Problem A. Advertising Strategy

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

Recently you have been hired by a company that wished to remain unnamed to promote videos on its Itube channel. There are n users subscribed to this channel. The rules of video going viral are complicated: on the one hand, every day, the number of subscribers who have seen the video can double; on the other hand, at most half of all subscribers who haven't seen the video yet will watch it.

To promote a new video you have been assigned a budget of k burles. At the beginning of each day you can spend x burles to make x persons watch the video. Obviously, you are going to pick subscribers who haven't seen it yet. Each day you can choose different value of x, but the total number of burles spent can not exceed k. Formally, on the i-th day the following happens:

1. Denote a_i the number of subscribers who have seen the video so far. At the beginning of the day you spend x_i burles, and now the number of subscribers who have seen the video is $b_i = a_i + x_i$.
2. During the day this number increases according to the rules described above, so $a_{i+1} = b_i + \min(b_i, \lfloor \frac{n-b_i}{2} \rfloor)$.

You have already noticed that according to the rules above you will have to pay at least for the first person to see the video and for the last person. Now you wonder what is the minimum number of days required to make all n subscribers to see the video, assuming you can not spend more than k burles?

Input

The only line of the input contains two integers n and k ($1 \le n \le 10^{18}$, $2 \le k \le 100\,000$) — the number of subscribers to the channel and the amount of money available (in burles).

Output

Print one integer — the minimum number of days required to make all subscribers see the video.

Examples

standard input	standard output
`7 4`	`3`
`10 2`	`6`

Note

In the first sample, one of the optimal solutions is:

1. Pay 1 burle at the beginning of the first day. At the end of the day the number of subscribers who have seen the video is 2.
2. Pay 1 more burle to make $b_3 = 3$, there will be 5 subscribers affected at the end of the day.
3. Pay 2 burles to make everyone see the video.

In the second sample, you pay one burle at the beginning of the first day, then the number of subscribers who have seen the video changes as: 2, 4, 7, 8, 9, and then you pay to make the last subsribers see the video.

Problem B. Byteland Trip

Input file:	`standard input`
Output file:	`standard output`
Time limit:	6 seconds
Memory limit:	512 megabytes

The Byteland Empire consists of n planets arranged in a row. Planets are numbered from 1 to n. You are planning your journey to Byteland, but lack of convenient means of transportation annoys you.

Nowadays the only transport available in the Byteland Empire is teleportation. Unfortunately, the teleportation system in Byteland is quite tricky to use. Each planet has a single teleportation center, which can be either forward-bound or backward-bound. The difference between forward-bound and backward-bound teleportation centers is that from a forward-bound teleportation center you can travel to any planet with greater index, while from a backward-bound teleportation center you can travel to any planet with smaller index. Formally, if planet i has a forward-bound teleportation center, you can use it to travel to any planet $j > i$, while if planet i has a backward-bound teleportation center, you can travel to any planet $j < i$.

Your dream is to visit every planet **exactly once**. Your trip can start on any planet. For each planet i of the Byteland Empire you would like to know the number of valid trips that start at any other planet, visit every planet exactly once, and end on the planet i. As it often happens with you, the values you want to compute may be too large, so you are only interested in their remainders modulo $10^9 + 7$.

Input

The input consists of a single string s of length n ($1 \le n \le 5000$) — the description of teleportation centers on all planets. Each character of this string is either '<' or '>'. The i-th character is equal to '<' if the i-th planet has a backward-bound teleportation center, and is equal to '>' if it has a forward-bound teleportation center.

Output

Print n integers — for each planet i the number of valid trips that end on planet i modulo $10^9 + 7$.

Examples

standard input	standard output
`>><`	`1 2 1`
`><>>><`	`1 2 2 4 8 1`

Note

In the first sample, the only valid trip that ends on planet 1 is $2 \to 3 \to 1$, all valid trips that end on planet 2 are $3 \to 1 \to 2$ and $1 \to 3 \to 2$, and the only valid trip that ends on planet 3 is $1 \to 2 \to 3$.

Problem C. Carpet

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

All cinema stars of Treeland are gathering in its capital to take part in the cinema academy award ceremony. This time organizers decided to prepare something very new for the welcoming part. Their idea is to replace old-fashioned red carpet with a one having a map of Treeland depicted on it.

Treeland consists of n cities connected with $n - 1$ bidirectional roads in such a way that it is possible to get from any city to any other city. Let us represent the carpet as a rectangular grid of $1\,000\,000 \times 20$ unit square cells. Cities should be painted in centers of some cells in such a way that no two cities occupy the same cell. After that roads are painted as segments connecting corresponding cell centers.

To make the carpet beautiful, it was decided to place cities in such a way that no two segments corresponding to roads intersect. This means that no two segments have common points different from their common endpoints.

Some clever mathematician already proved that a solution exists, but unfortunately his proof doesn't provide a way to construct the answer, so this task was assigned to you.

Input

The first line of the input contains a single integer n ($1 \le n \le 100\,000$) — the number of cities in Treeland.

Each of the next $n - 1$ lines contains a pair of integers u_i and v_i ($1 \le u_i, v_i \le n$, $u_i \ne v_i$) — indices of a road that connects cities u_i and v_i. It is guaranteed that it is possible to get from any city to any other city.

Output

Print n lines. The i-th of these lines should contain two integers x_i and y_i ($1 \le x_i \le 1\,000\,000$, $1 \le y_i \le 20$) — the coordinates of the cell to place the i-th city.

Examples

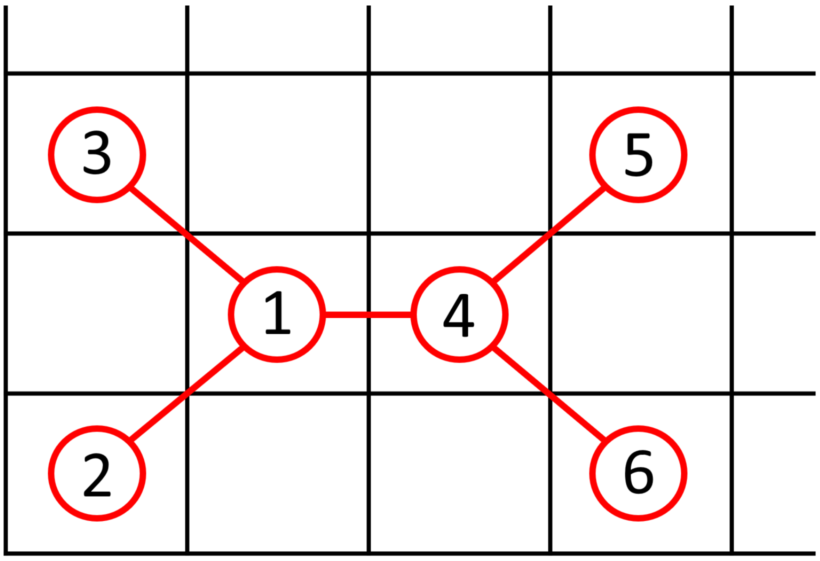
standard input	standard output

6
1 2
1 3
1 4
4 5
4 6
	2 2
	1 1
	1 3
	3 2
	4 3
	4 1

7
1 2
1 3
2 4
2 5
3 6
3 7
	1 1
	1 2
	2 2
	1 3
	2 3
	3 3
	4 3

Note

The picture below illustrates a possible solution for the first sample.


3
5
1
4
2
6

Problem D. Decoding of Varints

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

Varint is a type used to serializing integers using one or more bytes. The key idea is to have smaller values being encoded with a smaller number of bytes.

First, we would like to encode some unsigned integer x. Consider its binary representation $x = a_0a_1a_2 \dots a_{k-1}$, where a_i-th stands for the i-th significant bit, i.e. $x = a_0 \cdot 2^0 + a_1 \cdot 2^1 + \dots + a_{k-1} \cdot 2^{k-1}$, while $k - 1$ stands for the index of the most significant bit set to 1 or $k = 1$ if $x = 0$.

To encode x we will use $m = \lceil \frac{k}{7} \rceil$ bytes $b_0, b_1, \dots, b_{m-1}$. That means one byte for integers from 0 to 127, two bytes for integers from 128 to $2^{14} - 1 = 16383$ and so on, up to ten bytes for $2^{64} - 1$. For bytes $b_0, b_1, \dots, b_{m-2}$ the most significant bit is set to 1, while for byte b_{m-1} it is set to 0. Then, for each i from 0 to $k - 1$, $i \bmod 7$ bit of byte $b_{\lfloor \frac{i}{7} \rfloor}$ is set to a_i. Thus,

$$x = (b_0 - 128) \cdot 2^0 + (b_1 - 128) \cdot 2^7 + (b_2 - 128) \cdot 2^{14} + \dots + (b_{m-2} - 128) \cdot 2^{7 \cdot (m-2)} + b_{m-1} \cdot 2^{7 \cdot (m-1)}$$

In the formula above we subtract 128 from $b_0, b_1, \dots, b_{m-2}$ because their most significant bit was set to 1.

For example, integer 7 will be represented as a single byte $b_0 = 7$, while integer 260 is represented as two bytes $b_0 = 132$ and $b_1 = 2$.

To represent signed integers we introduce *ZigZag* encoding. As we want integers of small magnitude to have short representation we map signed integers to unsigned integers as follows. Integer 0 is mapped to 0, -1 to 1, 1 to 2, -2 to 3, 2 to 4, -3 to 5, 3 to 6 and so on, hence the name of the encoding. Formally, if $x \geq 0$, it is mapped to $2x$, while if $x < 0$, it is mapped to $-2x - 1$.

For example, integer 75 is mapped to 150 and is encoded as $b_0 = 150$, $b_1 = 1$, while -75 will be mapped to 149 and will be encoded as $b_0 = 149$, $b_1 = 1$. In this problem we only consider such encoding for integers from -2^{63} to $2^{63} - 1$ inclusive.

You are given a sequence of bytes that corresponds to a sequence of signed integers encoded as varints. Your goal is to decode and print the original sequence.

Input

The first line of the input contains one integer n ($1 \leq n \leq 10\,000$) — the length of the encoded sequence. The next line contains n integers from 0 to 255. You may assume that the input is correct, i.e. there exists a sequence of integers from -2^{63} to $2^{63} - 1$ that is encoded as a sequence of bytes given in the input.

Output

Print the decoded sequence of integers.

Example

standard input	standard output
5 0 194 31 195 31	0 2017 -2018

Problem E. Empire History

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	512 megabytes

Long time ago, there were two great Kingdoms Someland and Otherland. Each of them consisted of at least two cities and some roads connecting them. It was possible to get from any city of any kingdom to any other city of this kingdom using only these roads. However, no two cities of different kingdoms were connected by a road.

As an old legend says, when both kingdoms were conquered by the Emperor of New Roads, he decided to unite them by doing what he was very well known for — by building new roads. He picked some non-empty subset of cities of Someland $u_1, u_2, \ldots, u_k$ and some non-empty subset of cities of Otherland $v_1, v_2, \ldots, v_l$. Then, he ordered to build a new road between each pair of cities u_x and v_y ($1 \le x \le k$, $1 \le y \le l$), thus adding $k \cdot l$ new roads.

Hundreds of years have passed. Since then, no new cities and no new roads were built, and no cities or roads were destroyed. However, no one actually remembers which cities were part of Someland and which cities were part of Otherland. This year, the empire plans to celebrate millennium, and you were hired to restore the historical truth and determine the possible division of cities between Someland and Otherland.

Input

The first line of the input contains two integers n and m ($4 \le n \le 200\,000$, $n - 1 \le m \le 200\,000$) — the number of cities and the number of roads in the current map of the empire.

Each of the next m lines contains two integers u_i and v_i ($1 \le u_i, v_i \le n$, $u_i \ne v_i$) — the indices of cities connected by the i-th road. No road connects city with itself, and no two roads connect the same pair of cities.

Output

The first line of the output should contain two integers n_1 and k ($2 \le n_1 \le n - 2$, $1 \le k \le n_1$) — the number of cities in Someland and the number of cities of Someland selected by the Emperor of New Roads.

The second line should contain n_1 distinct integers — indices of the cities of the empire that one day were part of Someland. First k of them should denote the subset selected for building new roads.

The third line should contain two integers n_2 and l ($n_2 = n - n_1$, $1 \le l \le n_2$) — the number of cities in Otherland and the number of cities of Otherland selected to build the new roads.

The fourth line should list cities of Otherland in the same format as the cities of Someland.

In your solution, both Someland and Otherland should be connected. As no one really cares how these two kingdoms looked like, if there are several possible solutions, print any of them. It is guaranteed that at least one possible solution exists.

Examples

standard input	standard output

6 12
1 2
1 3
2 3
1 4
1 5
1 6
2 4
2 5
2 6
3 4
3 5
3 6
	4 4
	3 4 5 6
	2 2
	1 2

6 10
1 2
1 3
1 4
1 5
6 2
6 3
6 4
6 5
2 3
4 5
	2 2
	4 5
	4 2
	1 6 2 3

Problem F. Fake or Leak?

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

In Someland a great programming competition was held. The scoreboard is now frozen, and everyone is waiting for the final results announcement. At the same time, a group of hackers published a screenshot with the part of the final unfrozen scoreboard. However, looking on the published part of the scoreboard you have started to doubt it is not a fake.

You are given the frozen scoreboard and what is claimed to be a part of the final scoreboard. You have to find out whether it is possible that you are given a real part of the final scoreboard.

The rules of this competition are as follows:

1. There are m teams participating and n problems available to solve.
2. The contest lasts for five hours. The scoreboard is frozen after four hours.
3. Each time a team submits a solution for some problem, this submission is either accepted or rejected. Each team is allowed to submit each problem multiple times. A team never submits a solution for a problem for which it already made a successful submission.
4. Teams are ranked according to the most problems solved.
5. Teams that have solved the same number of problems are ranked by total time (the less the better) and, if need be, by the earliest time of the last accepted submission.
6. The total time is the sum of the time consumed for each problem solved. The time consumed for a solved problem is the time elapsed from the beginning of the contest to the submittal of the first accepted run plus 20 penalty minutes for every previously rejected run for that problem. There is no time consumed for a problem that is not solved.
7. If two or more teams are still tied by the total number of problems solved, the total time consumed and the time of the last accepted submission, they are ranked by their names in lexicographical order.

You are given a frozen scoreboard and what is claimed to be k **consecutive** lines of the final scoreboard. Your goal is to check whether it is possible that the claim is true.

Input

The first line of the input contains three integers n, m and k ($1 \le n \le 26$, $1 \le k \le m \le 1000$) — the size of the problem set, the number of teams participating, and the size of the leaked part of the final scoreboard.

In the next m lines, the results of the teams in the frozen scoreboard are listed from top to bottom, i.e. from the best to the worst result. Then follow k lines describing the allegedly leaked continuous part of the final scoreboard.

Each line that describes the results of a team obeys the following format. The line starts with a team name. The name is a non-empty string of at most 10 lower-case English letters. Then follow n triples c, a, t ($0 \le a \le 100$, $0 \le t \le 299$), describing the results of this team for each problem. The character c is one of '`+`', '`-`', or '`.`', meaning the team has already solved this problem, the team has made some unsuccessful attempts and haven't solved this problem yet, or the team has never attempted this problem respectively. Integer a stands for the number of submissions, and integer t stands for the minute when the last submission was made.

It is guaranteed that both scoreboards are self-consistent and non-contradictory. In particular, that means:

1. All names in the frozen scoreboard are distinct. All names in the allegedly leaked part of the final scoreboard are distinct. Any name that is present in the allegedly leaked part of the scoreboard is present in the frozen scoreboard.
2. If some c equals '`+`' or '`-`', the corresponding value of a is positive.
3. If some c equals '`.`', the corresponding values of a and t are 0.
4. If some a equals 0, corresponding value of t is 0 and c is '`.`'.
5. Both the frozen scoreboard and the allegedly leaked part have teams ranked according to the rules described above.
6. All values of t in the frozen scoreboard do not exceed 239.
7. Any team's results present in the allegedly leaked scoreboard is consistent with its results in the frozen scoreboard. Let c_1, a_1 and t_1 be some team's result for some problem in the frozen scoreboard, and c_2, a_2 and t_2 be results of the same team for the same problem in the allegedly leaked part of the scoreboard. Then,
   - $a_1 \le a_2$;
   - if $a_1 = a_2$, then $c_1 = c_2$ and $t_1 = t_2$;
   - if $c_1 =$ '`+`', then $c_2 =$'`+`' and $a_1 = a_2$;
   - if $a_1 < a_2$, then c_1 is either '`-`' or '`.`', c_2 is either '`+`' or '`-`', and $t_2 \ge 240$.

Output

If it is possible that the last k lines of the input are k consecutive lines of the final scoreboard, print "`Leaked`" (without quotes). Otherwise, print "`Fake`" (without quotes).

Examples

standard input	standard output

`3 3 2`
`crabs + 1 1 + 1 2 + 1 3`
`lions . 0 0 - 5 239 . 0 0`
`wombats . 0 0 . 0 0 . 0 0`
`wombats + 1 241 + 3 299 - 22 299`
`lions + 1 241 + 6 240 - 3 299`
	`Leaked`

`3 4 2`
`crabs + 1 1 + 1 2 + 1 3`
`lions . 0 0 + 5 239 . 0 0`
`wolves . 0 0 . 0 0 . 0 0`
`wombats . 0 0 . 0 0 . 0 0`
`crabs + 1 1 + 1 2 + 1 3`
`wombats . 0 0 + 2 299 . 0 0`
	`Fake`

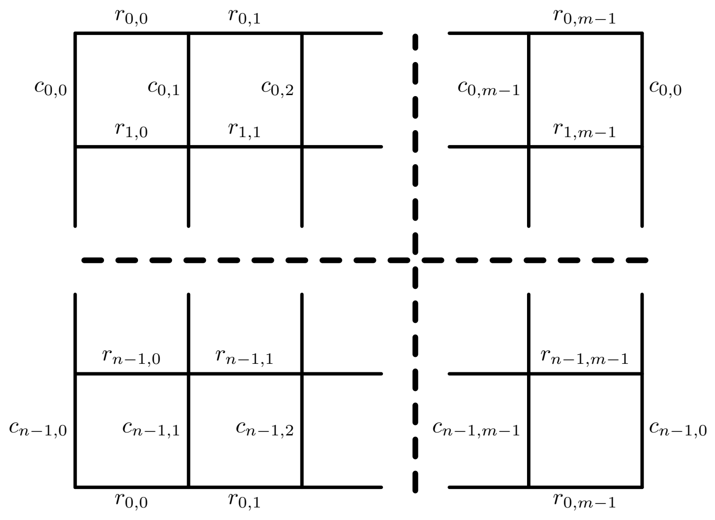
Problem G. God of Winds

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

During his famous journey, Odysseus visited Aeoli island which was ruled by Aeolus, the keeper of the winds. Odysseus spent a month there, resting and telling Aelous intriguing twists of his story. One evening Odysseus asked Aeoli if he was able to set up an arbitrary wind circulation over a toroidal map of a certain size.

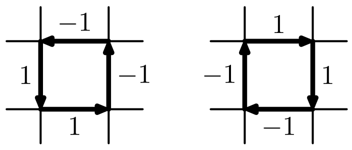
Formally, consider an $n \times m$ rectangular grid. The grid is toroidal which means that the topmost horizontal grid edges are identified with the bottommost grid edges, as are the leftmost vertical grid edges and the rightmost vertical edges. Hence, there are $2nm$ distinct grid edges in total. We denote the j-th from the left cell of the i-th from the top row as cell (i, j) (both indices are 0-based).

Define *wind* as an arbitrary assignment of integers to each of the grid edges. These numbers are denoted as given on the picture below, i.e. $r_{i,j}$ is an integer assigned to the top edge of the grid cell (i, j), and $c_{i,j}$ is an integer assigned to the left edge of the grid cell (i, j). Note that there are no dedicated values $r_{n,j}$ and $c_{i,m}$ since the grid is toroidal, but for the sake of simplicity we will define $r_{n,j}$ equal to $r_{0,j}$, and $c_{i,m}$ equal to $c_{i,0}$.



A positive value of $r_{i,j}$ corresponds to the wind flowing from left to right, a negative value of $r_{i,j}$ corresponds to the wind flowing from right to left. The absolute value of $r_{i,j}$ defines the wind flow. Similarly, positive $c_{i,j}$ corresponds to the wind flowing from top to bottom, and negative $c_{i,j}$ corresponds to the wind flowing from bottom to top.

Aeolus is capable of creating unit *cyclons* and *anticyclons*. Adding a unit *cyclon* around a grid cell (i, j) increases $r_{i+1,j}$ and $c_{i,j}$ by 1, and decreases $r_{i,j}$ and $c_{i,j+1}$ by 1; informally, this means that we add a unit of wind flow around the cell (i, j) in counter-clockwise direction. Adding a unit *anticyclone* results in completely the opposite: $r_{i+1,j}$ and $c_{i,j}$ are decreased by 1, $r_{i,j}$ and $c_{i,j+1}$ are increased by 1. As one can notice, adding one cyclon and one anticyclon to the same cell does not change anything.



Cyclone and anticyclone

Odysseus challenged Aeolus claiming that he would not be able to obtain a given wind by creating several cyclons and anticyclones starting from zero wind (i.e. wind with all edge values equal to zero). Write a program that determines if this is possible or not.

Input

The first line contains two integers n and m ($2 \le n, m \le 500$) — the dimensions of the grid size.

The following n lines define the desired wind. In the i-th of the following lines there are $2m$ integers $r_{i,0}, c_{i,0}, r_{i,1}, c_{i,1}, \ldots, r_{i,m-1}, c_{i,m-1}$ ($-10^7 \le r_{i,j}, c_{i,j} \le 10^7$).

Output

If it is possible to obtain the given wind by creating several cyclones and anticyclones, print “`Yes`” (without quotes). Otherwise, print “`No`” (without quotes).

Examples

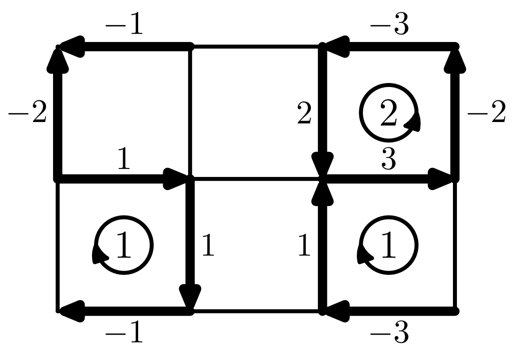
standard input	standard output

2 3
-1 -2 0 0 -3 2
1 0 0 1 3 -1
	Yes

2 2
0 0 0 1
1 0 -1 -1
	No

Note

One of the possible ways to obtain the wind from the first sample is to take two cyclones around the cell $(0, 2)$, one anticyclone around the cell $(1, 2)$, and one anticyclone around the cell $(1, 0)$.



Problem H. Hilarious Cooking

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

All laws of physics appearing in this problem statement are fictitious. Any resemblance to the real world is purely coincidental.

To discuss problem proposals and select a proper problemset for an upcoming competition jury team has gathered at a magnificent palace in Moscow suburbs (typically called *dacha*). Chief judge is now struggling to prepare a roast beef barbecue using a recipe he just found in the Internet.

According to recipe a roast beef should be cooked for exactly n seconds. The total amount of heat received by beef should be equal to T. Formally, if we denote the temperature inside the barbecue during the i-th second as t_i, the sum $t_1 + t_2 + \ldots + t_n$ should be equal to T. Moreover, there are m additional instructions, the i-th of them states that in order to achieve the perfect result the temperature inside the barbecue during the a_i-th second should be equal to b_i, i.e. t_{a_i} should be b_i.

However, the task is complicated by the fact that the barbecue that chief judge is using this time has some restrictions. First of all, the barbecue only allows t_i to take non-negative integer values. Second, t_i can't change too fast, i.e. the condition $|t_i - t_{i+1}| \le 1$ should be satisfied for all i from 1 to $n - 1$.

There are no other restrictions, in particular it is allowed to start and finish with any temperature, i.e. the values t_1 and t_n can be arbitrary non-negative integers, unless the opposite is directly specified in the recipe. Your goal is to help chief judge and determine whether his task is even possible, or he should look for another recipe.

Input

The first line of the input contains three integers T, n and m ($1 \le T \le 10^{18}$, $1 \le n \le 2 \cdot 10^9$, $1 \le m \le 100\,000$) — the total amount of heat that beef should receive, the exact number of seconds it should be cooked for, and the number of instructions in the recipe.

The next m lines contain the instructions in chronological order. The i-th instruction is defined by two integers a_i and b_i ($1 \le a_i \le n$, $0 \le b_i \le 10^9$) stating that the temperature inside the barbecue should be equal to b_i during the second a_i. It is guaranteed that $a_1 < a_2 < \ldots < a_m$.

Output

If there exists a sequence of non-negative $t_1, t_2, \ldots, t_n$ such that $\sum_{i=1}^{n} t_i = T$, $|t_i - t_{i+1}| \le 1$ for all $1 \le i \le n - 1$, and $t_{a_i} = b_i$ for all i from 1 to m, print "`Yes`" (without quotes) in the only line of the output. Otherwise, print "`No`" (without quotes).

Examples

standard input	standard output

`3 3 1`
`2 1`
	`Yes`

`10 3 1`
`1 1`
	`No`

`13 5 2`
`2 2`
`4 2`
	`Yes`

Note

In the first sample, a possible solution is $t_1 = t_2 = t_3 = 1$. In the third sample, a possible solution is $t_1 = 3$, $t_2 = 2$, $t_3 = 3$, $t_4 = 2$, $t_5 = 3$.

Problem I. Infinite Gift

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

Vasya the Infinite has received a very dull present — the integer lattice $\mathbb{Z}^k$. Formally, the lattice consists of all k-dimensional vectors with integer coordinates.

Disappointed by the present, Vasya decided to make the lattice prettier. He performed the following procedure n times: choose a vector $v_i \in \mathbb{Z}^k$, and then for every element $a \in \mathbb{Z}^k$ connect a and $a + v_i$ with a piece of string. Note that Vasya may have chosen the same vector more than once, but he has never chosen zero vector $v_i = (0, \ldots, 0)$.

After this tedious procedure Vasya discovered that the lattice is now connected, that is, for any pair of elements $a, b \in \mathbb{Z}^k$ one can get from a to b by travelling along pieces of string several times. It is allowed to traverse each piece of string in both ways.

To make the lattice even prettier, Vasya now wants to color each vector of his lattice either white or black so that any two vectors directly connected with a piece of string have different colors. Vasya's patience is not infinite, so you need to help him quick!

Input

The first line contains two integers n and k ($1 \le k \le n \le 1500$) — the number of times Vasya has chosen a vector and the dimension of the lattice.

Next n lines describe vectors $v_1, v_2, \ldots, v_n$. The i-th of these lines contains k integers $v_{i,1}, \ldots, v_{i,k}$ ($-1000 \le v_{i,j} \le 1000$). It is guaranteed that all vectors v_i are non-zero, and that the lattice is connected.

Output

Print "`Yes`" if it is possible to color the lattice in two colors, or "`No`" otherwise.

Examples

standard input	standard output

`2 2`
`1 0`
`0 1`
	`Yes`

`3 2`
`1 0`
`0 1`
`1 1`
	`No`

`2 1`
`1`
`-3`
	`Yes`

Note

In the first sample, a chessboard coloring is a valid coloring of the lattice.

In the second sample, there exists no valid coloring of vectors $(0, 0)$, $(1, 0)$, $(1, 1)$.

Problem J. Judging the Trick

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

Tricky Ricky is a world-famous magician. He even claims that he can overcome the laws of physics and geometry.

In his latest trick he covers a $w \times h$ rectangle with n triangles fully contained within the rectangle. Sounds easy? But what would you say if he told you that the total area of those triangles is smaller than $w \cdot h$?

You are of that kind of people that are not very fun at parties and magic shows, so you want to prove him that this is impossible. Given a description of n triangles, find any point of the rectangle which does not belong to any triangle interior or border.

Input

The first line of the input contains three integers n, w and h ($1 \le n \le 100\,000$, $1 \le w, h \le 10\,000$) — the number of triangles and the dimensions of the rectangle.

The following n lines contain descriptions of triangle vertices. Each line contains six integers $x_{i,1}$, $y_{i,1}$, $x_{i,2}$, $y_{i,2}$, $x_{i,3}$ and $y_{i,3}$ ($0 \le x_{i,j} \le w$, $0 \le y_{i,j} \le h$). It is guaranteed that all given triangles are non-degenerate, and the total area of triangles is smaller than $w \cdot h$.

Output

Print two real numbers x and y ($0 \le x \le w$, $0 \le y \le h$) defining the coordinates of a point that lies strictly outside of all of the triangles. The numbers should be printed as decimal fractions with **at most** 9 digits after decimal point. Note that usage of exponential format **is not allowed**. Your answer will be considered correct if the given point does not belong to any triangle interior or border. Please, note that your answer will be verified with no absolute or relative tolerance.

It is guaranteed (by Euclid and some other guys) that such point always exists.

Example

standard input	standard output

`5 4 3`
`0 0 3 0 0 2`
`3 3 0 1 0 3`
`1 1 3 1 2 3`
`3 0 4 0 4 3`
`4 3 3 2 4 1`
	`1.1 1.6`

Note

An illustration for the sample is given below.