Problem A. Advanced 2048

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

Only lazy did not play the 2048 game last year. Soon, the secret of success had been revealed and interest to the game has faded. Nevertheless, the developers are now working on new modifications to the game. One of the prominent directions for the future development was the increased size of the game field and the increased final score.

However, in order to pick the optimal field size, it is required to emulate the gameplay many times on the fields of different sizes to estimate the number of moves needed to reach the goal under new conditions, and also to estimate the possible score values.

You are to write a program that will determine the final score given the description of a game session: field size, placement of the tiles and their values, the sequence of the player's moves and the description of all new tiles appearing on the field.

For those who are newbies to the popular game, the game rules are the following. 2048 is played on a $N \times N$ tile grid, with numbered tiles that all slide when a player moves them using the four arrow keys. After every turn, a new tile will randomly appear in an empty spot on the board with a value of either 2 or 4. Tiles slide as far as possible in the chosen direction until they are stopped by either another tile or the edge of the grid. If two tiles of the same number collide while moving, they will merge into a tile with the total value of the two tiles that collided. The resulting tile cannot merge with another tile again in the same move. The tile moves and mergers are resolved from the first tile to the last one (in terms of movement direction) consequently. For example, after moving to the right, the row "`2 2 _ 2 _`" turns into "`_ _ _ 2 4`", and the row "`2 2 2 2 2 2`" turns into "`_ _ _ 4 4 4`".

The player's score starts at zero, and is incremented whenever two tiles combine by the value of the new tile.

Input

The first line of input contains an integer N, the size of the field ($4 \le N \le 100$). The second line of input contains an integer K, the number of tiles already on the field ($0 \le K \le N^2$). In each of the next K lines, three integers are given to describe the tiles on the field: a number written in the tile (one of the powers of two from first (2) to tenth (1024)) and the tile coordinates on the field: row and column. Rows are numbered from one starting from the top, and columns are numbered from one starting from the left. It is guaranteed that the given tiles occupy K different cells of the grid.

After that, an integer L is given on a separate line: the number of player's moves in a game session ($0 \le L \le 10\,000$). The next L lines contain descriptions of the player's moves in their order, one description per line. A description of a move starts with one of the characters "`L`" (left), "`R`" (right), "`U`" (up) or "`D`" (down). Then follow three integers separated by single spaces. They define the new tile that appeared on the field after the player's move with the value of either 2 or 4 and the coordinates of the tile: row and column. It is guaranteed that the new tile is placed on an empty tile of the field after each player's move.

Output

Output a single integer: the final score in the game which is described in the input.

Examples

standard input	standard output
6 5 2 1 2 2 1 3 4 1 5 4 1 6 2 3 1 2 L 4 2 1 U 2 6 2	20
4 4 1024 1 1 1024 1 2 1024 2 1 1024 2 2 2 D 4 1 1 R 4 1 1	8192

Problem B. Bring Your Own Bombs

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

A sabotage platoon has completed its mission and planted some bombs on an enemy territory. Now they are about to report to their commanders how many enemy units will be destroyed. They have assigned this task to you.

The enemy territory is represented by a tile grid where the enemy platoons are located. Enemy units are standing in platoons that are represented by accurate rectangles, and they do not intersect each other. Each tile contains at most one enemy unit. Bombs have awesome firepower that can destroy all enemy units on the same tile line with the bomb: either horizontal or vertical, but not both. Unfortunately, all bombs are created manually by inexperienced engineers, and therefore, it is very hard to say how and if they explode or not. Only two values are known for certain for each bomb: p_1, the probability that the bomb will explode horizontally, and p_2, the probability that the bomb will explode vertically. It is implicitly known that with probability of $100\% - p_1 - p_2$ the bomb will not explode at all.

The sabotage platoon only plants the bombs outside the enemy platoons. Given the description of the enemy territory and planted bombs, you are to calculate the expected number of destroyed enemy units.

Input

The first line contains two integers N and M: the number of enemy platoons and bombs respectively ($1 \le N, M \le 10^5$).

The next N lines of input contain the descriptions of enemy platoons, one per line. Each platoon is described by four integers x_1, y_1, x_2, y_2: the coordinates of the tiles at the opposite corners of the rectangle ($x_1 \le x_2$, $y_1 \le y_2$).

Then the next M lines of input contain the description of bombs, one per line. Each bomb is described by integers x, y and numbers p_1, p_2: the coordinates of the tiles with bombs and the quality characteristics of the bombs given as an non-negative integer percentage respectively ($p_1 + p_2 \le 100$).

All coordinates do not exceed 10^9 by absolute value.

Output

Output must contain a single real number: the answer to the problem with absolute or relative error no more than 10^{-9}.

Examples

standard input	standard output
1 1 -1 -1 1 1 0 3 33 33	0.990000000000000
1 2 1 1 5 5 0 2 100 0 2 0 0 100	9.000000000000000
2 2 0 3 2 5 3 0 5 2 1 1 30 60 4 4 50 40	4.980000000000000

Problem C. CIA Datacenter

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

The CIA has decided to keep up with technology advancements and try to capture all the information passing in the internet. In order to store that information, they need to build a new datacenter. Since the information on the internet cables is captured with a very high speed, the datacenter needs to be built with a very high write speed capability. The information on this stage is only collected for future processing and will be deleted in a small period of time, therefore the reliability of storage is not in question, only the total speed of writing information to disks is important.

In order to fit in a tight budget set by the Congress last year, agency has decided to use cheap commercial-grade disks and controllers. The datacenter storage architecture is simple: disks are connected to controllers (there is no limit how many disks can be connected to a single controller) that are in turn connected to the central information intake. Every disk and controller can operate in parallel with others thus writing the data simultaneously. However, there are limits on the maximum writing speed for disks and on the maximum speed with which the controller can process incoming data.

The total write speed of the disks connected to one controller is the minimum between the sum of all disks' write speed limits and the speed limit of the controller.

Given the projected speed of information capture, please help CIA technical personnel to minimize the money spent on disks and controllers. The structure of market prices is such that you can assume it is crucial to minimize the number of disks first and then to minimize the number of controllers (without changing the number of disks).

Input

The first line of input contains integers A, B and C: the write speed limit of the disk, the speed limit of the controller for processing incoming data and the expected information capture speed ($1 \le A, B, C \le 10^9$).

Output

The first line of output should contain two integers X and Y: the number of disks and controllers needed to be able to save all incoming data according to the problem statement.

Examples

standard input	standard output
2 10 100	50 10
10 2 100	50 50
20 35 140	7 7
20 35 141	8 5

Problem D. Do it Right!

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

Given two distinct positive integers A and B, find out if it is possible to find a third positive integer C so that a triangle with the sides A, B and C is a right triangle. Remember that a triangle is called a right triangle if one of its angles equals to 90 degrees.

Input

The first line of input contains two positive integers A and B: the lengths of the two given sides ($1 \le A < B \le 100$).

Output

Output "`YES`" if it is possible to find such an integer C, or "`NO`" otherwise.

Examples

standard input	standard output
3 4	YES
1 2	NO

Note

- In the first example, we can take $C = 5$.
- In the second example, it is impossible to find an integer C with the required property.

Problem E. Equal Digits

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

For the given integer N and digit D, find the minimal integer $K \geq 2$ such that the representation of N in the positional numeral system with base K contains the maximum possible consecutive number of digits D at the end.

Input

The input contains two integers N and D ($0 \leq N \leq 10^{15}$, $0 \leq D \leq 9$).

Output

Output two integers: K, the answer to the problem, and R, the the number of consecutive digits D at the end of the representation of N in the positional numeral system with base K.

Examples

standard input	standard output
3 1	2 2
29 9	10 1
0 4	2 0
90 1	89 2

Problem F. Friends

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

Three friends Alex, Dmitry and Petr want to go abroad for a better life. They start at `09:00` in the morning in their home city A and want to be in the foreign city B as soon as possible.

Alex wants to start immediately and decided to use a car and a highway. He has to drive directly to the customs point C, then spends D minutes there to pass the customs, and then drives directly to the city B. He drives the car with a speed of V kilometers per hour.

Dmitry has bought an airplane ticket. His plane departs at time T from the city A and flies for some time F directly to the city B.

Petr felt adventurous, so he decided to take his neighbor's tractor and drive straight to the city B with a speed of W kilometers per hour. If Petr's path lies through the customs point C, he spends D minutes there to pass the customs too. Otherwise, he does not stop during the whole trip.

Now they wonder who will be the first in the city B and can cook a welcoming dinner for all of them. We assume that friends live on a plane, and the distance between points is the common Euclidean distance.

Input

The first line of input contains six integers X_A, Y_A, X_B, Y_B, X_C and Y_C, the coordinates in kilometers of cities A and B and the customs point C respectively ($0 \le X_A, Y_A, X_B, Y_B, X_C, Y_C \le 1000$). It is guaranteed that the customs point C is closer to A than the city B and closer to B than the city A.

The second line contains two integers D and V, the time to pass the customs in minutes and the speed of Alex's car in kilometers per hour respectively ($0 \le D \le 1000$, $40 \le V \le 100$).

The third line contains the departure time T and the duration F of Dmitry's flight, both in the format "`HH:MM`". It is guaranteed that the departure time is correct, later than `09:00` and earlier than `24:00`. The flight duration is greater than `00:00` and less than `24:00`.

The fourth line contains one integer W, the speed of Petr's tractor in kilometers per hour ($10 \le W \le 50$).

Output

Output the name of the first friend in the city B. It is guaranteed that the absolute difference between arrival times for each pair of friends will be at least one second.

Examples

standard input	standard output
0 0 10 0 5 5 60 100 10:00 00:05 10	Petr
0 0 100 0 50 50 0 100 09:01 02:10 50	Alex
0 0 1000 0 500 500 0 100 10:00 02:10 33	Dmitry

Problem G. Genealogy

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	256 mebibytes

As you may know, the dates in historical documents and descriptions can be given rather inaccurately. It often occurs that in a description of a historic event, the closest thing to a date is something like *"In the times of Isaac, the son of Abraham"*.

The professor Mofenko who is an admirer of alternative history wants to determine the maximum length of the chain of people from father to son that can be composed of people who were referred as sons in the given list (the length of a chain is the number of people in that chain). With the help of this information, he can invent a lot of amusing alternative histories that are irrefutable by documents.

For each person in the list, his family name, his first name and the first name of his father are given. The family names of a father and his son are always equal. By a happy coincidence, it turned out that all first names of all sons in the list are distinct, but names of fathers could be the same, even if they are different persons.

When Mofenko could not say for sure whether two people with the same first name and family name are the same person or two different persons, he can pick the alternative which allows to construct a longer chain, as long as no person turns out to be that person's own ancestor.

Input

The first line of input contains an integer N: the number of the descriptions of people ($1 \le N \le 10^5$).

The next N lines contain the description of a man in the following format: the family name, the first name, the words "`son of`" and the first name of his father.

All first names and family names start with a capital English letter, and all letters except the first one are lowercase English letters. Consecutive words are separated by a single space character. The length of each first name and each family name is not greater than 10.

Output

Output a single integer: the number of people in the longest chain.

Examples

standard input	standard output
7 Bible Isaak son of Abraham Bible Jacob son of Isaak Ivanov Ivan son of Petr Ivanov Protopop son of Vasiliy Ivanov Vasiliy son of Ivan Ivanov Petr son of Vasiliy Bible Judah son of Jacob	4
3 Ivanov Ivan son of Petr Ivanov Petr son of Vasiliy Petrov Vasiliy son of Ivan	2

Note

- In the first example, the longest chain is: Ivanov Petr, Ivanov Ivan, Ivanov Vasiliy, Ivanov Protopop.

Professor can assume that Vasiliy, Petr's father, is not present in the list. He is another Ivanov Vasiliy, not "`Ivanov Vasiliy son of Ivan`" from the list.

- In the second example, the longest chain is: Ivanov Petr, Ivanov Ivan. We could not add Ivanov Vasiliy to the chain because he is not in the list.

Problem H. Holes

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

In this problem, we investigate the life of rabbits.

Our rabbits live in the glades, some of which are connected with two-way trails. No trail makes a loop: each of them connects two different glades. There is at most one trail between any pair of glades.

Our rabbits can dig holes in the ground where they can hide in case of danger. The holes can be dug in the center of each glade. In order for the rabbit to hide in the hole, the rabbit just needs to get to the glade where the hole is located, and then it hides there immediately. Each hole can accommodate any number of rabbits.

The rabbit universe has two important features: there is a path between any two glades, and each glades except at most one is connected with no more than two other glades.

Rabbits want to make some holes so that in case of danger, they can minimize the time for all rabbits to hide. Help them find the right glades to dig the holes. The time is calculated as a number of trails a rabbit uses to get to the hole. All rabbits are traveling and hiding independently.

It is also assumed that there are so many rabbits that each glade has at least one of them.

Input

The first line of input contains integers N, M and K: the number of glades, the number of trails and the number of holes to be dug respectively ($1 \le K \le N \le 1000$).

Each of the next M lines describes one trail and contains two integers x_i and y_i: the numbers of glades that are connected by this trail ($1 \le x_i, y_i \le N$).

Output

Output a single integer: the minimum possible amount of time needed for all rabbits to hide in case all K holes are placed optimally.

Examples

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

Problem I. Interactive Primes Guessing

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

The jury has chosen the numbers X_0 and P ($1 \le X_0 < P \le N$, P is a prime number). You are given the number N: the upper limit for P.

Your task is to guess the number P.

In order to do that, your program will ask the questions in the form of A_i ($2 \le A_i \le N$, A_i is a prime number) where $i = 1, 2, \ldots$ is the index of the question.

After each question, the answer comes. In case the i-th question does not guess the number P, the jury calculates $X_i = (X_{i-1} \cdot A_i) \mod P$ and returns your the result of comparison of X_i and X_{i-1} ("`>`", "`<`" or "`=`").

Your program must guess the number P by asking not more than 42 questions.

Input

The first line of input contains a single integer N ($2 \le N \le 3 \cdot 10^5$). The answers for the questions of your program are given on separate lines.

If the i-th question is the number P, then the string "`OK`" is given as an answer. In other cases, the result of comparison between X_i and X_{i-1} is returned. If $X_i > X_{i-1}$ then the answer is "`>`", if $X_i < X_{i-1}$ then the answer is "`<`", otherwise the answer is "`=`".

After receiving the answer "`OK`", your program must stop sending questions.

Output

Your program must output a single question on a separate line. Each question must consist of a single prime number A_i ($2 \le A_i \le N$).

Examples

standard input	standard output
100 > > > > > OK	29 43 23 17 29 17 73

Note

Among all possible pairs (X_0, P) where $P \le 100$, only the pair $(41, 73)$ satisfies the following equations for the questions given in the first example test: $X_1 < X_0$, $X_2 > X_1$, $X_3 > X_2$, $X_4 > X_3$, $X_5 > X_4$, $X_6 > X_5$ ($X_0 = 41$, $X_1 = 21$, $X_2 = 27$, $X_3 = 37$, $X_4 = 45$, $X_5 = 64$, $X_6 = 66$).

The pipe from your program to the interactor program and the pipe back have limited size. Your program must read from the standard input to avoid deadlock. Deadlock condition is reported as "`Time Limit Exceeded`".

To flush the standard output stream, use the following statements:

In C, use `fflush(stdout);`

In C++, use `cout.flush();`

In Java, use `System.out.flush();`

In Python, use `sys.stdout.flush()`.

If your program receives an EOF (end-of-file) condition on the standard input, it MUST exit immediately with exit code 0. Failure to comply with this requirement may result in "`Time Limit Exceeded`" error.

Problem J. JPEG is Awesome!

Input file:	*standard input*
Output file:	*standard output*
Time limit:	5 seconds
Memory limit:	512 mebibytes

Anna and her husband Boris has finally got out to Prague. And now she is going on excursions with her new digital SLR camera every day. She is not very good at it and all of its functions, therefore she is only able to make photos and delete them. Since all excursions are made early in the morning and her husband Boris is very fond of Czech beer, she is going on excursions alone. She asks her husband to set needed photo quality an evening before excursions (the memory card is not very big, and she frequently needs to delete photos).

At the end of their trip, she found out that she had deleted many good photos during the trip, others were just plain bad, and some awesome photos were made in a bad quality since she tried to fit everything on a single memory card. If only she could have known everything in advance, her photo set would have been so much better!

You are about to research this topic together with Boris. It is known that their trip took K days, the memory card in their camera contains L units of memory, and every photo in RAW mode takes D memory.

For each day, it is known how much photos Anna made and the awesomeness factor $Q_{i,j}$ of each photo. If Boris sets photo quality for the i-th day to α_i ($0 \le \alpha_i \le 1$), then each photo on that day takes $D \cdot \alpha_i$ memory (due to modern characteristics of memory used in their camera, this number can be non-integer), however the awesomeness factor of each photo is also multiplied by α_i. The α_i coefficient is the same for all photos taken on the same day but could be different for different days. It is also allowed to delete any photos at any time.

You are to find out the maximum possible awesomeness of the final set of photos. The awesomeness of the set of photos is the sum of awesomeness factors of each non-deleted photo.

Input

The first line of input contains three integers K, L and D ($1 \le K \le 10^6$, $1 \le L, D \le 10^9$).

The next K lines contain the descriptions of days. The first integer on a line is N_I, the number of photos taken on i-th day ($1 \le N_i \le 10^6$). The next N_i integers $Q_{i,j}$ are the awesomeness factors of the photos made on that day in the order they were taken ($1 \le Q_{i,j} \le 10^9$). The sum of all N_i is less than or equal to one million.

Output

The first line of output should contain the answer to the problem. If the answer is integer, output that integer, otherwise output the answer in the form "$r + p/q$" (without quotes and with single spaces before and after the "+" sign) where r, p and q are non-negative integers such that $0 < p < q$ and p and q are relatively prime.

It is guaranteed that the answer is always a rational number.

Examples

standard input	standard output
2 8 3 2 9 6 2 8 7	21 + 1/2
1 7 2 7 1 10 1 100 1 10 1	120

Problem K. Killing The Time

Input file:	*standard input*
Output file:	*standard output*
Time limit:	10 seconds
Memory limit:	512 mebibytes

Roma has downloaded a modern strategic game with RPG elements on his smartphone. In the game, the player is using a turn-based mechanism to command three characters and fight the hordes of enemies. Since the game is a shareware and Roma is a serious guy that does not spend money on games, this game is available to him in a very limited mode.

The number of moves is severely limited, enemies do not move, enemies and characters do not die, but the score at the end of the game is still displayed. As Roma has figured out from his attempts to play this game, the score consists of the sum of damage dealt by characters to the enemies and life healed to the characters minus the sum of damage dealt by the enemies to the characters and life healed by the characters to the enemies. Roma is about to set the absolute record for each of the levels. And you are about to help him!

The game field is a rectangular tiled grid of size $N \times M$. Some tiles are occupied by enemies, some other by impassable obstacles, yet some other by three Roma's characters, and all other tiles are free. The game process is working in the following way. Roma moves first, then moves the almighty artificial intelligence. In his turn, Roma chooses the order in which the characters will move, and then makes those moves sequentially, character by character. During his turn, Roma can move the character and then apply one of its abilities. The characters move by steps (one tile by each step) either horizontally or vertically so that the total number of moves does not exceed that character's stamina. Characters can not pass through the impassable obstacles, enemies or other characters. Roma can also decide not to move the character at all, or not to apply any of its abilities, or not to do anything by that character at all. After Roma's turn, the enemies take action, and artificial intelligence tries to minimize the player's score. Then a new turn begins or the game ends.

Each of the characters' abilities is one of the following:

- "`TELEPORT` L": the character can teleport to any free tile of the game field with Manhattan distance not more than L from the current tile.
- "`SWAP` L: the character can exchange places with any of the other characters with Manhattan distance not more than L from the current tile.
- "`SINGLE ATTACK` r R d": the character deals d base damage to any one selected tile with Manhattan distance not less than r and not greater than R from his current position.
- "`SPLASH ATTACK` r R d": the character deals d base damage to all tiles with Manhattan distance not less than r and not greater than R from his current position.
- "`SINGLE HEAL` r R d": the character heals d life to any one selected tile with Manhattan distance not less than r and not greater than R from his current position.
- "`SPLASH HEAL` r R d": the character heals d life to all tiles with Manhattan distance not less than r and not greater than R from his current position.

Each type of enemy is one of the following:

- "`SINGLE` r R d": the enemy deals d base damage to any one selected tile with Manhattan distance not less than r and not greater than R from its current position.
- "`SPLASH` r R d": the enemy deals d base damage to all tiles with the Manhattan distance not less than r and not greater than R from its current position.

Each enemy has three multipliers for the dealt damage: it deals $d \cdot M_1$ damage to the first character, $d \cdot M_2$ to the second character and $d \cdot M_3$ to the third character. Additionally, each enemy has three multipliers for the damage received: damage from the first character is multiplied by S_1, from the second by S_2, and from the third by S_3.

Since the game is played on hardcore level, the characters can deal damage to each other (base damage value) and also can heal enemies, but the enemies do not deal damage to each other.

The Manhattan distance between two positions is the sum of the absolute difference between row numbers of these two positions and of the absolute difference between column numbers of these two positions.

You are to calculate the maximum score that Roma can earn in the given game.

Input

The first line of input contains integers N, M, D and K: the dimensions of the game field, the length of the game and the number of types of enemies on the level K ($1 \le N, M, D \le 8$, $0 \le K \le 26$).

On the next N lines, the game field is given. Each of them contains exactly M characters: "`.`" denotes the field is empty; "`#`" denotes the field is not passable; "`1`", "`2`" or "`3`" denotes the positions of the characters; "`A`.."α" (where α is the K-th letter of English alphabet) denotes the type of the enemy on the field. It is guaranteed that there is exactly one of each character "`1`", "`2`" and "`3`" on the field.

On the next K lines, the descriptions of the enemy types are given. First goes the type of damage, and then follow nine numbers r, R, d, M_1, M_2, M_3, S_1, S_2, S_3 ($0 \le r \le R \le 20$, $1 \le d, M_1, M_2, M_3, S_1, S_2, S_3 \le 1000$). The first description is for type "`A`", the second one for type "`B`" and so on.

Then the descriptions of three characters are given. On the first line of each of these descriptions, two numbers are given: V, the stamina of the character and A, the number of his abilities ($0 \le V \le 5$, $0 \le A \le 100$). Then A lines describe the abilities in the format "`name of ability` L" for movement abilities ($0 \le L \le 5$) and "`name of ability` r R d" for the other abilities ($0 \le r \le R \le 20$, $1 \le d \le 1000$).

Output

Output one integer: the maximum score that Roma can earn in the given game.

Examples

standard input	standard output
4 4 1 1 .... .12. .... AA3A SINGLE 0 1 10 1 1 100 10 10 1 1 2 SPLASH HEAL 0 3 4 TELEPORT 1 1 2 SINGLE ATTACK 0 10 10 SWAP 2 1 1 SPLASH ATTACK 1 1 1000	1984
1 6 8 2 123#BA SINGLE 5 5 10 1000 100 1 1 1 1 SINGLE 2 2 5 300 1 100 1 1 1 5 1 SINGLE ATTACK 5 5 1 5 1 SPLASH HEAL 1 1 1 5 2 SPLASH ATTACK 0 1 100 SWAP 2	-5574

Problem L. Linear Systems

Input file:	*standard input*
Output file:	*standard output*
Time limit:	8 seconds
Memory limit:	512 mebibytes

After dealing with sociology problem last year, you now have some troubles with linear algebra.

An old professor likes to give his students huge systems of linear equations by prime modulo P to solve as a homework. He can dictate them for hours, but he found out that students still solve them quite fast and giving even bigger systems of linear equations is not an option due to limited duration of the seminar. He invented one feature though: he has his favorite number L, and if he has already dictated L coefficients of the equation, he can stop and instruct students to copy the other $(N - L)$ coefficients from the the beginning of one of the previous equations, and then dictates the right part of the equation.

OK, you'll have to deal with the task of automating the solving of a system of linear equations given in that form.

Input

On the first line of input, integers N, M, L and P are given, where N is the number of unknown variables, M is the number of equations, L is the favorite number of the professor and P is the number serving as a modulo ($1 \le N, M \le 4000$, $1 \le L < N$, $2 \le P \le 10^9$, P is prime).

The next M lines contain the equation descriptions. The first integer denotes the type of the equation: 1 if the equation is defined by professor in complete form, or 2 if he referenced the previous equation in its definition. In the former case, the line then contains $(N + 1)$ integers: the coefficients for the unknown variables and the right side of equation. In the latter case, the line then contains $(L + 2)$ integers: the L coefficients for the unknown variables, the index n_i of the equation referenced by the professor ($1 \le n_i < i$) and the right side of equation. All coefficients are non-negative and less than P.

The total amount of coefficients for the unknown variables dictated by the professor is not greater than 10 000.

Output

The first line of output must contain "`No solution`" if there is no solution for the given system of linear equations.

If a solution exists, then the first line of output must look like "`There are` P `^` k `solutions`", where k is the dimension of the solution space. The second line in this case must contain N integers x_i: the solution itself ($0 \le x_i < P$). If there are multiple solutions with these criteria, output the one with the minimal x_1. If there are still multiple solutions, output the one with the minimal x_2, and so on.

Examples

standard input	standard output
4 4 2 13 1 1 0 0 0 1 1 0 1 0 0 2 1 0 0 1 0 3 1 0 0 0 1 4	There are 13 ^ 0 solutions 1 2 3 4
4 3 1 19 1 1 0 0 0 1 2 1 1 2 2 1 2 3	There are 19 ^ 1 solutions 1 1 1 0
2 2 1 11 1 1 1 1 2 1 1 2	No solution