Problem A. LaLa and Magic Circle (LiLi Version)

Input file:	`standard input`
Output file:	`standard output`
Time limit:	10 seconds
Memory limit:	1024 megabytes

This is an output-only problem.

LaLa has a pile of magic circles in her laboratory.

A magic circle can be represented as a simple polygon drawn with special ink, and is **usable** if it is convex. i.e. all of its internal angles are equal or less than π.

LaLa plans to turn every magic circle into a usable one. However, they may lose all their magical power if done incorrectly. Thankfully, LaLa has the perfect magical tool for that.

The tool works as follows. When you toss in a magic circle, if the circle is usable, the tool reports that it is. Otherwise, it takes two distinct points u and v such that

- u and v lie on the boundary of the convex hull of the magic circle, and
- none of the points on the path from u to v through the boundary of the magic circle in counterclockwise order lie on the boundary of the convex hull of the magic circle, except for u and v.

Then, it rotates the u-v path by π around the midpoint of u and v. In other words, for each point w on the u-v path, w becomes $u + v - w$ where the addition is done coordinate-wise over the two dimensional coordinate system over the paper the magic circle is drawn on. Note that the result of this modification is also a simple polygon.

Little did LaLa know, LaLa's sister, LiLi, overheard LaLa's plan. Knowing how lazy LaLa is, as a prank, LiLi will sneak in a magic circle that takes large amount of applications of the tool to make it usable. More specifically, LiLi will add a magic circle to the pile which is a union of equal or less than 1 000 line segments and the tool can perform some sequence of modifications with between 120 000 and 1 000 000 steps that turns it circle into a usable magic circle.

Write a program to help LiLi compute one such magic circle.

Output

The output should be in the following format:

$$
\begin{array}{cccc}
N & & & \\
x_0 & y_0 & & \\
x_1 & y_1 & & \\
 & \vdots & & \\
x_{N-1} & y_{N-1} & & \\
Q & & & \\
a_0 & b_0 & c_0 & d_0 \\
a_1 & b_1 & c_1 & d_1 \\
 & & \vdots & \\
a_{Q-1} & b_{Q-1} & c_{Q-1} & d_{Q-1}
\end{array}
$$

where the initial magic circle is the union of N line segments connecting points (x_i, y_i) and $(x_{(i+1 \bmod N)}, y_{(i+1 \bmod N)})$ for all integers $0 \le i < N$, and it has a sequence of modifications by the tool of length Q such that i-th modification choose the counterclockwise path from point (a_i, b_i) to (c_i, d_i).

The output should satisfy the following constraints:

- All the numbers in the output are integers.
- $3 \le N \le 1\,000$
- $120\,000 \le Q \le 1\,000\,000$
- $0 \le x_i, y_i, a_j, b_j, c_j, d_j \le 1\,000\,000\,000$ for all integers $0 \le i < N$ and $0 \le j < Q$.
- $x_i \ne x_j$ or $y_i \ne y_j$ for all integers $0 \le i < j < N$.
- $a_i \ne c_i$ or $b_i \ne d_i$ for all integers $0 \le i < Q$.
- The sequence of points $(x_0, y_0), (x_1, y_1), \cdots, (x_{N-1}, y_{N-1})$ defines a counterclockwise traversal of the boundary of a simple polygon.
- Choosing the counterclockwise path from (a_i, b_i) to (c_i, d_i) for the modification on the magic circle after the first i modifications is valid for all integers $0 \le i < Q$.
- The final magic circle is usable.

Note that it's allowed to have two consecutive segments meeting at the angle π. Also note that it is not required to minimize any number, your program just have to satisfy all the output constraints.

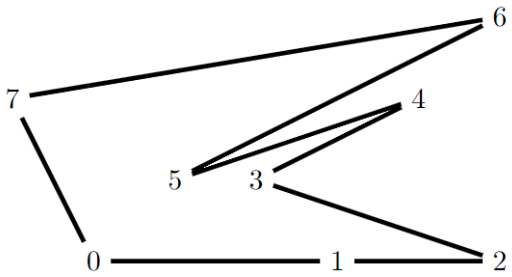
Example

standard input	standard output
	8 1 0 4 0 6 0 3 1 5 2 2 1 6 3 0 2 4 6 0 6 3 10 2 6 3 10 2 9 4 9 4 0 2

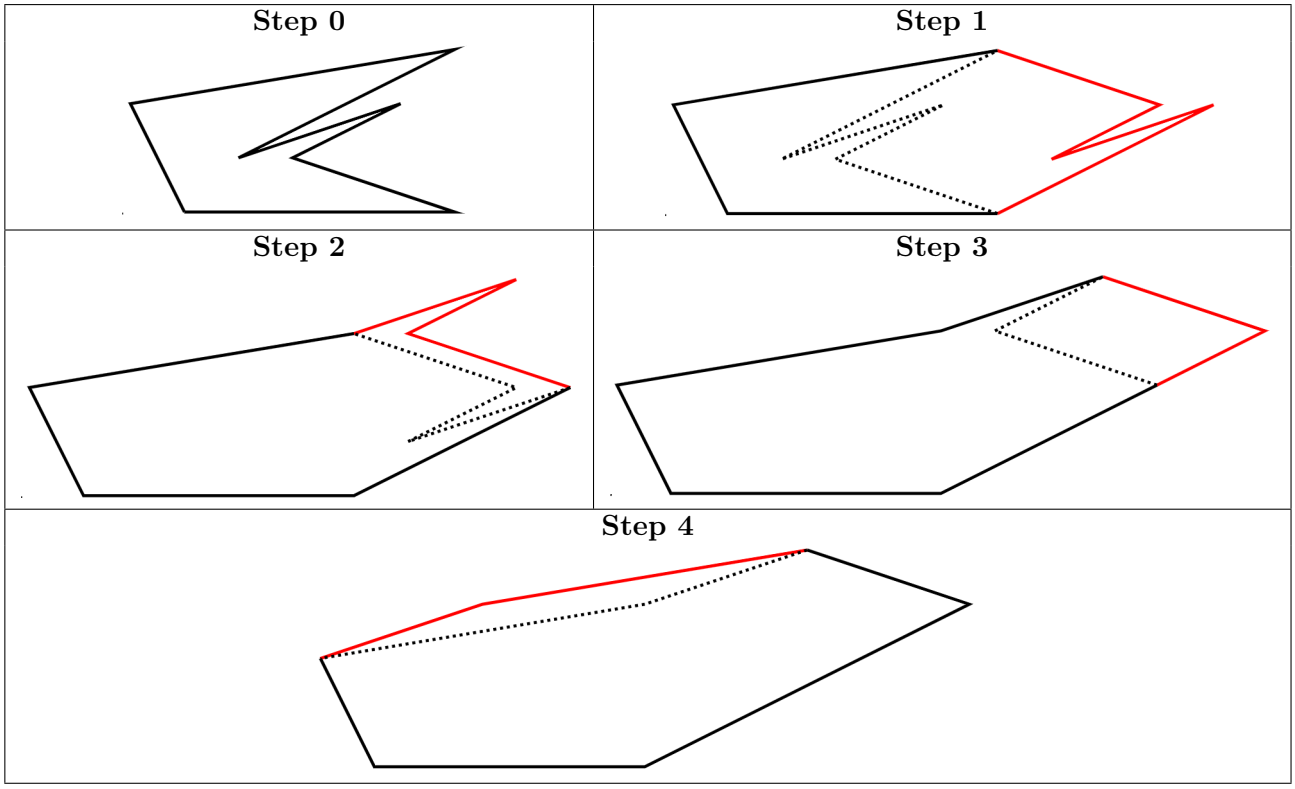
Note

Please note that the sample output above does not satisfy the condition $120\,000 \le Q \le 1\,000\,000$, thus it will give Wrong Answer verdict upon submission. It is there only to present the output format.

The following illustrates the initial magic circle for the sample output.



The following illustrates the sequence of usage of the tool to make it usable. The dotted part of the boundary is the path modified by the tool, which becomes the red part after the modification.



Problem B. LaLa and Magic Circle (LaLa Version)

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	1024 megabytes

LaLa has a pile of magic circles in her laboratory.

A magic circle can be represented as a simple polygon drawn with special ink, and is **usable** if and only if it is convex. i.e. all of its internal angles are equal or less than π.

LaLa plans to turn every magic circle into a usable one. However, it may lose all its magical power if done incorrectly. Thankfully, LaLa has the perfect magical tool for that.

The tool works as follows. When you toss in a magic circle, if it's usable, it reports that it is. Otherwise, it takes two distinct points u and v such that

- u and v lie on the boundary of the convex hull of the magic circle, and
- none of the points on the path from u to v through the boundary of the magic circle in counterclockwise order lie on the boundary of the convex hull of the magic circle, except for u and v.

And then it rotates the u-v path by π around the midpoint of u and v. In other words, for each point w on the u-v path, w becomes $u + v - w$ where the addition is done coordinate-wise over the two dimensional coordinate system over the paper the magic circle is drawn on. Note that the result of this modification is also a simple polygon.

LaLa got annoyed by how long it takes to convert them. In order to finish and take a nap ASAP, LaLa made the following observations.

1. A magic circle always turns into a usable one within a finite number of applications of the tool.
2. The set of points on the final magic circle is independent of the intermediate modifications. In other words, the shape and location of the final magic circle is a function of the initial magic circle.

Therefore, LaLa doesn't have to manually turn magic circles into usable ones with the tool. Instead, LaLa will compute the usable magic circle that can be made from the initial magic circle by a sequence of modifications by the tool and modify it in one go.

Write a program to help LaLa compute the final magic circle so that she can go take a nap.

Input

The input is given in the following format:

$$\begin{array}{ll} N & \\ x_0 & y_0 \\ x_1 & y_1 \\ & \vdots \\ x_{N-1} & y_{N-1} \end{array}$$

where the initial magic circle is the union of N line segments connecting points (x_i, y_i) and $(x_{(i+1 \bmod N)}, y_{(i+1 \bmod N)})$ for all integers $0 \leq i < N$.

The input satisfies the following constraints:

- All numbers in the input are integers.

- $3 \leq N \leq 100\,000$
- $0 \leq x_i \leq 300\,000$ and $0 \leq y_i \leq 300\,000$ for all integers $0 \leq i < N$.
- $x_i \neq x_j$ or $y_i \neq y_j$ for all integers $0 \leq i < j < N$.
- The input defines a counterclockwise traversal of the boundary of a simple polygon. In particular, it does not intersect with itself.

Output

The output should be in the following format:

$$\begin{array}{ll} M & \\ z_0 & w_0 \\ z_1 & w_1 \\ & \vdots \\ z_{M-1} & w_{M-1} \end{array}$$

where the final usable magic circle is the union of M line segments connecting points (z_i, w_i) and $(z_{(i+1 \bmod M)}, w_{(i+1 \bmod M)})$ for all integers $0 \leq i < M$,

The output should satisfy the following constraints:

- All the numbers in the output are integers.
- The point (z_0, w_0) is lexicographically smaller than the point (z_i, w_i) for all integers $1 \leq i < M$. i.e. $(z_0 < z_i)$ or $(z_0 = z_i$ and $w_0 < w_i)$.
- Points (z_i, w_i), $(z_{(i+1 \bmod M)}, w_{(i+1 \bmod M)})$, and $(z_{(i+2 \bmod M)}, w_{(i+2 \bmod M)})$ are not collinear for all integers $0 \leq i < M$.
- The output defines a counterclockwise traversal of the boundary of a convex polygon.

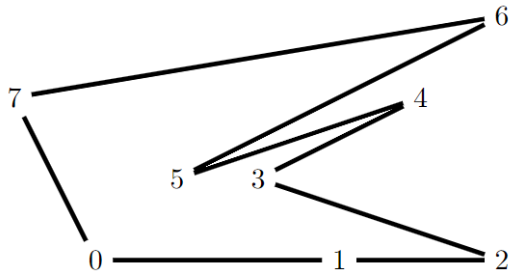
It can be proved that the output satisfying the above constraints is unique.

Example

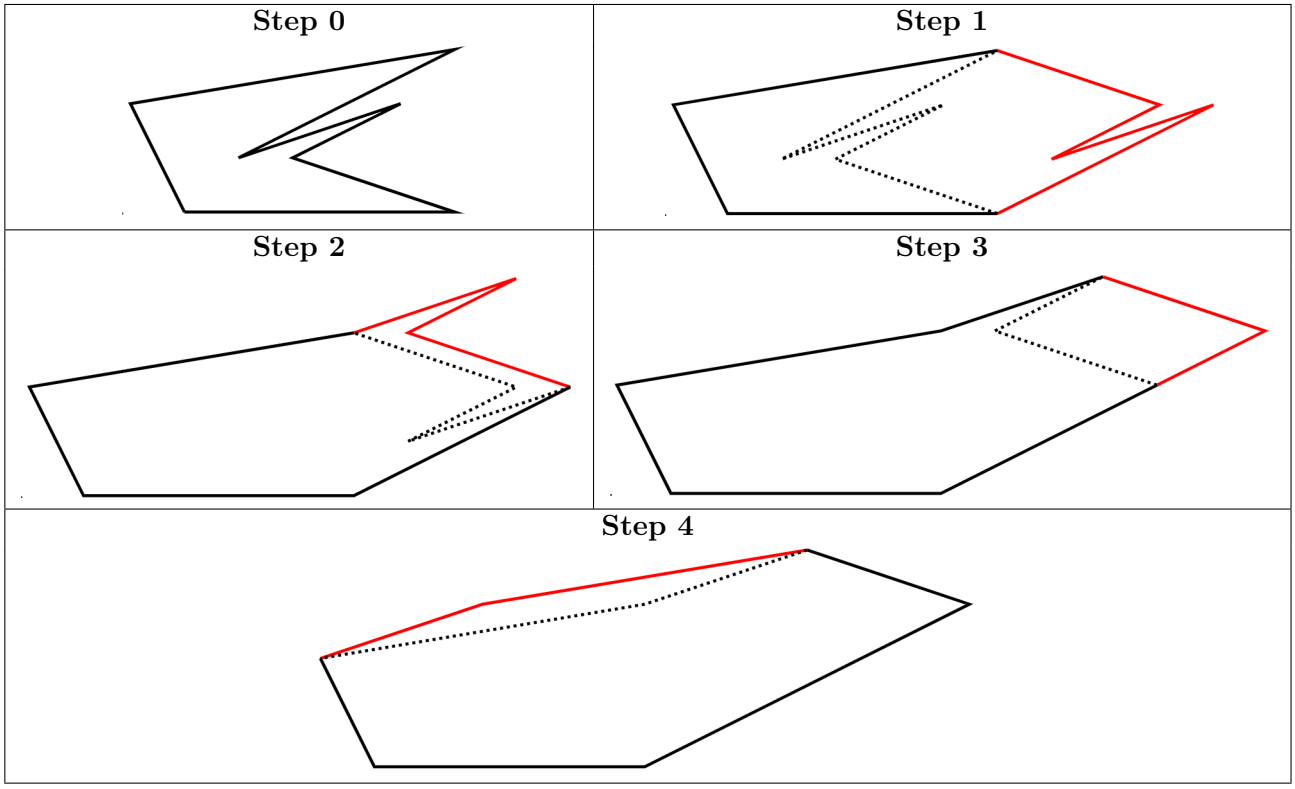
standard input	standard output
8 1 0 4 0 6 0 3 1 5 2 2 1 6 3 0 2	6 0 2 1 0 6 0 12 3 9 4 3 3

Note

The following illustrates the initial magic circle for the first sample.



The following illustrates the sequence of usage of the tool to make it usable. The dotted part of the boundary is the path modified by the tool, which becomes the red part after the modification.



Problem C. LaLa and Lamp

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	1024 megabytes

When LaLa laid down on her pet Leo's back to fall asleep, she noticed that the lamp is all messed up, which must have been the act of her sister LiLi.

The lamp can be modeled as a regular triangular grid where each cell contains a bulb which is either **on** or **off**.

LaLa wants to turn off the lamp (that is, set the state of all bulbs to off). LaLa can pick any of the three directions parallel to the side of a lamp, pick any row parallel to that direction, and then flip the state of all the bulbs in the row (on to off and off to on) with her magic. LaLa also could just walk over to the lamp and manually turn every bulb off, but she would prefer not to.

Write a program that determines whether LaLa can turn off the lamp with her magic.

Input

The input is given in the following format:

N
S_0
S_1
$\vdots$
S_{N-1}

where N is the number of bulbs in a side of the lamp, and S_i is the binary string of length $i+1$ representing the initial states of bulbs in the i-th row, where the j-th character of S_i is '1' if and only if the j-th bulb is on.

The input satisfies the following constraint:

- N is an integer.
- $2 \leq N \leq 2\,000$

Output

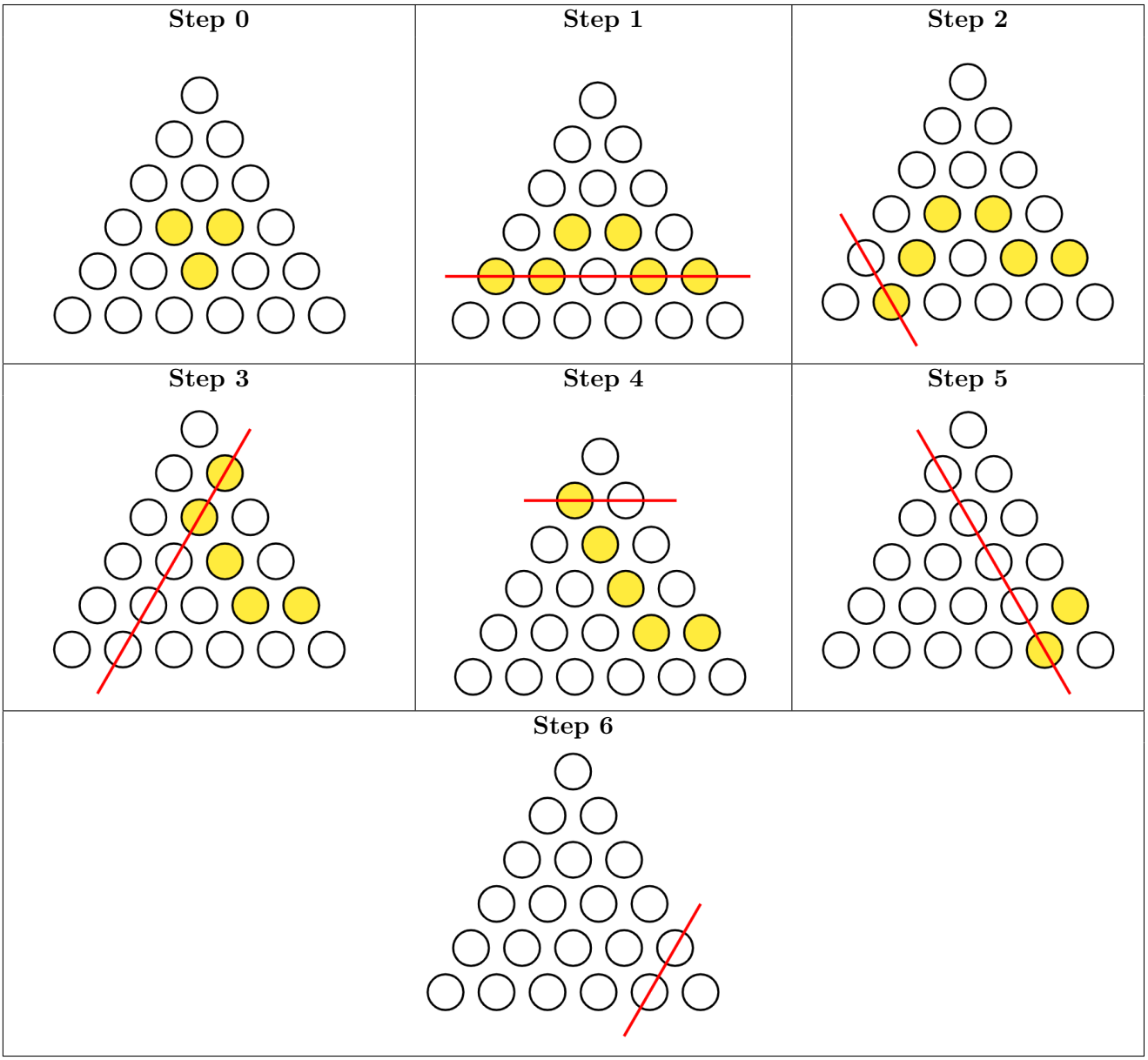
If LaLa can turn off the lamp with magic, print a single string "Yes". Otherwise, print a single string "No". You may print each character in either case (lower or upper).

Example

standard input	standard output
6 0 00 000 0110 00100 000000	Yes

Note

The following illustrates a sequence of magic LaLa should cast to turn off the lamp given in the sample. Empty circles denote the bulbs that are off, yellow circles denote the bulbs that are on, and red line is the choosen row for magic.


Step 0
Step 1
Step 2
Step 3
Step 4
Step 5
Step 6

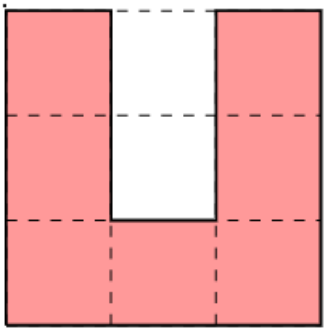
Problem D. LaLa and Magic Stone

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

One day, LaLa realized that she ran out of magic stone components for her magic tools. (Think of them as a battery in our world.) So LaLa rushed to a nearby store and bought a slab of magic stone.

LaLa wants to cut the slab into magic stone components. The slab consists of $N \times M$ cells. Unfortunately, some cells are incompatible with LaLa's magic tools.

The required magic stone component is a 7-cell U-shaped piece.



As LaLa forgot to buy magic stone glue, LaLa can't merge smaller pieces to form the required shape.

Furthermore, since LaLa hates wasting magic stones, LaLa will be satisfied if and only if the slab is cut so that every single compatible cell belongs to a required shape.

Write a program that computes the number of ways to cut the slab so that LaLa is satisfied, modulo 998 244 353. Two ways to cut pieces are different if and only if there exist two compatible cells such that they belong to the same piece in one and to different pieces in the other.

Input

The input is given in the following format:

N M
S_0
S_1
$\vdots$
S_{N-1}

where N is the number of rows of the slab, M is the number of columns, and S_i is a binary string of length M where j-th character is '1' if and only if the cell at the i-th row from the top and j-th column from the left is incompatible.

The input satisfies the following constraints:

- N and M are integers.
- $3 \leq N, M \leq 1\,000$

Output

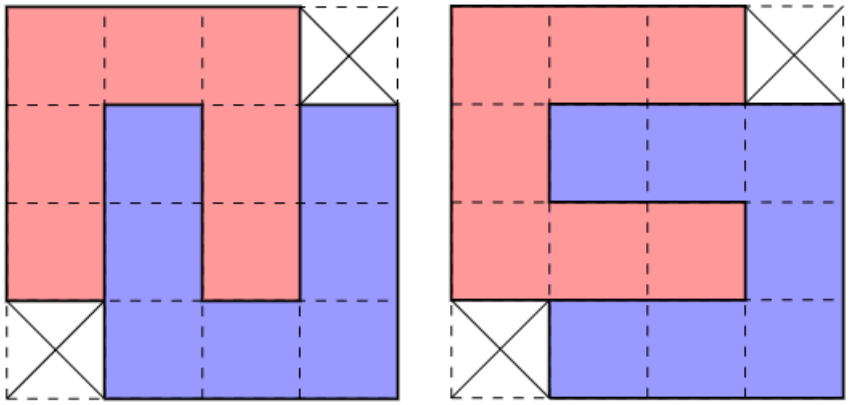
The output should be a integer equal to the number of ways to cut the slab so that LaLa is satisfied, modulo 998 244 353.

Examples

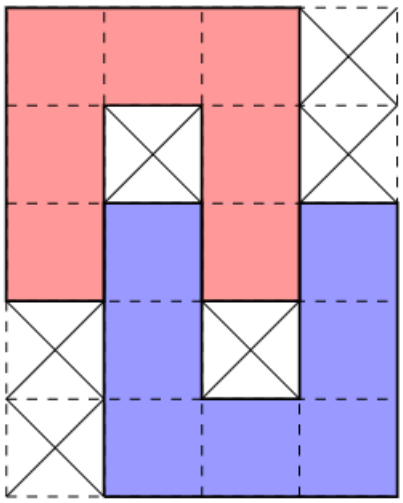
standard input	standard output
4 4 0001 0000 0000 1000	2
5 4 0001 0101 0000 1010 1000	1

Note

The following illustrates two possible ways to cut the slab in the first sample.



The following illustrates the only way to cut the slab in the second sample.



Problem E. LaLa and Monster Hunting (Part 1)

Input file:	`standard input`
Output file:	`standard output`
Time limit:	5 seconds
Memory limit:	1024 megabytes

A dreadful monster has been witnessed in a forest near the city of magic Sharia, and a group of valorous adventurers will hunt it down in few days before it hurts anyone. However, LaLa knows that the real reason those adventurers are willing to take the risk is to obtain the rare magic stone that the monster is known to produce in its intestines. LaLa would like to obtain the magic stone before they do, as it is known to be quite beautiful.

LaLa will first locate the monster with her magic. LaLa has placed a bunch of magic tools within the forest, each of which has some power associated with it.

Consider the circles centered at each magic tool with radius equal to its power. LaLa's magic will successfully locate the monster if and only if the convex hull of the circles contains the location of the monster.

Write a program that determines whether LaLa will be able to locate the monster.

Input

The input is given in the following format:

$$\begin{array}{lll} N & & \\ x_0 & y_0 & r_0 \\ x_1 & y_1 & r_1 \\ & \vdots & \\ x_{N-1} & y_{N-1} & r_{N-1} \end{array}$$

where N is the number of magic tools placed in the forest, the i-th of which is located at (x_i, y_i) and has power r_i. Here, assume that the forest is a two-dimensional plane where the monster is located at $(0, 0)$.

The input satisfies the following constraints:

- All the numbers in the input are integers.
- $1 \leq N \leq 1\,000\,000$
- $-1\,000\,000 \leq x_i, y_i \leq 1\,000\,000$ for all integers $0 \leq i < N$
- $0 \leq r_i \leq 1\,000\,000$ for all integers $0 \leq i < N$
- **The distance between point $(0, 0)$ and the boundary of the convex hull of N circles, i-th of which is centered at (x_i, y_i) and has radius r_i, is at least 1.**

Output

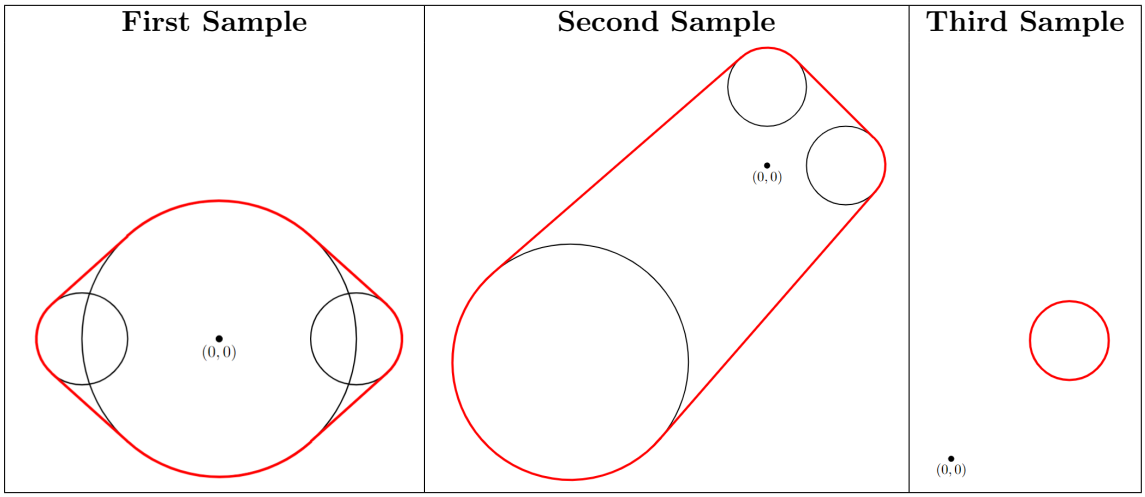
If LaLa's magic will successfully locate the monster, print a single string "Yes". Otherwise, print a single string "No". You may print each character in either case (lower or upper).

Examples

standard input	standard output
3 -3 0 1 0 0 3 3 0 1	Yes
3 2 0 1 0 2 1 -5 -5 3	Yes
1 3 3 1	No

Note

The following illustrates the configuration of the magic tools for the sample tests. The red curve denotes the boundary of the convex hull.



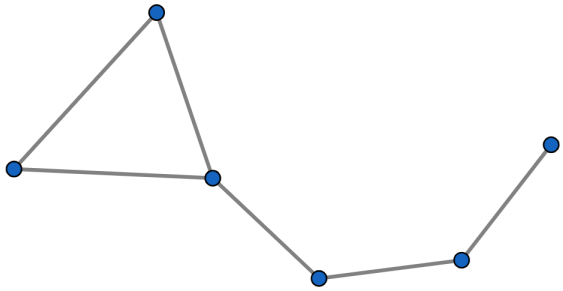
Problem F. LaLa and Monster Hunting (Part 2)

Input file:	`standard input`
Output file:	`standard output`
Time limit:	6 seconds
Memory limit:	1024 megabytes

A dreadful monster has been witnessed in a forest near the city of magic Sharia, and a group of valorous adventurers will hunt it down in few days before it hurt anyone. However, LaLa knows that the real reason those adventurers are willing to take the risk is to obtain the rare magic stone that the monster is known to produce in its intestines. LaLa would like to obtain the magic stone before they do, as it is known to be quite beautiful.

Currently, LaLa knows a rough estimate of the location of the monster. However, the monster excels at camouflage, so it's really hard to hunt it down when it's hiding in the network of branches.

For the sake of simplicity, we'll model the monster as a graph G with 6 vertices described below:



The network of branches can be modeled as a simple graph H. A **candidate** is a subgraph of H that is isomorphic to G. In other words, it is a graph obtained by deleting some edges from H, and then deleting some vertices that none of the remaining edges are incident to, whose vertices can be renumbered so that it coincides with G. LaLa will now have to examine all possible candidates to search and hunt the monster down.

Write a program that computes the number of candidates LaLa will have to examine, modulo 998 244 353.

Input

The input describes the branch network H and is given in the following format:

N	M
u_0	v_0
u_1	v_1
	$\vdots$
u_{M-1}	v_{M-1}

where N is the number of joints, numbered from 0 to $N-1$ and M is the number of branches, i-th of which connects the joints u_i and v_i.

The input satisfies the following constraints:

- All the numbers in the input are integers.
- $2 \le N \le 100\,000$
- $0 \le M \le 100\,000$
- $0 \le u_i < v_i < N$ for all integers $0 \le i < M$
- $u_i \ne u_j$ or $v_i \ne v_j$ for all integers $0 \le i < j < M$

Note that the network is not necessarily connected.

Output

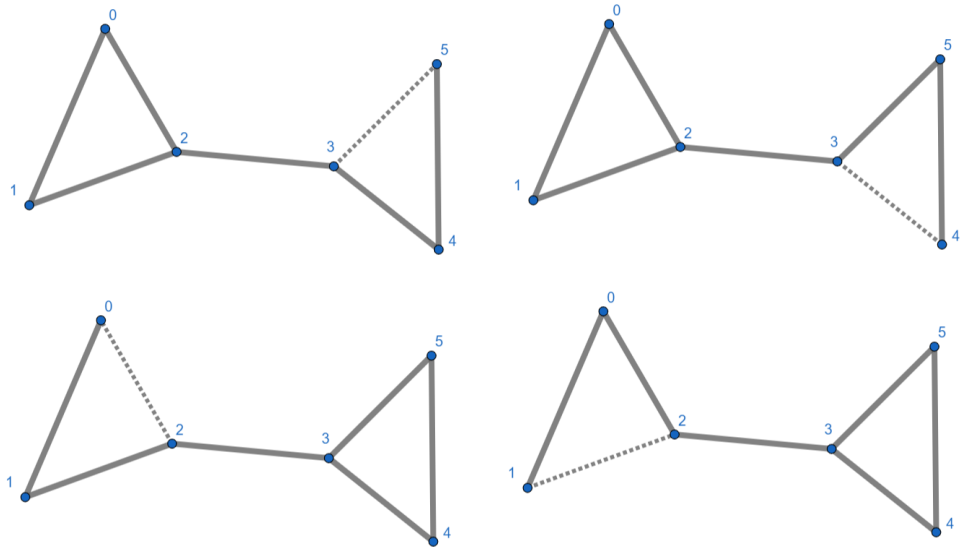
The output should be a single integer equal to the number of candidates, modulo 998 244 353.

Examples

standard input	standard output
6 7 0 1 1 2 0 2 2 3 3 4 4 5 3 5	4
6 15 0 1 0 2 0 3 0 4 0 5 1 2 1 3 1 4 1 5 2 3 2 4 2 5 3 4 3 5 4 5	360

Note

The followings illustrate the 4 candidates (the regular edges) of the branch network in the first sample test.



Problem G. LaLa and Divination Magic

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	1024 megabytes

LaLa specializes in divination magic.

Let's say there are M events $E_0, \cdots, E_{M-1}$ that LaLa's interested in forecasting. Each event is associated with one of two outcomes: **catastrophe** or **salvation**.

With a single use of LaLa's divination magic, LaLa obtains the knowledge of one of the following four forms:

1. Knowledge$(i, j, 1)$: either E_i is catastrophe or E_j is catastrophe (possibly both).
2. Knowledge$(i, j, 2)$: either E_i is salvation or E_j is catastrophe (possibly both).
3. Knowledge$(i, j, 3)$: either E_i is catastrophe or E_j is salvation (possibly both).
4. Knowledge$(i, j, 4)$: either E_i is salvation or E_j is salvation (possibly both).

LaLa cast her magic several times, possibly 0, and wrote down all M-tuples of the outcomes of events that are consistent with her knowledge: this is called the **result of the forecasting**. And then, LaLa fell asleep.

When LaLa woke up, she found out that her pet, Leo, ruined all the predictions of her magic. Though LaLa was able to find the result of her forecasting, she is unsure if that data was ruined by Leo as well.

Write a program that determines whether there exists a set of predictions of LaLa's magic whose result of the forecasting matches the one LaLa has, and finds a possible set of predictions if there is one.

Input

The input is given in the following format:

N M
S_0
S_1
$\vdots$
S_{N-1}

where N is the number of outcomes in the result, M of events, and S_i is a binary string of length M where j-th character is '1' if and only if the i-th result forecasts that j-th event will be in salvation.

The input satisfies the following constraints:

- N and M are integers.
- $1 \leq N, M \leq 2\,000$
- $S_i \neq S_j$ for all integers $0 \leq i < j < N$.

Output

If there is no such prediction, the output should be a single integer -1.

Otherwise, the output should be in the following format:

K		
I_0	J_0	t_0
I_1	J_1	t_1
	$\vdots$	
I_{K-1}	J_{K-1}	t_{K-1}

where K is the size of a possible set S of predictions, and, for each $0 \leq i < K$, S contains the prediction Knowledge(I_i, J_i, t_i).

The output should satisfy the following constraint:

- $0 \leq K \leq 2 \cdot M^2$

It can be proved that if there is such a set of predictions, there also is one satisfying the constraint.

Examples

standard input	standard output
2 1 1 0	0
3 3 101 011 111	6 0 2 3 0 1 4 0 2 4 1 2 3 1 2 4 2 2 4

Problem H. LaLa and Harvesting

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	1024 megabytes

Every other winter, LaLa helps her aunt, Aisha, harvest a crop native to the Biheiril Kingdom.

The crop can be modeled as a graph where an edge corresponds to a branch, a vertex to a joint, and a fruit with tastiness T_u grows on each joint u.

The cultivation of the crop can be divided into three phases.

1. At the start of the first phase, a seed is planted on an open field. The seed eventually grows to form a cactus graph, a simple connected graph where every edge belongs to at most one cycle. This is the only phase where new joints grow.
2. Consider the DFS-tree of the cactus grown during the first phase. At the start of the second phase, Aisha encloses the crop with a ring-shaped framework by attaching the leaves (vertices of degree 1, where only the edges in the DFS-tree are counted for degree, which possibly includes the root of the DFS-tree) to the ring. This forces the crop to grow additional branches joining adjacent joints on the framework, forming a cycle graph, and to start producing the fruits. Please read the input specification for detailed description.
3. At the start of the third phase, Aisha connects to the framework a magic tool that feeds the crop with the constant flow of extra nutrition. As a result, some new branches grow. The union of the branches grown during this phase form a very dense tree: every non-leaf vertex of the tree has degree ≥ 12. The harvesting begins at the end of this phase.

LaLa's goal is to maximize the sum of tastiness of the fruits she has harvested. However, LaLa is not allowed to harvest fruits at two adjacent vertices, as it will stress and kill all the trunks directly connecting them.

Write a program that computes a set of fruits LaLa can harvest which has maximum possible sum of tastiness.

Input

The input describes the state of a crop and is given in the following format:

$$
\begin{array}{lll}
N & M & \\
T_0 & T_1 & \cdots \quad T_{N-1} \\
u_0 & v_0 & \\
u_1 & v_1 & \\
\multicolumn{2}{c}{\vdots} & \\
u_{M-1} & v_{M-1} & \\
K & & \\
x_0 & y_0 & \\
x_1 & y_1 & \\
\multicolumn{2}{c}{\vdots} & \\
x_{K-1} & y_{K-1} &
\end{array}
$$

where N is the number of joints, numbered from 0 to $N-1$, M the number of branches grown during the first phase, and the i-th branch connects the joints u_i and v_i for all integers $0 \leq i < M$.

Consider the DFS-tree built over the cactus graph grown during the first phase where the DFS traversal starts at joint 0 and the order of the neighbors of each joints is given by the input order i.e. the traversal

prioritizes branches given sooner in the input. Note that the above description uniquely defines a DFS-traversal and its corresponding DFS-tree. Let $c_0, \cdots, c_{l-1}$ be the subsequence of the DFS-order consisting of all joints which has degree 1 in the DFS-tree. Then the cycle graph grown during the second phase is defined by the l edges $(c_0, c_1), \cdots, (c_{l-1}, c_0)$.

In addition, K is the number of branches grown on the third phase, i-th of which connects the joints x_i and y_i for all integers $0 \leq i < K$.

The input satisfies the following constraints:

- All the numbers in the input are integers.
- $2 \leq N \leq 500$, $N - 1 \leq M \leq 2N$, $1 \leq K \leq \min(N - 1, 100)$
- $1 \leq T_u \leq 200\,000$ for all integers $0 \leq u < N$
- $0 \leq u_i < v_i < N$ for all integers $0 \leq i < M$
- $u_i \neq u_j$ or $v_i \neq v_j$ for all integers $0 \leq i < j < M$
- $0 \leq x_i < y_i < N$ for all integers $0 \leq i < K$
- $x_i \neq x_j$ or $y_i \neq y_j$ for all integers $0 \leq i < j < K$
- The union of edges (u_i, v_i) forms a cactus over the vertices $\{0, 1, \cdots, N - 1\}$.
- The union of edges (x_i, y_i) forms a tree over the set of vertices $\{x_0, y_0, x_1, y_1, \cdots, x_{K-1}, y_{K-1}\}$ where each vertex with degree greater than 1 has degree at least 12.

Output

The output should be in the following format:

W	L		
s_0	s_1	$\cdots$	s_{L-1}

where W is the maximum sum of tastiness of a set $s = \{s_0, \cdots, s_{L-1}\}$ of fruits LaLa can harvest from the crop without harming any branches.

The output should satisfy the following constraints:

- All the numbers in the output are integers.
- $0 \leq L \leq N$
- $0 \leq s_0 < s_1 < \cdots < s_{L-1} < N$
- There are no edges directly connecting the vertices s_i and s_j for all integers $0 \leq i < j < L$.
- $\sum_{i=0}^{L-1} T_{s_i} = W$

Example

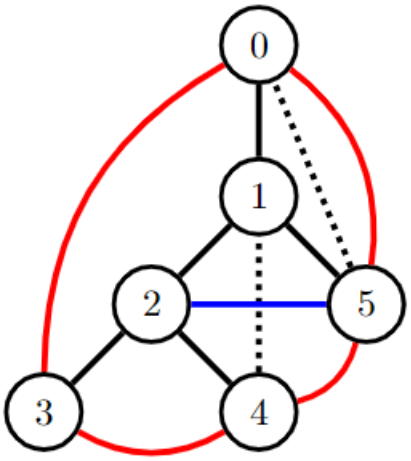
standard input	standard output
6 7 1 1 1 1 1 1 0 1 1 2 2 3 2 4 1 5 1 4 0 5 1 2 5	2 2 0 4

Note

The following illustrates the crop described in the sample test.

- The regular black edges denote the branches grown during the first phase that is part of the DFS-tree.
- The dotted black edges denote the branches grown during the first phase that is NOT part of the DFS-tree.
- The red edges denote the branches grown during the second phase.
- The blue edges denote the branches grown during the third phase.

Note that there can be multiple branches directly connecting the same pair of joints.



This page is intentionally left blank.

Problem I. LaLa and Spirit Summoning

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	1024 megabytes

LaLa's younger sister LiLi is helping LaLa cast the spirit summoning magic.

While LaLa was asleep, LiLi had already built a prototype of the spirit to summon. The spirit consists of N magic joints which allow any magic bars attached to them to freely move around them, and M magic bars of various colors, each of which connects two magic joints and whose length can be adjusted to any non-negative real number before the summoning (but not after).

When it comes to spirit summoning, LaLa has a far higher standard than LiLi. Of course, LaLa was not satisfied with LiLi's work whatsoever. LaLa would like to fabulize the prototype by getting rid of some magic bars so that

1. the spirit is **beautiful**, which means there should not be two magic bars of the same color present, and
2. the spirit is as easy to control as possible, which means the **degree of freedom** of the spirit must be minimum over all beautiful spirits obtainable by eliminating some magic bars. Note that the minimum always exists as she can always eliminate all magic bars to create a beautiful spirit. See the note below for the exact definition of the degree of freedom.

Write a program that computes the degree of freedom of the spirit fabulized by LaLa.

Input

The input describes the prototype spirit made by LiLi and is given in the following format:

N	M	
u_0	v_0	c_0
u_1	v_1	c_1
	$\vdots$	
u_{M-1}	v_{M-1}	c_{M-1}

where N is the number of magic joints, numbered from 0 to $N-1$, M is the number of magic bars, and for each integer $0 \le i < M$, the i-th magic bar has color c_i and connects the magic joint u_i and v_i.

The input satisfies the following constraints:

- All the numbers in the input are integers.
- $2 \le N \le 200$
- $0 \le M \le 1\,000$
- $0 \le u_i < v_i < N$ and $0 \le c_i < M$ for all integers $0 \le i < M$

Note that there can be multiple magic bars connecting the same pair of magic joints.

Output

The output should be a single integer equal to the degree of freedom of the spirit fabulized by LaLa.

Examples

standard input	standard output
3 3 0 1 0 0 2 0 1 2 0	5
3 3 0 1 0 0 2 1 1 2 2	3
4 4 0 1 0 1 2 1 2 3 2 0 3 3	4
5 4 0 1 0 1 2 1 2 3 2 3 4 3	6

Note

Intuitively, the degree of freedom is the number of axis of motions preserving edge lengths of the spirit embedded on a plane.

More formally, let E be an assignment of planar coordinates (we'll call this an **embedding**) to all magic joints of a spirit. Note that such an embedding can be identified with an element in $\mathbb{R}^{2N}$ by concatenating all coordinates, where N is the number of magic joints.

Let $C(E)$ be the set of embeddings continuously reachable from E as an element of $\mathbb{R}^{2N}$ while preserving edge lengths. i.e. for each element E' of $C(E)$ and each magic bars of the spirit connecting magic joints u and v, the euclidean distance between u and v must be the same in E and E'.

The **degree of freedom** of E is the minimum non-negative integer k such that there exists a continuous bijective mapping $F : D \to C(E)$ where D is a connected subset of $\mathbb{R}^k$.

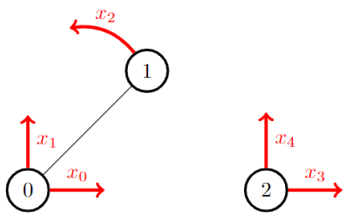
The **degree of freedom** of a spirit is the maximum degree of freedom over all such embeddings E.

The following illustrate the spirit fabulized by LaLa along with one of the optimal embedding and the mapping F for each sample tests in order.

1. $k = 5$, $D = \mathbb{R}^2 \times [0, 2\pi) \times \mathbb{R}^2$

   $F : (x_0, x_1, x_2, x_3, x_4) \mapsto \langle (x_0, x_1), (x_0, x_1) + (\cos x_2, \sin x_2), (x_3, x_4) \rangle$

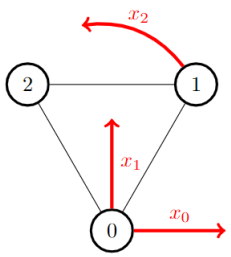
   The following illustrates the 5 degrees of freedom associated with each variables.



2. $k = 3$, $D = \mathbb{R}^2 \times [0, 2\pi)$

$F : (x_0, x_1, x_2) \mapsto \langle (x_0, x_1), (x_0, x_1) + (\cos x_2, \sin x_2), (x_0, x_1) + (\cos(\frac{\pi}{3} + x_2), \sin(\frac{\pi}{3} + x_2)) \rangle$

The following illustrates the 3 degrees of freedom associated with each variables.

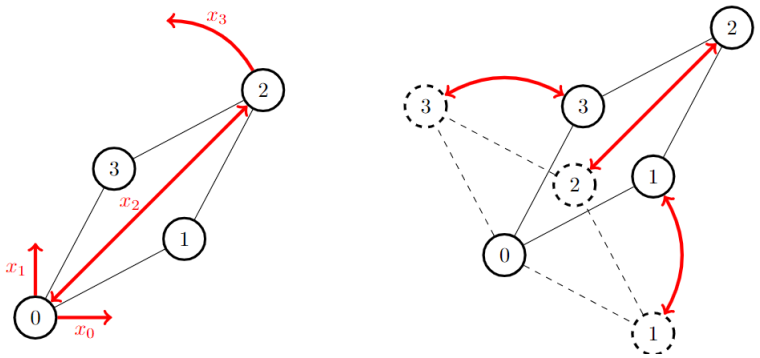


3. $k = 4$, $D = \mathbb{R}^2 \times (((0, 2] \times [0, 2\pi)) \cup (\{0\} \times [0, \pi)))$

   $F : (x_0, x_1, x_2, x_3) \mapsto \langle P_0, P_0 + \frac{x_2}{2} P_1 + \sqrt{1 - \frac{x_2^2}{4}} P_2, P_0 + x_2 P_1, P_0 + \frac{x_2}{2} P_1 - \sqrt{1 - \frac{x_2^2}{4}} P_2 \rangle$

   where $P_0 = (x_0, x_1)$, $P_1 = (\cos x_3, \sin x_3)$ and $P_2 = (\sin x_3, -\cos x_3)$.

   The following figure on the left illustrates the 4 degrees of freedom associated with each variables. Note that the motion associated with the variable x_2 is non-rigid. The one on the right illustrates the motion associated with x_2 in detail.

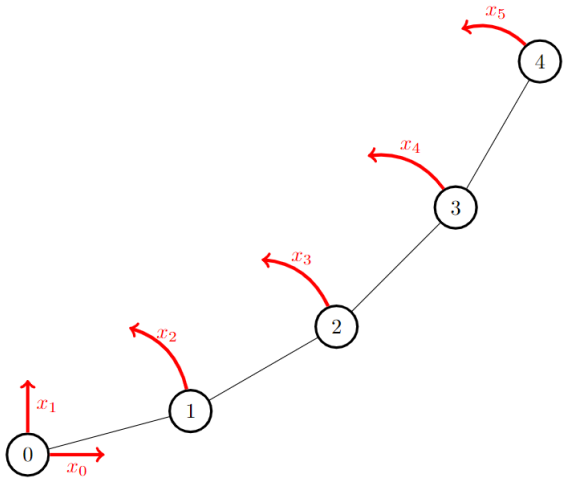


4. $k = 6$, $D = \mathbb{R}^2 \times [0, 2\pi)^4$

   $F : (x_0, x_1, x_2, x_3, x_4, x_5) \mapsto \langle P_0, P_1, P_2, P_3, P_4 \rangle$

   where $P_0 = (x_0, x_1)$ and $P_i = P_{i-1} + (\cos x_{i+1}, \sin x_{i+1})$ for all integers $1 \le i \le 4$.

   The following illustrates the 6 degrees of freedom associated with each variables.



Problem J. LaLa and Magical Beast Summoning

Input file:	`standard input`
Output file:	`standard output`
Time limit:	5 seconds
Memory limit:	1024 megabytes

LaLa is about to cast a magical beast summoning magic.

The first thing LaLa do is creating a **summoning field**, which has 3 constants associated with it: **nullity** M, **elasticity** E, and **viscosity** V. Such summoning field is denoted by $\mathcal{F}(M, E, V)$

A magical beast summoning magic is performed over a **summoning cell** within the summoning field, which is square-shaped and is associated with 3 variables: **side length** L, **agility** A, and **intelligence** I. Such summoning cell is denoted by $\mathcal{C}(L, A, I)$.

$\mathcal{C}(L, A, I)$ is in a **null state** if $L = 0$. Otherwise, it is in a **positive state**.

The **density** of $\mathcal{C}(L, A, I)$ in positive state is defined as $(A \times I)/L^2$.

The problem of determining whether a magical beast summoning magic will succeed requires very heavy computation involving solving a system of 9999999999-th order partial differential equations over 9999999999999999 variables. Fortunately, LaLa already did all the math for you!

The magical beast summoning magic over $\mathcal{C}(L, A, I)$ within $\mathcal{F}(M, E, V)$ succeeds if and only if the function valid(M, E, V, L, A, I) defined by the pseudocode in the note section returns true. We'll call such summoning cell **valid**.

Sometimes, LaLa isn't satisfied with the set of summoning cells she has, and wants to generate new ones by combining them. The problem of determining the result of combination of two valid summoning cells $C_0 = \mathcal{C}(L_0, A_0, I_0)$ and $C_1 = \mathcal{C}(L_1, A_1, I_1)$ within $F = \mathcal{F}(M, E, V)$ requires another heavy computation, but thankfully, LaLa already did all the math for you again!

The result of combining two such cells C_0 and C_1 within F, denoted by $\text{Combine}_F(C_0, C_1)$, is given by the function combine$(M, E, V, L_0, A_0, I_0, L_1, A_1, I_1)$ defined by the pseudocode in the note section, which returns a triple L_2, A_2, I_2 satisfying $\mathcal{C}(L_2, A_2, I_2) = \text{Combine}_F(C_0, C_1)$. Here, it can be proved that $\text{Combine}_F(C_0, C_1)$ is also valid. Note that swapping the order of C_0 and C_1 affects the result.

The result of combining $K \geq 3$ cells $C_0, \cdots, C_{K-1}$ within F is given recursively by

$$\text{Combine}_F(C_0, \cdots, C_{K-1}) = \text{Combine}_F(\text{Combine}_F(C_0, \cdots, C_{K-2}), C_{K-1})$$

For the sake of completeness, we define $\text{Combine}_F(C) = C$.

LaLa **is aware of a very special property about the combining operation that allows her to efficiently solve the range density query problem below.** Can you figure it out?

You're given a summoning field $F = \mathcal{F}(M, E, V)$ and an array of N valid summoning cells

$$C_0 = \mathcal{C}(L_0, A_0, I_0), \cdots, C_{N-1} = \mathcal{C}(L_{N-1}, A_{N-1}, I_{N-1})$$

within F. Write a program that processes the following two types of Q queries:

- `1 i L A I`
  - Set $C_i \leftarrow \mathcal{C}(L, A, I)$.
- `2 l r`
  - Let $R = \text{Combine}_F(C_l, \cdots, C_{r-1})$. If R is in the null state, print a single integer -1. Otherwise, print the density of R, modulo M. Here, an irreducible fraction p/q, where p is a non-negative integer and q is a positive integer not divisible by M, modulo M is defined to be the unique integer $p \times q^{-1} \bmod M$ where q^{-1} is the multiplicative inverse of q modulo M. It can be proved that if R is in the positive state, the denominator of the density of R as an irreducible fraction is not divisible by M within the constraints of this problem.

Input

The input is given in the following format:

$$
\begin{array}{lll}
M & E & V \\
N & & \\
L_0 & A_0 & I_0 \\
 & \vdots & \\
L_{N-1} & A_{N-1} & I_{N-1} \\
Q & & \\
q_0 & & \\
\vdots & & \\
q_{Q-1} & &
\end{array}
$$

Here, q_i denotes the i-th query, and is given in the format described in the statement.

The input satisfies the following constraints:

- All the numbers in the input are integers.
- M is a prime such that $900\,000\,000 \leq M \leq 1\,000\,000\,000$
- $1 \leq E, V \leq 100$
- $1 \leq N, Q \leq 100\,000$
- $0 \leq L_i, A_i, I_i < M$ for all integers $0 \leq i < N$
- $\mathcal{C}(L_i, A_i, I_i)$ within $\mathcal{F}(M, E, V)$ is valid for all integers $0 \leq i < N$.
- For each query `1 i L A I`, $0 \leq i < N$, $0 \leq L, A, I < M$, and $\mathcal{C}(L, A, I)$ is valid within $\mathcal{F}(M, E, V)$.
- For each query `2 l r`, $0 \leq l < r \leq N$

Output

For each query of the second type, print its answer in a single line.

Example

standard input	standard output
998244353 1 2 3 2 998244352 3 4 998244351 6 4 929561374 68682991 7 2 0 2 2 0 3 2 1 3 1 1 6 9 998244350 2 0 2 2 0 3 2 1 3	-1 748683259 156877648 748683265 499122176 342244251

Note

The following pseudocode defines the validity of summoning cells and the Combine operation.

Both functions do not modify their arguments

```
function VALID(M, E, V, L, A, I)
    if min(L, A, I) < 0  or  M ≤ max(L, A, I) then
        return False
    end if
    /*Every operations and comparisons below are done mod M*/
    if L = 0  and  (A + I ≠ 0  or  A = I) then
        return False
    end if
    return A^3−A^2L+3A^2I+EAL^2+L^3V+2ALI+EL^2I+3AI^2−LI^2+I^3 ≠ 0
end function
```

```
function COMBINE(M, E, V, L_0, A_0, I_0, L_1, A_1, I_1)
    Ensure VALID(M, E, V, L_0, A_0, I_0)
    Ensure VALID(M, E, V, L_1, A_1, I_1)
    /*Every operations and comparisons below are done mod M*/
    if L_1 = 0 then
        return L_0, A_0, I_0
    end if
    if L_0 = 0 then
        return L_1, I_1, A_1
    end if
    B_0 ← (A_0 + I_0) · L_1, B_1 ← (I_1 + A_1) · L_0
    C_0 ← (A_0 − I_0) · L_1, C_1 ← (I_1 − A_1) · L_0
    if B0 = B1 then
        if C_0 + C_1 = 0 then
            return 0, 3, −3
        end if
        Sum ← A_0 + I_0, Dif ← A_0 − I_0
        B ← 3 · Sum^2 + E · L_0^2
        C ← 2 · Dif · L_0
        D ← 2 · C · Sum · Dif
        E ← B^2 − 2 · D
        X ← C · E, Y ← B · (D − E) − 2 · C^2 · Dif^2
        return 2 · C^3, X + Y, X − Y
    else
        B ← B_0 − B_1, C ← C_0 − C_1, D = L_0 · L_1
        E ← C^2 · D − B^2 · (B_0 + B_1)
        X ← B · E, Y ← C · (B_0 · B^2 − E) − C_0 · B^3
        return 2 · B^3 · D, X + Y, X − Y
    end if
end function
```

The author has attached a C++ implementation which will get "Time Limit Exceeded" verdict upon submission, but it will always print the correct answer within finite time. You may reuse some part of the implementation on your submission. You can find it on the "Problemset" tab on the domjudge site.