rectangular grid, and has size W by H. There are no obstacles inside the cells of the maze, so a robot can easily be located there. However, there may be walls between cells; nothing can pass through them. Each wall has two sides (that would be important below). The maze is always fully surrounded by walls and the robot cannot get out of it.

Modern technologies allow to create portals in the walls. Any portal connects one side of a wall with another side of the same or another wall. When any particle dashes into the wall from the side, where the entrance of the portal is located, the particle dashes out of the other wall, which has a paired entrance of the portal, from the side of that entrance. Note that the particle always dashes in and out of a portal perpendicular to the surface of the wall, and, hence, the direction of the particle may change as a result of flying through the portal. An example of a flight through a portal is shown in the picture at the end of the statement of this problem.

To make it more interesting to perform tasks in the maze, its configuration changes during the competition. Initially there are no walls inside the maze, though there are walls along its border. At any moment, a new wall may appear on the grid between some cells or an old wall can suddenly disappear. Moreover, a portal might appear, which connects two sides of the walls that do not have portals on them, or an old portal can disappear. When a portal disappears, the walls, where its entrances were located, remain in place, only their two sides get free of the entrances.

If a particle can freely fly along a certain path, it can also fly freely along the same path in the opposite direction, which means that all the portals are bidirectional. However, if the entrance of the portal is located at the wall, the particle can dash in it from one side only. Moreover, there could be portal entrances on both sides of the same wall: they could be entrances of two different portals or two paired entrances of a single portal.

The robot model that will be used by programmers in the virtual competition is equipped with a rangefinder. To use the rangefinder, it must be placed into a cell of the maze and be directed parallel to a side of the maze. When the rangefinder is used, a ray is launched from it that flies directly to the first contact with an impassable wall. If the ray hits a portal, it passes through by the above rules, and it not considered as a contact with the wall. As a result of using the rangefinder, the programmer should be told how many cells the ray flew before it hit a wall. It should be noted that this number could be infinite if the released ray is flying circles. The robot itself and its rangefinder are transparent and do not affect the flight of the ray.

The jury of the championship are more than enough skilled to assemble robots by their hands but they are not as good in algorithmic programming. They need to implement the program, which at any moment can simulate the use of the rangefinder and report the distance traveled by the ray. They were able to write a program that does it, but because of the huge scale of the competition, the program works too slowly. Therefore, they are asking for your help.

Input

In the first line of the input file, there are three integers H, W and Q: H is height of the maze in cells, W is width of the maze in cells, Q is number of queries ($1 \le H \le 10^6$, $1 \le W \le 10^6$, $1 \le Q \le 2 \cdot 10^5$).

In each of the other Q lines, there is only one query. The queries are listed in order of their execution.

Two cells of the grid are called adjacent when they share a common side. Each cell has its coordinates (r, c), where r is number of the row, and c is number of the column on the grid ($1 \le r \le H$, $1 \le c \le W$). On the grid, you can define four coordinate directions, each of which is denoted by a single letter:

- `D` — down, with r increasing,
- `U` — up, with r decreasing,
- `R` — to the right, with c increasing,
- `L` — to the left, with c decreasing.

The coordinates of a cell (r, c) together with a direction d define the position of a potential wall: it is located on the boundary between the cell with coordinates (r, c) and the adjacent cell, which can be reached by taking a step out of the cell (r, c) in the direction of d. Moreover, the particular side hit when taking this step is called the *nearest* side of the wall.

The query can be one of the following five types:

- Adding or deleting the wall: *type* r c d

  Here (r, c, d) together determine the position of the potential wall following the rule described above, and *type* determines the type of query. If *type* equals to `W+`, then the new wall appears at this place. If *type* equals to `W-`, then the old wall disappears at this place. It is guaranteed that at the time of removal of a wall, there are no portal entrances on its sides. The walls surrounding the maze never disappear.

- Creating or deleting of the portal: *type* r_1 c_1 d_1 r_2 c_2 d_2

  Here (r_1, c_1, d_1) together determine the position of the first wall following the rule described above, and (r_2, c_2, d_2) — the position of the second wall. It is guaranteed that in these two places there are walls for sure. Entrances to the portal are located on the *nearest* sides of these walls. It is guaranteed that the two sides are different, although the wall may be the same.

  type determines the type of query. If *type* equals to `P+`, then these two sides get connected with a new portal, it is guaranteed that there were no portal entrances on these sides just before the query. If *type* equals to `P-`, then the portal between these two sides disappears, it is guaranteed that there was a portal which connected these two sides just before.

- Using the rangefinder: `M?` r c d

  Rangefinder is put into a cell of the maze with the coordinates (r, c), and directed in the direction of d. Determine how many cells the ray will fly when used.

All cell coordinates are within the maze, each direction is given by one of the four letters described above.

Output

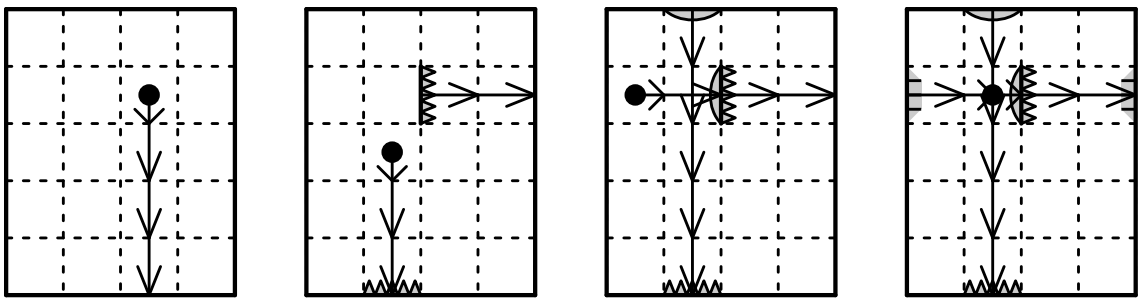
For each rangefinder using query, display the result of its use, i.e. the number of cells the ray flies before hitting a wall. If the ray flies forever, output number -1 as an answer to the query. Give the answers to the queries in order of their execution.

Example

input.txt	output.txt
5 4 18 M? 2 3 D W+ 2 2 R W- 2 3 L W+ 2 2 R M? 2 1 R P+ 2 3 L 5 2 D M? 3 2 D M? 2 3 L M? 2 2 R P+ 2 2 R 1 2 U M? 2 1 R P+ 2 1 L 2 4 R M? 2 2 R P- 1 2 U 2 2 R M? 2 2 L W+ 4 2 U P+ 4 2 U 3 2 D M? 5 2 U	3 1 4 5 0 8 -1 8 4

Example explanation

The picture below shows several usages of the rangefinder: the first, the third, the sixth, and the seventh in the input file.



Problem 10. Hive

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

There is a plane, which is tiled with regular hexagons. There are honeycombs on this plane. Worker bees at first misunderstood the project documentation, and now they have to turn beehive honeycombs around queen bee by 60 degree clockwise.

Beehive consists of hexagonal honeycombs, which are oriented in such manner that there are hexagon nodes below and above, and there are edges to the left and right which the honeycomb shares with its adjacent honeycombs in the row. Every consequent row is shifted relative to the previous row by half a honeycomb. The Ox axis goes from left to right along the horizontal row of honeycombs. The Oy axis is inclined 60 degrees relative to the Ox axis. The axes intersect at the honeycomb with coordinates $(0, 0)$. Example explanation contains illustrations showing the tiles numeration.

Input

In the first line of the input file there are three integers N, X and Y, where N is number of honeycombs in the hive (excluding the honeycomb of a queen bee), X and Y are coordinates of the honeycomb of a queen bee ($1 \le N \le 10^4; |X|, |Y| \le 10^4$). In each of the following N lines there is a pair of integers x and y, being the coordinates of a honeycomb of the hive ($|x|, |y| \le 10^4$). The coordinates of all honeycombs in the input file are different.

Output

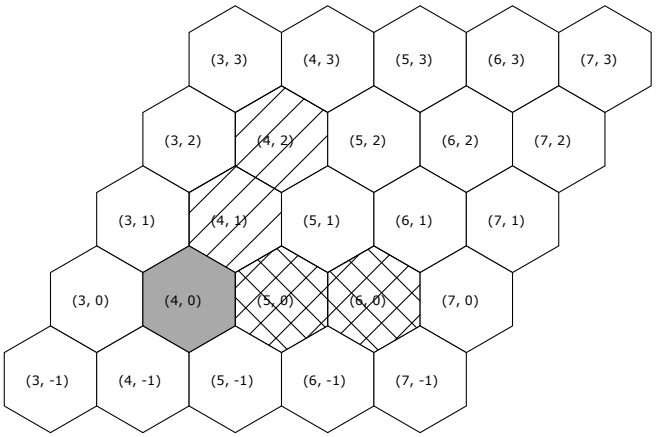
For each of N honeycombs, you should output two integers on the separate line: its coordinates after the turn. The coordinates must be ordered as they are in the input file.

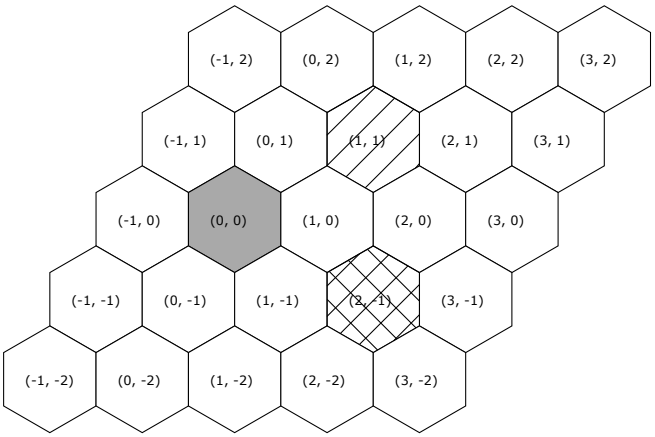
Example

input.txt	output.txt
2 4 0 4 1 4 2	5 0 6 0
1 0 0 1 1	2 -1

Example explanation

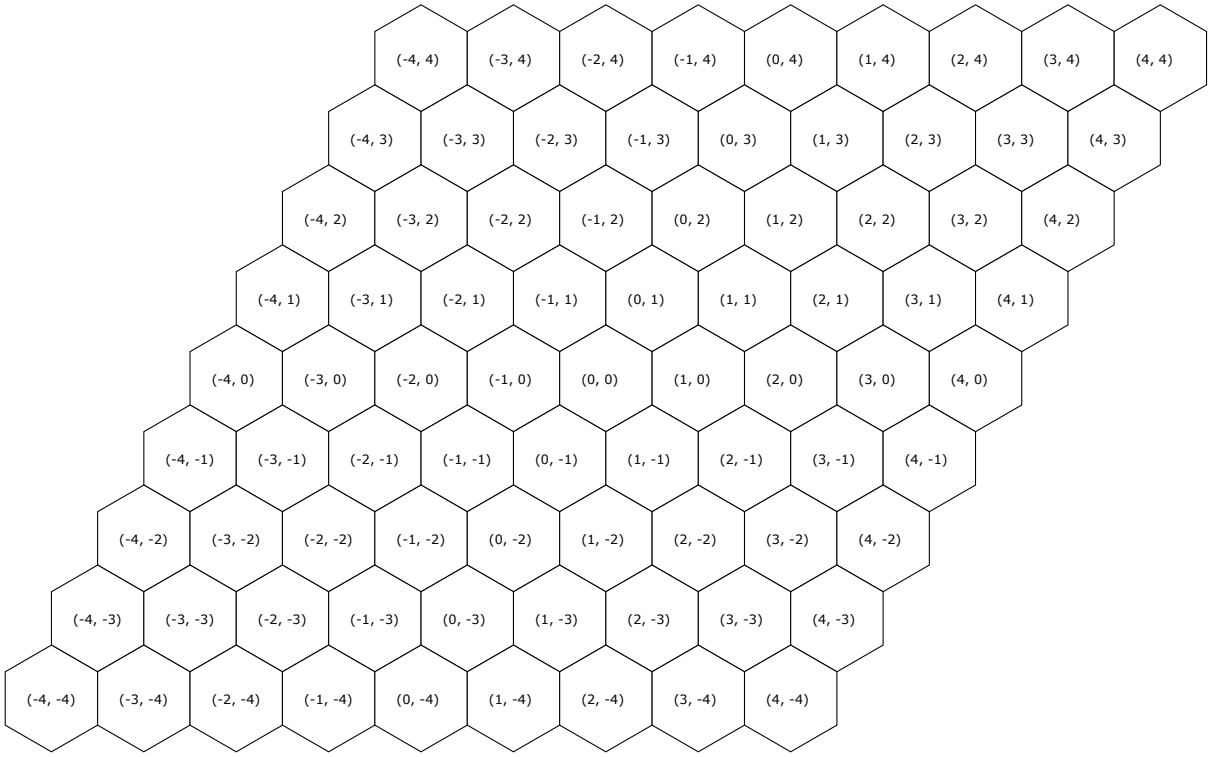
Hives from the examples are presented in the figures below. The honeycomb of the queen bee is marked in gray, honeycombs in their initial position are shaded with parallel lines, honeycombs after the turn are shaded with intersecting lines.





Illustration

One may draw on this hive. ;-)



Problem 11. Side effects

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

The programmer is developing an application. Being a part of a club "Young Friends of Functional Programming", he wants to know how many of its functions have side effects. Initially, for each function of the program it is known, whether it has a side effect or it has not; also, it is known that no function calls any other. Yet the programmer decides to change the logic of the application and starts to put some calls of the functions to other functions. The programmer does not write new functions. If a function calls a function with side effects, then the caller function also starts to have side effects (and so on in the chain of calls). Recursion in calls is allowed. Also, one function can call another several times. You need to determine how many functions with side effects are present in the program after each addition of a function call by the programmer.

Input

The first line of the input file contains three integers N, K and M, where N is total number of functions, K is initial number of functions with side effects, M is number of function calls added by the programmer ($1 \le N, K, M \le 10^5; K \le N$). Functions are numbered in order from 1 to N.

Next, there are K different numbers ranging from 1 to N in one line, being indices of functions which have side effects initially. In each of the following M lines there is a pair of integers a and b, which means that the programmer has added the call of function with the number b from the function number a ($1 \le a, b \le N$).

Output

For each of M additions of function calls, print the number of functions with side effects at the time after the addition of this call.

Example

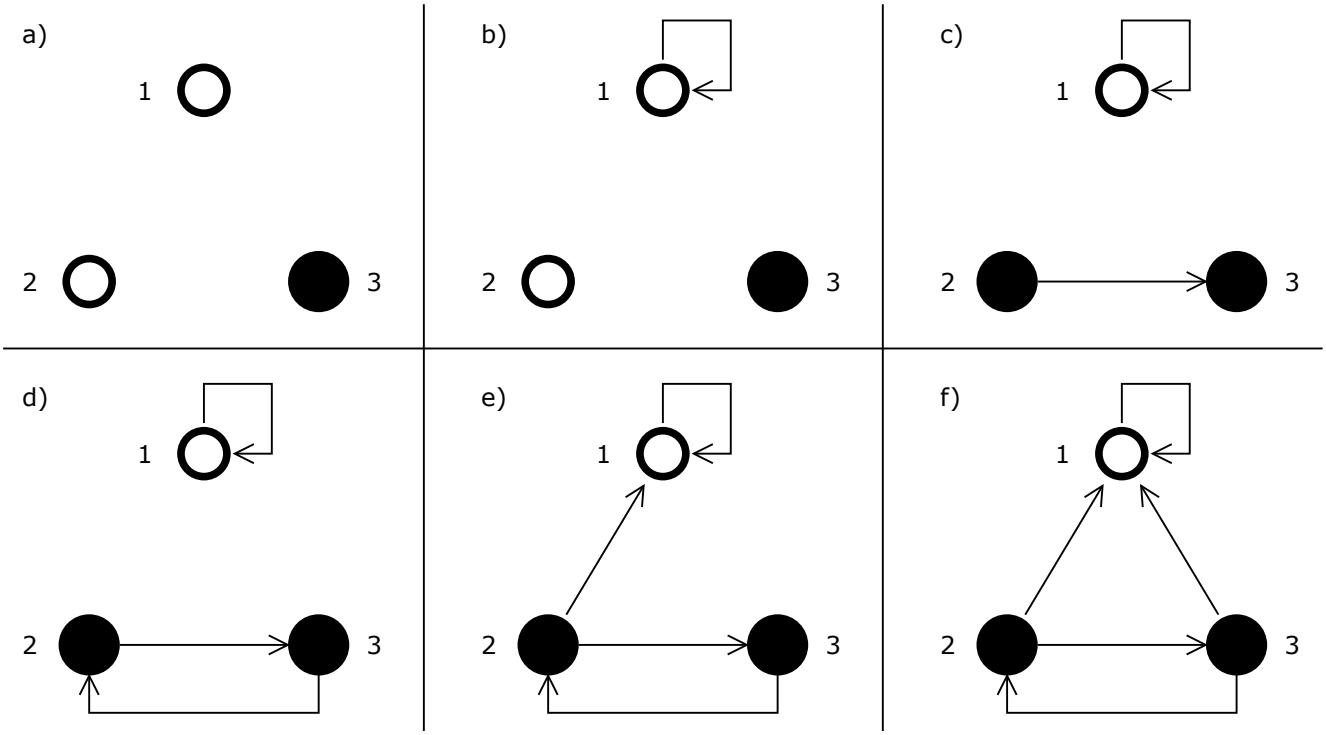
input.txt	output.txt
3 1 5 3 1 1 2 3 3 2 2 1 3 1	1 2 2 2 2

Example explanation

The picture a) shows functions before adding calls. Pictures b) - f) show consecutive additions of function calls. The white circle with black outline is a function with no side effects, black circle is a function with side effects, the arrow shows a function call.

Firstly, the recursive function call is added to the function 1 from itself, from this action, obviously, the answer will not change. After adding a call to the function 3 from the function 2, the program

starts to have two functions with side effects: 2 and 3 because the function 3 initially had side effects. Subsequent additions of calls do not increase the number of functions with side effects.



Problem 12. Cypher

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

Boris has created a new cypher machine "SumUp" for personal use. The machine is given an integer x for the input, machine performs the sequence of operations according to the program recorded in it, and gives an integer y to the output. Anna knows the program inside the machine and wants to learn how to restore the input number x by the output number y.

The machine is set up to work with integers in the range from 0 to $M-1$ for some positive integer M. The machine has k registers that are numbered with indexes from 0 to $k-1$, and are denoted by `R0`, `R1`, `R2` and so on. Before the program starts working, the value of the register `R0` is equal to the input number x and all other registers are equal to zero. After performing all the operations of the program, the value of the register `R0` is issued out of the machine as an output number y.

The program consists of the sequence of commands that are performed in the order they were recorded. There are only two types of commands:

1. `add` *regD regS* — add the value of source register *regS* to the value of destination register *regD*.
2. `inc` *regD* — increase the value of destination register *regD* by one.

At any moment the value of any register is an integer within the range from 0 to $M-1$ inclusive. If an overflow occurs during the command execution, the remainder after dividing the result by M is recorded to the destination. The input number x must belong to the range from 0 to $M-1$, otherwise the machine will not operate. Clearly, the output number y must also lie within these limits.

When Anna accidentally got physical access to the machine, she managed to take a picture of the program used in the machine. However, the number M, which indicates the working range of the machine, is not seen in the picture. Anna only remembers how many digits are used in this number as well as some of those digits. In addition, Anna knows the output value y, openly distributed by Boris.

Anna uses a brute force algorithm to solve her problem. She iterates over all the possible numbers M and x. For each pair she checks whether the machine will produce y as an output value, if she sets the operating range of the machine to M and the input number to x. Each time when the value y is successfully obtained, she adds the number x to the piece of paper.

You need to calculate the sum of all the numbers Anna has written on the piece of paper after the algorithm has finished. Since this sum can be large, print its remainder modulo prime number $10^9 + 7$.

Input

In the input file, there are several independent tests written. The first line consists of T — the number of tests in the file ($1 \le T \le 10$). The rest of the file consists of descriptions of T tests.

Each description starts with the line of three integers k, m and y, where k is the number of registers in the machine, m is the number of commands in the program, y is the output number, which must be obtained ($1 \le k \le 10$, $0 \le m \le 15$, $0 \le y < 10^{15}$).

The commands of the program are given in the following m lines in order of their execution, one command per line.

Each command consists of two or three parts, the parts are separated by a single space. The first part indicates the type of command (`add` for addition, `inc` for the increment), the second part — a destination register being modified, and the third (if exists) — a source register being added. A register is denoted with a Latin letter `R` and register's index immediately after it, which is within the range from 0 to $k - 1$ inclusive.

The last line gives the information about the number M. It contains a single string consisting only of decimal digits and question marks. A question mark indicates that there can be any decimal digit in its place. The string is nonempty, its length does not exceed 15 characters. It is guaranteed that the first character in the string is a nonzero digit.

Output

For each test, written in the input file, you should print a single integer into the output file: the sum of the numbers Anna has written on the piece of paper modulo $10^9 + 7$. If there are no numbers written on the paper, print 0 as an answer for the test.

Print answers for the tests in the same order as the tests are given in the input file.

Example

input.txt	output.txt
4 1 1 17 inc R0 3? 2 1 17 inc R0 1? 2 2 10 inc R1 add R0 R0 2? 2 2 1 add R1 R0 add R0 R1 1?3	160 32 135 745

Example explanation

In the first test the program increases the input number x by 1. Working range M is between 30 and 39 inclusive. It is easy to see that for any of these values of M, the input number x must be 16.

The second test is almost the same as the first one, but M is between 10 and 19. Since the output number y must be within the working range, only $M = 18, 19$ are suitable.

In the third test, the input number x is doubled. The increment command does not affect the output value. For any of the ten possible values of M, the input value $x = 5$ suits. Moreover, for any even M, $x = 5 + \frac{M}{2}$ also suits.