Problem A. A Two Floors Dungeon

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

It was the last day of the summer camp you strayed into the labyrinth on the way to the University of Tokyo, Komaba Campus. The contest has just begun. Your teammates must be waiting impatiently for you. So you have to escape from this labyrinth as soon as possible.

The labyrinth is represented by a grid map. Initially, each grid except for walls and stairs is either on the first floor or on the second floor. Some grids have a switch which can move up or down some of the grids (the grids on the first floor move to the second floor, and the grids on the second floor to the first floor) by the switch.

In each step, you can take one of the following actions:

- Move to an adjacent grid (includes stairs) on the same floor you are now in.
- Move to another floor (if you are in the stairs grid).
- Operate the switch (if you are in a grid with a switch).

Luckily, you have just found a map of the labyrinth for some unknown reason. Let's calculate the minimum step to escape from the labyrinth, and go to the place your teammates are waiting!

Input

The format of the input is as follows.

The first line contains two integers W ($3 \leq W \leq 50$) and H ($3 \leq H \leq 50$). They represent the width and height of the labyrinth, respectively. The following H lines represent the initial state of the labyrinth. Each of M_{ij} is one of the following symbols:

- '`#`' representing a wall,
- '`|`' representing a stairs,
- '`_`' representing a grid which is initially on the first floor,
- '`^`' representing a grid which is initially on the second floor,
- lowercase letter from '`a`' to '`j`' representing a switch the grid has, and the grid is initially on the first floor,
- uppercase letter from '`A`' to '`J`' representing a switch the grid has, and the grid is initially on the second floor,
- '`%`' representing the grid you are initially in (which is initially on the first floor) or
- '`&`' representing the exit of the labyrinth (which is initially on the first floor).

The next line contains one integer S ($0 \leq S \leq 10$), and then the following SH lines represent the information of the switches. Each of MS_{kij} is one of:

- '`#`' if M_{ij} is a '`#`',
- '`|`' if M_{ij} is a '`|`',
- '`*`' if the grid is moved by the switch represented by k-th alphabet letter, or
- '`.`' otherwise.

Note that the grid which contains switch may be moved by operating that switch. In this case, you will move together with the grid. You may assume each of the '`%`' (start) and '`&`' (goal) appears exactly once, that the map is surround by the walls, and that each alphabet in the map is any of the letters from '`A`' (or '`a`') to S-th alphabet letter.

Output

Print the minimum step to reach the goal in one line. If there is no solution, print "-1".

Examples

standard input	standard output
6 6 ###### #_|A%# #B#_|# #^BBa# #B&A## ###### 2 ###### #*|*.# #.#.|# #*.**# #...## ###### ###### #*|*.# #*#.|# #..**# #..*## ######	21

standard input	standard output
8 3 ######## #%||Aa&# ######## 2 ######## #*||*..# ######## ######## #.||*.*# ########	7
3 4 ### #%# #&# ### 0	1
3 5 ### #%# #^# #&# ### 0	-1

Problem B. Billiard

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

You have a billiard table. The playing area of the table is rectangular. This billiard table is special as it has no pockets, and the playing area is completely surrounded with a cushion.

You succeeded in producing a ultra-precision billiards playing robot. When you put some balls on the table, the machine hits one of those balls. The hit ball stops after 10,000 unit distance in total moved.

When a ball collided with the cushion of the table, the ball takes the orbit like mirror reflection. When a ball collided with the corner, the ball bounces back on the course that came.

Your mission is to predict which ball collides first with the ball which the robot hits .

Input

First line of a input file contains an positive integer n, which represents the number of balls on the table ($2 \leq n \leq 11$). The next line contains five integers (w, h, r, v_x, v_y) separated by a single space, where w and h are the width and the length of the playing area of the table respectively ($4 \leq w, h \leq 1{,}000$), r is the radius of the balls ($1 \leq r \leq 100$). The robot hits the ball in the vector (v_x, v_y) direction ($-10{,}000 \leq v_x, v_y \leq 10{,}000$ and $(v_x, v_y) \neq (0, 0)$).

The following n lines give position of balls. Each line consists two integers separated by a single space, (x_i, y_i) means the center position of the i-th ball on the table in the initial state ($r < x_i < w - r$, $r < y_i < h - r$). $(0, 0)$ indicates the position of the north-west corner of the playing area, and (w, h) indicates the position of the south-east corner of the playing area. You can assume that, in the initial state, the balls do not touch each other nor the cushion.

The robot always hits the first ball in the list. You can assume that the given values do not have errors.

Output

Print the index of the ball which first collides with the ball the robot hits. When the hit ball collides with no ball until it stops moving, print "`-1`".

You can assume that no more than one ball collides with the hit ball first, at the same time.

It is also guaranteed that, when r changes by eps ($eps < 10^{-9}$), the ball which collides first and the way of the collision do not change.

Examples

standard input	standard output
3 26 16 1 8 4 10 6 9 2 9 10	3
3 71 363 4 8 0 52 238 25 33 59 288	-1

Problem C. Counting 1's

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

Let $b_i(x)$ be the i-th least significant bit of x, i.e. the i-th least significant digit of x in base 2 ($i \geq 1$). For example, since $6 = (110)_2$, $b_1(6) = 0$, $b_2(6) = 1$, $b_3(6) = 1$, $b_4(6) = 0$, $b_5(6) = 0$, and so on.

Let A and B be integers that satisfy $1 \leq A \leq B \leq 10^{18}$, and k_i be the number of integers x such that $A \leq x \leq B$ and $b_i(x) = 1$.

Your task is to write a program that determines A and B for a given $\{k_i\}$.

Input

The input consists of up to 35 000 test cases. The first line of each case contains an integer n ($1 \leq n \leq 64$). Then n lines follow, each of which contains k_i ($0 \leq k_i \leq 2^{63} - 1$). For all $i > n$, $k_i = 0$.

Output

Print one line for each case.

- If A and B can be uniquely determined, output A and B. Separate the numbers by a single space.
- If there exists more than one possible pair of A and B, output "`Many`" (without quotes).
- Otherwise, i.e. if there exists no possible pair, output "`None`" (without quotes).

Examples

standard input	standard output
3 2 2 1	1 4
4 1 1 1 1	Many
3 4 2 2	None

Problem D. Do use segment tree

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

Given a tree with n ($1 \le n \le 200{,}000$) nodes and a list of q ($1 \le q \le 100{,}000$) queries, process the queries in order and output a value for each output query. The given tree is connected and each node on the tree has a weight w_i ($-10{,}000 \le w_i \le 10{,}000$).

Each query consists of a number t_i ($t_i = 1, 2$), which indicates the type of query , and three numbers a_i, b_i and c_i. Depending on the query type, do one of the following:

- ($t_i = 1$: modify query) Change the weights of all nodes on the shortest path between a_i and b_i (both inclusive) to c_i.
- ($t_i = 2$: output query) First, create a list of weights, which contains weights on the shortest path between a_i and b_i (both inclusive) in order. After that, output the maximum sum of a non-empty continuous subsequence of the weights on the list.

Note that the following conditions are satisfied for all queries: $(1 \le a_i, b_i \le n, -10,000 \le c_i \le 10{,}000)$.

Input

The first line contains two integers n and q. On the second line, there are n integers which indicate w_1, w_2, ... , w_n.

Each of the following $n - 1$ lines consists of two integers s_i and e_i ($1 \le s_i, e_i \le n$), which means that there is a edge between s_i and e_i.

Finally the following q lines give the list of queries, each of which contains four numbers in the format described above. Queries must be processed one by one from top to bottom.

Output

For each output query, output the maximum sum in one line.

Examples

standard input	standard output
3 4 1 2 3 1 2 2 3 2 1 3 0 1 2 2 -4 2 1 3 0 2 2 2 0	6 3 -4
7 5 -8 5 5 5 5 5 5 1 2 2 3 1 4 4 5 1 6 6 7 2 3 7 0 2 5 2 0 2 4 3 0 1 1 1 -1 2 3 7 0	12 10 10 19

Problem E. Eclipse

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

In the year 20XX, mankind is hit by an unprecedented crisis. The power balance between the sun and the moon was broken by the total eclipse of the sun, and the end is looming! To save the world, the secret society called "Sun and Moon" decided to perform the ritual to balance the power of the sun and the moon called "Ritual of Sun and Moon".

The ritual consists of "Ritual of Sun" and "Ritual of Moon". "Ritual of Moon" is performed after "Ritual of Sun". A member of the society is divided into two groups, "Messengers of Sun" and "Messengers of Moon". Each member has some offerings and magic power.

First, the society performs "Ritual of Sun". In the ritual, each member sacrifices an offering. If he can not sacrifice an offering here, he will be killed. After the sacrifice, his magic power is multiplied by the original number of his offerings. Each member must perform the sacrifice just once.

Second, the society performs "Ritual of Moon". In the ritual, each member sacrifices all remaining offerings. After the sacrifice, his magic power is multiplied by x^p. Here x is the number of days from the eclipse (the eclipse is 0th day) and p is the number of his sacrificed offerings. Each member must perform the sacrifice just once.

After two rituals, all "Messengers of Sun" and "Messengers of Moon" give all magic power to the "magical reactor". If the total power of "Messengers of Sun" and the total power of "Messengers of Moon" are equal, the society will succeed in the ritual and save the world.

It is very expensive to perform "Ritual of Sun". It may not be able to perform "Ritual of Sun" because the society is in financial trouble. Please write a program to calculate the minimum number of days from the eclipse in which the society can succeed in "Ritual of Sun and Moon" whether "Ritual of Sun" can be performed or not. The society cannot perform the ritual on the eclipse day (0-th day).

Input

The first line contains an integer N that is the number of members of the society ($0 \le N \le 1{,}000$).

Each of the following N lines contains two integers O_i ($0 \le O_i \le 1{,}000{,}000{,}000$) and P_i ($1 \le |P_i| \le 1{,}000{,}000{,}000{,}000{,}000$). O_i is the number of the i-th member's offerings and $|P_i|$ is the strength of his magic power. If P_i is a positive integer, the i-th member belongs to "Messengers of Sun", otherwise he belongs to "Messengers of Moon".

Output

If there exists the number of days from the eclipse that satisfies the above condition , print the minimum number of days preceded by "`Yes␣`". Otherwise print "`No`".

Of course, the answer must be a positive integer.

Examples

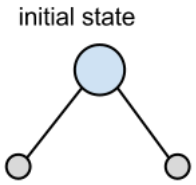
standard input	standard output
9 2 1 1 -1 1 -1 1 -1 1 -1 0 1 0 1 0 1 0 1	Yes 2
2 1 1 0 -1	No

Problem F. Figures and Trees

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

You are a researcher investigating algorithms on binary trees. Binary tree is a data structure composed of branch nodes and leaf nodes. Every branch nodes have left child and right child, and each child is either a branch node or a leaf node. The root of a binary tree is the branch node which has no parent.

You are preparing for your presentation and you have to make a figure of binary trees using a software. You have already made a figure corresponding to a binary tree which is composed of one branch node and two leaf nodes



However, suddenly the function of your software to create a figure was broken. You are very upset because in five hours you have to make the presentation in front of large audience.

You decided to make the figure of the binary trees using only copy function, shrink function and paste function of the presentation tool. That is, you can do only following two types of operations.

- Copy the current figure.
- Paste the copied figure (with shrinking, if needed), putting the root of the pasted figure on a leaf node of the current figure. You *can* paste the copied figure *multiple* times.

Moreover, you decided to make the figure using the minimum number of paste operations because paste operation is very time cosuming. Now, your job is to calculate the minimum possible number of paste operations to produce the target figure.

For example, the answer for the instance of sample 1 (Figure 3) is 3, because you can produce the target figure by following operations and this is minimum.

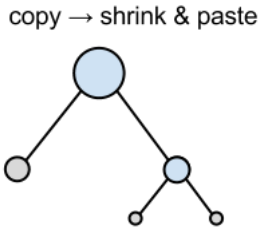


Figure 1

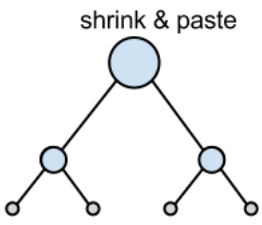


Figure 2

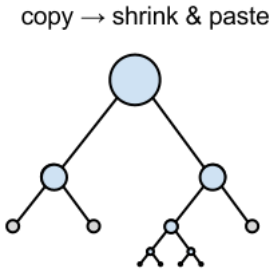


Figure 3

Input

The input is a line which indicates a binary tree. The grammar of the expression is given by the following BNF.

<tree> ::= <leaf> | "(" <tree> <tree> ")"

<leaf> ::= "()"

Every input obeys this syntax.

You can assume that every tree in the input has at least 1 branch node, and that it has no more than 10^5 branch nodes.

Output

A line containing the minimum possible number of paste operations to make the given binary tree.

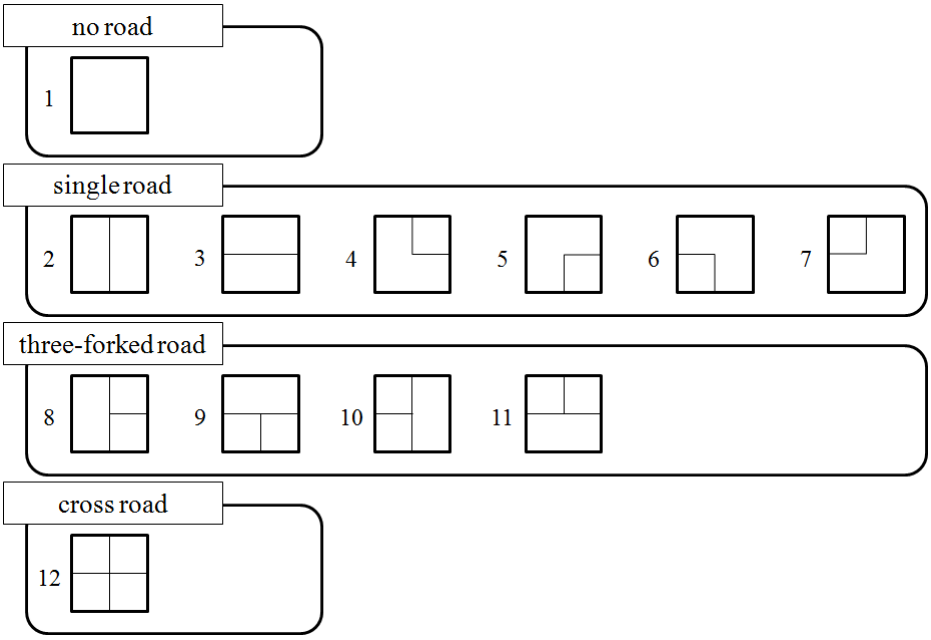
Examples

standard input	standard output
(((())(((())(()()))(())))	3
((((())(()()))(()()))	4
(((()(()))(((()(()))(())))	3
(()())	0

Problem G. Game With Tiles

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

There are twelve types of tiles:



You were asked to fill the table with $R \times C$ cells with these tiles. R is the number of rows and C is the number of columns.

How many arrangements in the table meet the following constraints?

- Each cell has one tile.
- the center of the upper left cell $(1, 1)$ and the center of the lower right cell (C, R) are connected by some roads.

Input

The first line contains two integers R and C ($2 \le R \times C \le 15$). You can safely assume at least one of R and C is greater than 1. The second line contains twelve integers, t_1, t_2, ..., t_{12} ($0 \le t_1 + ... + t_{12} \le 15$). t_i represents the number of the i-th tiles you have.

Output

Output the answer in a line.

Example

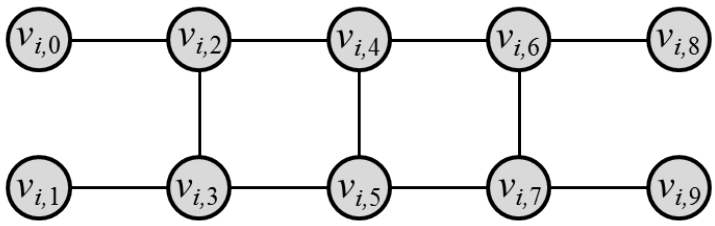
standard input	standard output
3 3 4 2 2 0 0 0 0 0 0 0 0 1	2
3 3 0 1 1 0 0 0 0 0 0 0 0 7	66
3 3 0 0 0 0 0 0 0 0 0 0 0 10	1
2 4 0 0 1 1 1 2 0 1 0 0 1 1	2012
5 2 0 1 1 1 0 1 2 1 2 0 0 1	8512

Problem H. Hashigo Sama

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

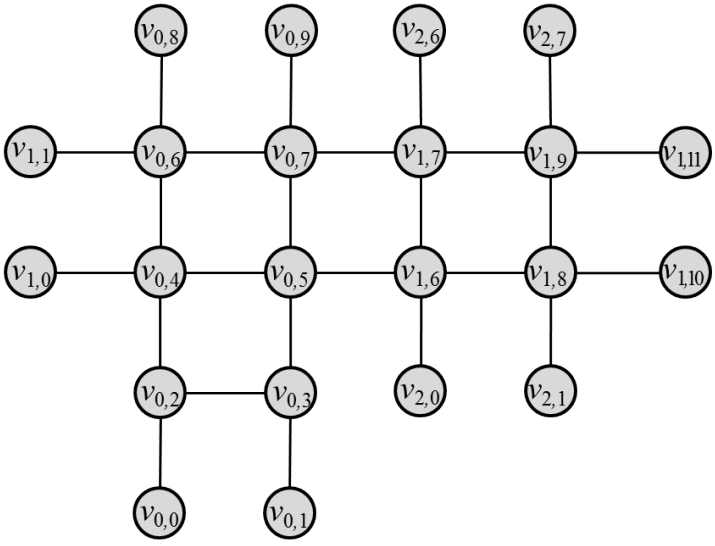
Chelsea is a modern artist. She decided to make her next work with ladders. She wants to combine some ladders and paint some beautiful pattern.

A ladder can be considered as a graph called *hashigo*. There are n *hashigos* numbered from 0 to $n-1$. *Hashigo* i of length l_i has $2l_i+6$ vertices $v_{i,0}, v_{i,1}, ..., v_{i,2l_i+5}$ and has edges between the pair of vertices $(v_{i,j}, v_{i,j+2})$ $(0 \le j \le 2l_i+3)$ and $(v_{i,2j}, v_{i,2j+1})$ $(1 \le j \le l_i+1)$. The figure below is example of a *hashigo* of length 2. This corresponds to the graph given in the first dataset in the sample input.



Two *hashigos* i and j are combined at position p $(0 \le p \le l_i-1)$ and q $(0 \le q \le l_j-1)$ merging each pair of vertices $(v_{i,2p+2}, v_{j,2q+2})$, $(v_{i,2p+3}, v_{j,2q+4})$, $(v_{i,2p+4}, v_{j,2q+3})$ and $(v_{i,2p+5}, v_{j,2q+5})$.

Chelsea performs this operation $n-1$ times to combine the n *hashigos*. After this operation, the graph should be connected and the maximum degree of the graph should not exceed 4. The figure below (flattened for simpilcity) is a example of the graph obtained by combining three *hashigos*. This corresponds to the graph given in the second dataset in the sample input.



(note that while merging two *hashigos* no extra connections with other *hashigos* are created (illusion of those extra connections may appear while looking at this figure)

Now she decided to paint each vertex by black or white with satisfying the following condition:

- The number of vertices in each of the components formed by the connected vertices painted by the same color is less than or equals to k.

She would like to try all the patterns and choose the best. However, the number of painting way can be very huge. Since she is not good at math nor computing, she cannot calculate the number. So please help her with your superb programming skill!

Input

The first line of input file contains two integers n ($1 \leq n \leq 30$) and k ($1 \leq k \leq 8$).

The next line contains n integers l_i ($1 \leq l_i \leq 30$), each denotes the length of *hashigo* i.

The following $n-1$ lines each contains four integers f_i ($0 \leq f_i \leq n-1$), p_i ($0 \leq p_i \leq l_{f_i}-1$), t_i ($0 \leq t_i \leq n-1$), q_i ($0 \leq q_i \leq l_{t_i}-1$). It represents the *hashigo* f_i and the *hashigo* t_i are combined at the position p_i and the position q_i. You may assume that the graph obtained by combining n *hashigos* is connected and the degree of each vertex of the graph does not exceed 4.

Output

Print the number of different colorings modulo 1,000,000,007 in a line.

Examples

standard input	standard output
1 5 2	708
3 7 2 3 1 0 1 1 0 1 2 2 0	1900484
2 8 5 6 0 2 1 2	438404500
2 8 1 1 0 0 1 0	3878
2 2 2 2 0 1 1 0	496
2 3 3 3 0 2 1 1	14246
2 4 3 1 1 0 0 1	9768

Problem I. Interesting Game

Input file:	*standard input*
Output file:	*standard output*
Time limit:	5 seconds
Memory limit:	256 mebibytes

You are playing a solitaire puzzle called "Connect", which uses several letter tiles.

There are $R \times C$ empty cells. For each i $(1 \le i \le R)$, you must put a string s_i $(1 \le |s_i| \le C)$ in the i-th row of the table, without changing the letter order. i.e. choose an integer sequence a such that $1 \le a_1 < a_2 < ... < a_{|s_i|} \le C$ and put the j-th character of the string s_i in the a_j-th column $(1 \le j \le |s_i|)$.

For example, when $C = 8$ and $s_i =$"ICPC", you can put s_i like followings.

```
I_C_P_C_
ICPC____
_IC___PC
```

'_' represents an empty cell.

For each non-empty cell x, you get points equal to the number of adjacent cells which have the same character as x. Two cells are adjacent if they share an edge.

Calculate the maximum total points you can get.

Input

The first line contains two integers R and C $(1 \le R \le 128, 1 \le C \le 16)$.

Then R lines follow, each of which contains s_i $(1 \le |s_i| \le C)$.

All characters in s_i are uppercase English letters.

Output

Output the answer in a line.

Examples

standard input	standard output
2 4 ACM ICPC	2
2 9 PROBLEMF CONNECT	6
4 16 INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST	18

Problem J. Join Usoperanto Community

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

Usoperanto is an artificial spoken language designed and regulated by Usoperanto Academy. The academy is now in study to establish Strict Usoperanto, a variation of the language intended for formal documents.

In Usoperanto, each word can modify at most one other word, and modifiers are always put before modifiees. For example, with a noun *uso* ("truth") modified by an adjective *makka* ("total"), people say *makka uso*, not *uso makka*. On the other hand, there have been no rules about the order among multiple words modifying the same word, so in case *uso* is modified by one more adjective *beta* ("obvious"), people could say both *makka beta uso* and *beta makka uso*.

In Strict Usoperanto, the word order will be restricted according to *modification costs*. Words in a phrase must be arranged so that the total modification cost is minimized. Each pair of a modifier and a modifiee is assigned a cost equal to the number of letters between the two words; the total modification cost is the sum of the costs over all modifier-modifiee pairs in the phrase. For example, the pair of *makka* and *uso* in a phrase *makka beta uso* has the cost of 4 for *beta* (four letters). As the pair of *beta* and *uso* has no words in between and thus the cost of zero, *makka beta uso* has the total modification cost of 4. Similarly *beta makka uso* has the total modification cost of 5. Applying the "minimum total modification cost" rule, *makka beta uso* is preferred to *beta makka uso* in Strict Usoperanto.

Your mission in this problem is to write a program that, given a set of words in a phrase, finds the correct word order in Strict Usoperanto and reports the total modification cost.

Input

The first line contains an integer N ($1 \le N \le 10^6$). N is the number of words in a phrase.

Each of the following N lines contains two integers M_i ($1 \le M_i \le 10$) and L_i ($-1 \le L_i \le N-1$, $L_i \ne i$) describing the i-th word ($0 \le i \le N-1$). M_i is the number of the letters in the word. L_i specifies the modification: $L_i = -1$ indicates it does not modify any word; otherwise it modifies the L_i-th word.

Note the first sample input below can be interpreted as the *uso-beta-makka* case.

Output

Print the total modification cost.

Examples

standard input	standard output
3 3 -1 4 0 5 0	4
3 10 -1 10 0 10 1	0
4 1 -1 1 0 1 1 1 0	1

Problem K. King and Pipes

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

King of Byteland is planning to repair the water pipe at a certain point in the water pipe network in his capital. The network consists of water pipe segments, stop valves and source point. A water pipe is represented by a segment on a 2D-plane and intersected pair of water pipe segments are connected at the intersection point. A stop valve, which prevents from water flowing into the repairing point while repairing, is represented by a point on some water pipe segment. In the network, just one source point exists and water is supplied to the network from this point.

Of course, while repairing, we have to stop water supply in some areas, but, in order to reduce the risk of riots, the length of water pipes stopping water supply must be minimized. What you have to do is to write a program to minimize the length of water pipes needed to stop water supply when the coordinates of end points of water pipe segments, stop valves, source point and repairing point are given.

Input

The first line of the input contains two integers, N ($1 \leq N \leq 300$) and M ($0 \leq M \leq 1{,}000$) that indicate the number of water pipe segments and stop valves. The following N lines describe the end points of water pipe segments. The i-th line contains four integers, x_{si}, y_{si}, x_{di} and y_{di} that indicate the pair of coordinates of end points of i-th water pipe segment. The following M lines describe the points of stop valves. The i-th line contains two integers, x_{vi} and y_{vi} that indicate the coordinate of end points of i-th stop valve. The following line contains two integers, x_b and y_b that indicate the coordinate of the source point. The last line contains two integers, x_c and y_c that indicate the coordinate of the repairing point.

You may assume that any absolute values of coordinate integers are less than 1,000 (inclusive.) You may also assume each of the stop valves, the source point and the repairing point is always on one of water pipe segments and that that each pair among the stop valves, the source point and the repairing point are different. And, there is not more than one intersection between each pair of water pipe segments. Finally, the water pipe network is connected, that is, all the water pipes are received water supply initially.

Output

Print the minimal length of water pipes needed to stop water supply in a line. The absolute or relative error should be less than or 10^{-6}. When you cannot stop water supply to the repairing point even though you close all stop valves, print "`-1`" in a line.

Examples

standard input	standard output
1 2 0 0 10 0 1 0 9 0 0 0 5 0	9.0
5 3 0 4 2 4 0 2 2 2 0 0 2 0 0 0 0 4 2 0 2 4 0 2 1 0 2 2 1 4 2 1	3.0
2 1 0 0 0 4 0 2 2 2 1 2 0 1 0 3	-1

Problem L. Let's Move On Dice

Input file:	*standard input*
Output file:	*standard output*
Time limit:	5 seconds
Memory limit:	256 mebibytes

There is a cube on a rectangle map with H-rows and W-columns grid. Two special squares are marked on the map, a start and a goal. Initially, the cube is on the start square. Let's repeat to roll it and take it to the goal square. To roll the cube means to select one of four edges which faces the map, and push the cube down without detaching the edge from the map. That is, there are four directions you can move the dice toward.

Also, directions where we can roll the cube are limited depending on each square. An instruction is written in each square and represented by a single character as follows:

'+' all

'|' only vertical

'-' only horizontal

'<' only to left

'>' only to right

'^' only to up

'v' only to down

'.' none

However, regardless of instructions, it is not allowed to roll the dice to outside of the map.

Also, on each face of the cube, a string is written. Let's output the string which concatenates strings written on the top face seen during the rolls from the start square to the goal square. Since there may be multiple paths that take the cube to the goal square, choose the minimal string in ascending lexicographic order.

Please note that there are cases where no path exists from the start to the goal, or the cases you can make the lexicographically minimal string infinitely longer.

Input

The first line of the input contains two integers H ($1 \leq H \leq 12$) and W ($1 \leq W \leq 12$), which indicate the number of rows and columns of the map respectively. The following W lines describe the map. The j-th character of the i-th line indicates the instruction of the square, which is placed on i-th row (from the top) and j-th column (from the left).

Then the following six lines describe the strings on each face of the cube. All of these strings are not empty and shorter than 12 characters (inclusive). In addition, they only consist of uppercase alphabets or digits. The faces where the strings are written are given as figure 1. Initially, the cube is placed on the start square in a direction as the face No. 1 is facing top and the upper direction of face No. 1 faces toward the top row of the map.

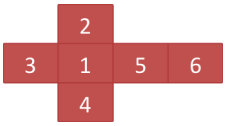


Figure 1: a net of a cube

The last two lines contain two integers each that indicate the row number and column number of the start square and the goal square in this order. You can assume that the start square and the goal square are always different.

Output

Print the lexicographically minimal string in a line. If there is no path, print "no" in a line. If you can make the lexicographically minimal string infinitely longer, print "infinite" in a line.

Examples

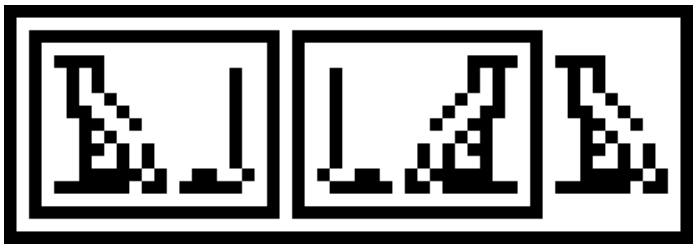
standard input	standard output
1 3 +++ 6 5 4 3 2 1 1 3 1 1	621
1 3 +++ 1 2 3 4 5 6 1 3 1 1	infinite

standard input	standard output
1 3 ... 1 2 3 4 5 6 1 3 1 1	no
3 3 ->| ..v .^< JAG 2012 SUMMER HOGE HOGE CAMP 1 1 2 2	JAGSUMMERCAMP2012JAGSUMMER2012

Problem M. Monolith

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

One day in a forest, Alice found an old monolith.



She investigated the monolith, and found this was a sentence written in an old language. A sentence consists of glyphs and rectangles which surrounds them. For the above example, there are following glyphs.

Notice that some glyphs are flipped horizontally in a sentence.

She decided to transliterate it using ASCII letters assigning an alphabet to each glyph, and '`[`' and '`]`' to rectangles. If a sentence contains a flipped glyph, she transliterates it from right to left. For example, she could assign '`a`' and '`b`' to the above glyphs respectively. Then the above sentence would be transliterated into "`[[ab][ab]a]`".

After examining the sentence, Alice figured out that the sentence was organized by the following structure:

- A sentence ⟨seq⟩ is a sequence consists of zero or more ⟨term⟩s.
- A term ⟨term⟩ is either a glyph or a ⟨box⟩. A glyph may be flipped.
- ⟨box⟩ is a rectangle surrounding a ⟨seq⟩. The height of a box is larger than any glyphs inside.

Now, the sentence written on the monolith is a nonempty ⟨seq⟩. Each term in the sequence fits in a rectangular bounding box, although the boxes are not explicitly indicated for glyphs. Those bounding boxes for adjacent terms have no overlap to each other. Alice formalized the transliteration rules as follows.

Let f be the transliterate function.

Each sequence $s = t_1 t_2 \cdots t_m$ is written either from left to right, or from right to left. However, please note here t_1 corresponds to the leftmost term in the sentence, t_2 to the 2nd term from the left, and so on.

Let's define $\bar{g}$ to be the flipped glyph g. A sequence must have been written from right to left, when a sequence contains one or more single-glyph term which is unreadable without flipping, i.e. there exists an integer i where t_i is a single glyph g, and g is not in the glyph dictionary given separately whereas $\bar{g}$ is. In such cases $f(s)$ is defined to be $f(t_m)f(t_{m-1})\cdots f(t_1)$, otherwise $f(s) = f(t_1)f(t_2)\cdots f(t_m)$. It is guaranteed that all the glyphs in the sequence are flipped if there is one or more glyph which is unreadable without flipping.

If the term t_i is a box enclosing a sequence s', $f(t_i) =$ '`[`'$f(s')$'`]`'. If the term t_i is a glyph g, $f(t_i)$ is mapped to an alphabet letter corresponding to the glyph g, or $\bar{g}$ if the sequence containing g is written from right to left.

Please make a program to transliterate the sentences on the monoliths to help Alice.

Input

Each dataset has the following format.

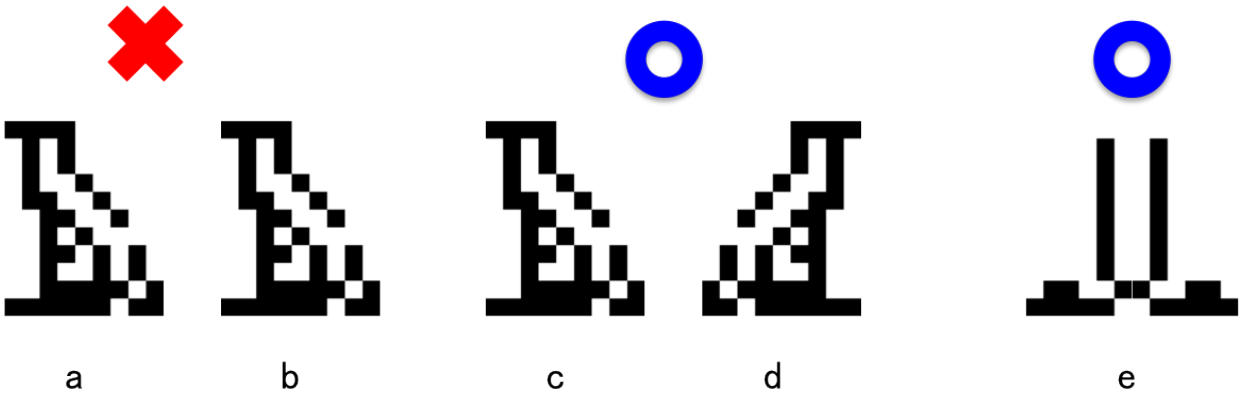
$n\ m$
$glyph_1$
...
$glyph_n$
$string_1$
...
$string_m$

n ($1 \leq n \leq 26$) is the number of glyphs and m ($1 \leq m \leq 10$) is the number of monoliths. $glyph_i$ is given by the following format.

$c\ h\ w$
$b_{11}...b_{1w}$
...
$b_{h1}...b_{hw}$

c is a lower-case alphabet that Alice assigned to the glyph. h and w ($1 \leq h \leq 15$, $1 \leq w \leq 15$) specify the height and width of the bitmap of the glyph, respectively. The matrix b indicates the bitmap. A white cell is represented by '.' and a black cell is represented by '*'.

You can assume that all the glyphs are assigned distinct characters. You can assume that every column of the bitmap contains at least one black cell, and the first and last row of the bitmap contains at least one black cell. Every bitmap is different to each other, but it is possible that a flipped bitmap is same to another bitmap. Moreover there may be symmetric bitmaps.



$string_i$ is given by the following format.

$h\ w$
$b_{11}...b_{1w}$
...
$b_{h1}...b_{hw}$

h and w ($1 \leq h \leq 100$, $1 \leq w \leq 1000$) is the height and width of the bitmap of the sequence. Similarly to the glyph dictionary, b indicates the bitmap where A white cell is represented by '.' and a black cell by '*'.

There is no noise: every black cell in the bitmap belongs to either one glyph or one rectangle box. The height of a rectangle is at least 3 and more than the maximum height of the tallest glyph. The width of a rectangle is at least 3.

A box must have a margin of 1 pixel around the edge of it.

You can assume the glyphs are never arranged vertically. Moreover, you can assume that if two rectangles, or a rectangle and a glyph, are in the same column, then one of them contains the other. For all bounding boxes of glyphs and black cells of rectangles, there is at least one white cell between every two of them. You can assume at least one cell in the bitmap is black.

Output

For each monolith, output the transliterated sentence in one line.

Examples

standard input
3 1 a 2 2 .* ** b 2 1 * * c 3 3 *** *.* *** 11 16 **************** *..............* *....*********.* *....*.......*.* *.*..*.***...*.* *.**.*.*.*.*.*.* *....*.***.*.*.* *....*.......*.* *....*********.* *..............* ****************
standard output
[[cb]a]

standard input

```
2 1
a 11 9
****.....
.*.*.....
.*.*.....
.*..*....
.**..*...
..**..*..
..*.*....
..**.*.*.
..*..*.*.
..*****.*
******.**
b 10 6
....*.
....*.
....*.
....*.
....*.
....*.
....*.
....*.
.**..*
*****.
19 55
*********************************************************
*.......................................................*
*.*******************.*******************.............*
*.*.................*.*.................*.............*
*.*.****............*.*............****.*.****........*
*.*..*.*.........*..*.*..*.........*.*..*..*.*........*
*.*..*.*.........*..*.*..*.........*.*..*..*.*........*
*.*..*..*........*..*.*..*........*..*..*..*..*.....*
*.*..**..*.......*..*.*..*.......*..**..*..**..*....*
*.*...**..*......*..*.*..*......*..**...*...**..*...*
*.*...*.*........*..*.*..*........*.*...*...*.*.....*
*.*...**.*.*.....*..*.*..*.....*.*.**...*...**.*.*..*
*.*...*..*.*.....*..*.*..*.....*.*..*...*...*..*.*..*
*.*...*****.*..**..*.*.*..**..*.*****...*...*****.*.*
*.*.******.**.*****..*.*..*****.**.******.*.******.**.*
*.*.................*.*.................*.............*
*.*******************.*******************.............*
*.......................................................*
*********************************************************
```

standard output

```
[[ab][ab]a]
```

standard input
2 3 a 2 3 .*. *** b 3 3 .*. *** .*. 2 3 .*. *** 4 3 *** *.* *.* *** 9 13 ************. *..........*. *.......*..*. *...*..***.*. *..***.....*. *...*......*. *..........*. ************. ....
standard output
a [] [ba]