Problem A. Thread Tree

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

Nathan O. Davis has been running an electronic bulletin board system named JAG-channel. He is now having hard time to add a new feature there — threaded view.

Like many other bulletin board systems, JAG-channel is thread-based. Here a thread (also called a topic) refers to a single conversation with a collection of posts. Each post can be an opening post, which initiates a new thread, or a reply to a previous post in an existing thread.

Threaded view is a tree-like view that reflects the logical reply structure among the posts: each post forms a node of the tree and contains its replies as its subnodes in the chronological order (i.e. older replies precede newer ones). Note that a post along with its direct and indirect replies forms a subtree as a whole.

Let us take an example. Suppose: a user made an opening post with a message hoge; another user replied to it with fuga; yet another user also replied to the opening post with piyo; someone else replied to the second post (i.e. fuga") with foobar; and the fifth user replied to the same post with jagjag.

For easier implementation, Nathan is thinking of a simpler format: the depth of each post from the opening post is represented by dots. Each reply gets one more dot than its parent post. The tree of the above thread would then look like:

```
hoge
.fuga
..foobar
..jagjag
.piyo
```

Your task in this problem is to help Nathan by writing a program that prints a tree in the Nathan's format for the given posts in a single thread.

Input

Input contains a single dataset in the following format:

The first line contains an integer n ($1 \leqslant n \leqslant 1000$), which is the number of posts in the thread. Then $2n$ lines follow. Each post is represented by two lines: the first line contains an integer k_i ($k_1 = 0$, $1 \leqslant k_i < i$ for $2 \leqslant i \leqslant n$) and indicates the i-th post is a reply to the k_i-th post; the second line contains a string M_i and represents the message of the i-th post. k_1 is always 0, which means the first post is not replying to any other post, i.e. it is an opening post.

Each message contains 1 to 50 characters, consisting of uppercase, lowercase, and numeric letters.

Output

Print the given n messages as specified in the problem statement.

Examples

standard input	standard output
1 0 icpc	icpc
8 0 jagjag 1 hogehoge 1 buhihi 2 fugafuga 4 ponyoponyo 5 evaeva 4 nowawa 5 pokemon	jagjag .hogehoge ..fugafuga ...ponyoponyo ....evaeva ....pokemon ...nowawa .buhihi
6 0 nakachan 1 fan 2 yamemasu 3 nennryou2 4 dannyaku4 5 kouzai11	nakachan .fan ..yamemasu ...nennryou2 ....dannyaku4 .....kouzai11

Problem B. Trodden Cable

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

Problem Statement Nathan O. Davis is running a company. His company owns a web service which has a lot of users. So his office is full of servers, routers and messy LAN cables.

He is now very puzzling over the messy cables, because they are causing many kinds of problems. For example, staff working at the company often trip over a cable. No damage if the cable is disconnected. It's just lucky. Computers may fall down and get broken if the cable is connected. He is about to introduce a new computer and a new cable. He wants to minimize staff's steps over the new cable.

His office is laid-out in a two-dimensional grid with $H \times W$ cells. The new cable should weave along edges of the cells. Each end of the cable is at a corner of a cell. The grid is expressed in zero-origin coordinates, where the upper left corner is $(0, 0)$.

Each staff starts his/her work from inside a certain cell and walks in the office along the grid in a fixed repeated pattern every day. A walking pattern is described by a string of four characters '`U`', '`D`', '`L`' and '`R`'. '`U`' means moving up, '`D`' means moving down, '`R`' means moving to the right, and '`L`' means moving to the left. For example, "`UULLDDRR`" means moving up, up, left, left, down, down, right and right in order. The staff repeats the pattern fixed T times. Note that the staff stays in the cell if the staff is going out of the grid through the wall.

You have the moving patterns of all staff and the positions of both ends of the new cable. Your job is to find an optimal placement of the new cable, which minimizes the total number his staff would step over the cable.

Input

The first line of the input contains three integers which represent the dimension of the office W, H ($1 \leqslant W, H \leqslant 500$), and the number of staff N ($1 \leqslant N \leqslant 1000$), respectively. The next line contains two $x - y$ pairs ($0 \leqslant x \leqslant W$, $0 \leqslant y \leqslant H$), which mean the position of two endpoints of a LAN cable to be connected. These values represents the coordinates of the cells to which the cable is plugged in its top-left corner. Exceptionally, $x = W$ means the right edge of the rightmost cell, and $y = H$ means the bottom edge of the bottommost cell.

Following lines describe staff's initial positions and their moving patterns. The first line includes an $x - y$ pair ($0 \leqslant x < W$, $0 \leqslant y < H$), which represents the coordinate of a staff's initial cell. The next line has an integer T ($1 \leqslant T \leqslant 100$) and a string which consists of U, D, L and R, whose meaning is described as above. The length of a pattern string is greater than or equal to 1, and no more than 1,000. These two lines are repeated N times.

Output

Output the minimum number of times his staff step over the cable in a single line.

Examples

standard input	standard output
3 3 1 1 1 3 3 0 0 1 RRDDLLUU	1
3 3 1 0 0 3 3 0 0 1 RRDDLLUU	0
3 3 1 1 1 3 3 0 0 10 RRDDLLUU	10
3 3 4 1 1 3 3 0 0 10 R 2 0 10 D 2 2 10 L 0 2 10 U	1

Problem C. Fox Observation

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

Ievan Ritola is a researcher of behavioral ecology. Her group visited a forest to analyze an ecological system of some kinds of foxes.

The forest can be expressed as a two-dimensional plane. With her previous research, foxes in the forest are known to live at lattice points. Here, lattice points are the points whose x and y coordinates are both integers. Two or more foxes might live at the same point.

To observe the biology of these foxes, they decided to put a pair of sensors in the forest. The sensors can be put at lattice points. Then, they will be able to observe all foxes inside the bounding rectangle (including the boundary) where the sensors are catty-corner to each other. The sensors cannot be placed at the points that have the same x or y coordinate; in other words the rectangle must have non-zero area.

The more foxes can be observed, the more data can be collected; on the other hand, monitoring a large area consumes a large amount of energy. So they want to maximize the value given by $N'/(|x_1-x_2|\times|y_1-y_2|)$, where N' is the number of foxes observed and (x_1, y_1) and (x_2, y_2) are the positions of the two sensors.

Let's help her observe cute foxes!

Input

The first line contains a single integer N ($1 \leqslant N \leqslant 10^5$) indicating the number of the fox lairs in the forest. Each of the next N lines contains three integers x_i, y_i ($|x_i|, |y_i| \leqslant 10^9$) and w_i ($1 \leqslant w_i \leqslant 10^4$), which represent there are w_i foxes at the point (x_i, y_i). It is guaranteed that all points are mutually different.

Output

Output the minimized maximized value as a fraction a/b where a and b are integers representing the numerator and the denominator respectively. There should be exactly one space before and after the slash. The fraction should be written in the simplest form, that is, a and b must not have a common integer divisor greater than one.

If the value becomes an integer, print a fraction with the denominator of one (e.g. "`5 / 1`" to represent 5). This implies zero should be printed as "`0 / 1`" (without quotes).

Examples

standard input	standard output
2 1 1 2 2 2 3	5 / 1

Problem D. Removing Magical Tiles

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

Ayimok is a wizard.

His daily task is to arrange magical tiles in a line, and then cast a magic spell.

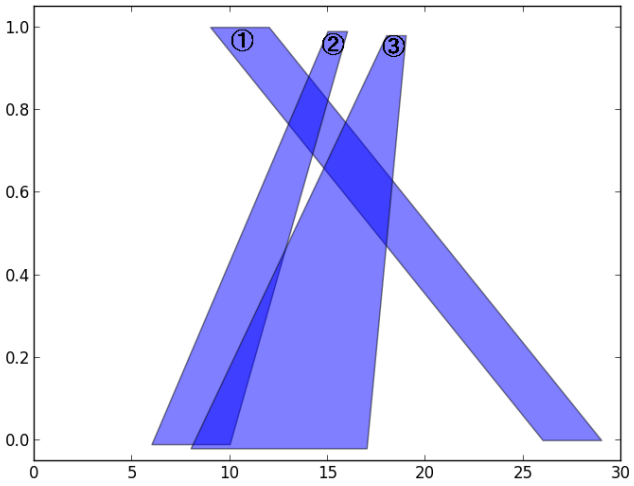
Magical tiles and their arrangement follow the rules below:

- Each magical tile has the shape of a trapezoid with a height of 1.
- Some magical tiles may overlap with other magical tiles.
- Magical tiles are arranged on the 2D coordinate system.
- Magical tiles' upper edge lay on a horizontal line, where its y coordinate is 1.
- Magical tiles' lower edge lay on a horizontal line, where its y coordinate is 0.

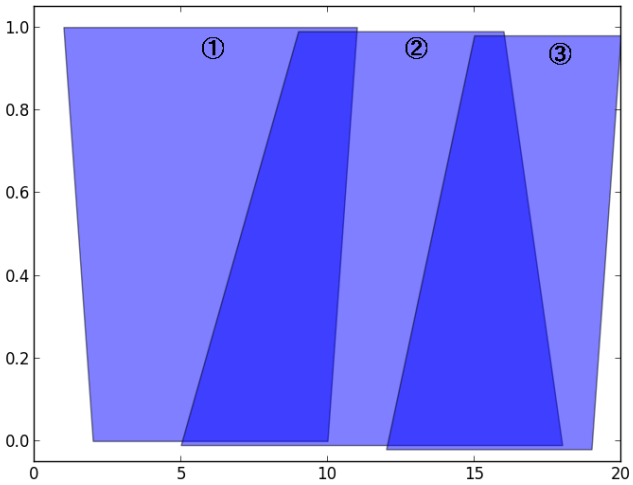
He has to remove these magical tiles away after he finishes casting a magic spell. One day, he found that removing a number of magical tiles is so tiresome, so he decided to simplify its process.

Now, he has learned "Magnetization Spell" to remove magical tiles efficiently. The spell removes a set of magical tiles at once. Any pair of two magical tiles in the set must overlap with each other.

For example, in figure below, he can cast Magnetization Spell on all of three magical tiles to remove them away at once, since they all overlap with each other. (Note that the heights of magical tiles are intentionally changed for easier understandings.)

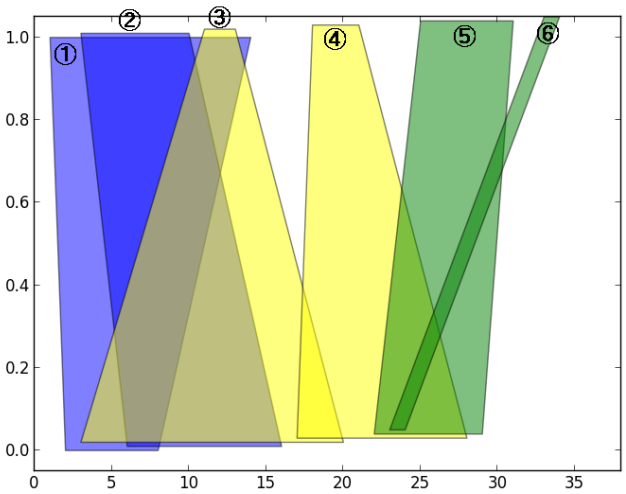


However, in second figure, he can not cast Magnetization Spell on all of three magical tiles at once, since tile 1 and tile 3 are not overlapped.

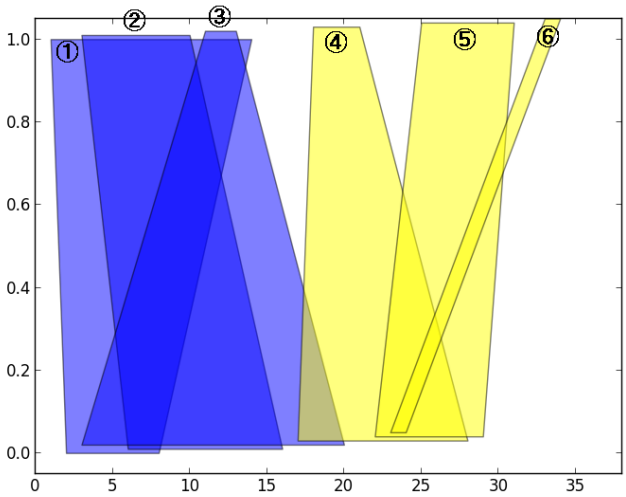


He wants to remove all the magical tiles with the minimum number of Magnetization Spells, because casting the spell requires magic power exhaustively.

Let's consider next case (third picture). He can remove all of magical tiles by casting Magnetization Spell three times; first spell for tile 1 and 2, second spell for tile 3 and 4, and third spell for tile 5 and 6.



Actually, he can remove all of the magical tiles with casting Magnetization Spell just twice; first spell for tile 1, 2 and 3, second spell for tile 4, 5 and 6. And it is the minimum number of required spells for removing all magical tiles (look at picture below):



Given a set of magical tiles, your task is to create a program which outputs the minimum number of casting Magnetization Spell to remove all of these magical tiles away.

There might be a magical tile which does not overlap with other tiles, however, he still needs to cast Magnetization Spell to remove it.

Input

The first line contains an integer N, which means the number of magical tiles.

Then, N lines which contain the information of magical tiles follow.

The (i+1)-th line includes four integers $xLower_{i1}$, $xLower_{i2}$, $xUpper_{i1}$, and $xUpper_{i2}$, which means i-th magical tile's corner points are located at $(xLower_{i1}, 0)$, $(xLower_{i2}, 0)$, $(xUpper_{i1}, 1)$, $(xUpper_{i2}, 1)$.

You can assume that no two magical tiles will share their corner points. That is, all $xLower_{ij}$ are distinct, and all $xUpper_{ij}$ are also distinct. The input follows the following constraints: $1 \leqslant N \leqslant 10^5$, $1 \leqslant xLower_{i1} < xLower_{i2} \leqslant 10^9$, $1 \leqslant xUpper_{i1} < xUpper_{i2} \leqslant 10^9$. All $xLower_{ij}$ are distinct. All $xUpper_{ij}$ are distinct.

Output

Your program must output the required minimum number of casting Magnetization Spell to remove all the magical tiles.

Examples

standard input	standard output
3 26 29 9 12 6 10 15 16 8 17 18 19	1
3 2 10 1 11 5 18 9 16 12 19 15 20	2
6 2 8 1 14 6 16 3 10 3 20 11 13 17 28 18 21 22 29 25 31 23 24 33 34	2

Problem E. Elementary arithmetic

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 Mebibytes

Fox arithmetic is very elementary. There are just two operations: "`WITH`" and "`WITHOUT`" which mean addition and subtraction respectively.

Fox mathematicians did not yet invent brackets, so all expressions are calculated from right to left. For example, the expression "`3 WITH 5 WITHOUT 4 WITHOUT 7`" is calculated as $(3 + (5 - (4 - 7)))$.

Scientists of Forest Institute of Fox Arithmetic need a program to evaluate such expressions.

Input

First line of input file contains a single integer N. Following $2 \times N + 1$ lines describe the expression. Odd lines contain integer operands, even lines contain operations ($1 \leqslant N \leqslant 10^3$). All operands are in range from 0 to 10^3.

Output

Output file must contain a single integer — calculation result.

Examples

standard input	standard output
1 17 WITH 18	35
3 10 WITHOUT 5 WITH 3 WITH 15	-13

Problem F. Graph Automata Player

Input file:	`standard input`
Output file:	`standard output`
Time limit:	10 seconds
Memory limit:	256 mebibytes

You and your grandma are playing with graph automata, which is generalization of cell automata.

A graph automaton is expressed by a graph. The vertices of the graph have a time-dependent value, which can be either 0 or 1. There is no more than one edge between any of two vertices, but self-loops might exist.

The values of vertices change regularly according to the following rule: At time $t + 1$, the value of vertex i will be 1 if and only if there are an odd number of edges from the vertex i to a vertex which has the value 1 at time t; otherwise 0.

Now, your forgetful grandma forgot the past states of the automaton. Your task is to write a program which recovers the past states from the current time and states — time machines are way too expensive. There may be multiple candidates or no consistent states. For such cases, you must print an appropriate error message.

Input

The first line contains one integer N ($2 \leqslant N \leqslant 300$). N indicates the number of vertices. The following N lines indicate the adjacent matrix of the graph. When (i, j)-element is 1, there is an edge from vertex i to vertex j. Otherwise, there is no edge. The following N lines indicate the value vector of the vertices. The i-th element indicates the value of vertex i at time 0. Each element of the matrix and the vector can be 0 or 1. The last line contains one integer T ($1 \leqslant T \leqslant 10^8$). $-T$ is the time your grandma wants to know the status at.

Output

Print the value vector at time $-T$. Each value must be separated with one white space. If there is no consistent value vectors, you should print "`none`" in one line. Or if there are multiple candidates and the solution is not unique (i.e. the solution is not unique), you should print "`ambiguous`" in one line.

Examples

standard input	standard output
2 1 1 0 1 1 1 1	0 1
2 0 1 0 0 1 0 1	ambiguous
2 0 1 0 0 1 0 2	none

Problem G. Spotlight Movement

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

Ciel, an idol whose appearance and behavior are similar to a fox, is participating in a rehearsal for the live concert which takes place in a few days. To become a top idol, a large amount of effort is required!

The live stage can be expressed as a two-dimensional surface. The live stage has N spotlights to illuminate the stage. The i-th spotlight casts a light in the range of a circle with radius r_i. The center of the light cast by the i-th spotlight moves along an orbital path R_i. R_i is described as a closed polygon, though R_i might contain self-intersections. The spotlight begins to move from the first vertex of R_i. All of the orbital period of each spotlight are the same. Each spotlight moves at a constant speed, and all of them return to the starting point at the same time.

In the rehearsal, Ciel has to move from the starting point to the ending point indicated on the live stage. To achieve her goal, it is not allowed to fall out from the area illuminated with the spotlight. But, she doesn't have to be illuminated with the spotlight as long as she is standing on the starting point. You may assume that she can move fast enough. Answer if it is possible for her to move to the ending point.

Input

All inputs are integer. All coordinate information does not exceed 10^4 by absolute value.

N ($1 \leqslant N \leqslant 100$) indicates the number of spotlights. (sx, sy) and (ex, ey) indicate the starting point and the ending point of Ciel's path, respectively. The following N lines denote the information of each spotlight. r_i ($1 \leqslant r_i \leqslant 100$) indicates the radius of the spotlight. K_i ($2 \leqslant K_i \leqslant 10$) indicates the number of vertices in the orbital path. Then, K_i vertices are given. Two consecutive vertices on the same orbital path are located at different places. The spotlight moves from the first point (x_{i1}, y_{i1}) to the second point (x_{i2}, y_{i2}), then moves to the third point (x_{i3}, y_{i3}), and so on. After moving to the K_i-th point (x_{iK_i}, y_{iK_i}), the spotlight returns to the first point (x_{i1}, y_{i1}) and repeats again its movement.

Let d_{ij} be the closest distance between two central points of spotlight i and spotlight j. d_{ij} satisfies either of the following: $d_{ij} > r_i + r_j + 0.000001$ or $d_{ij} < r_i + r_j - 0.000001$.

Furthermore, let d_i be the closest distance between a central point of spotlight i and either the starting point or the ending point. d_i satisfies either of the following: $d_i > r_i + 0.000001$ or $d_i < r_i - 0.000001$.

Output

If it is possible for Ciel to move the ending point without falling out from the illuminated area, output "`Yes`" in one line. Otherwise, output "`No`".

Examples

standard input	standard output
2 1 1 9 -1 2 2 1 1 -9 1 1 2 -1 -1 9 -1	Yes
2 1 1 9 -1 2 2 1 1 -9 1 1 2 9 -1 -1 -1	No
2 11 6 -9 1 6 4 10 5 10 0 5 0 5 5 5 4 -10 0 -10 5 -5 5 -5 0	Yes

Problem H. Gravity Point

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

You are practicing a juggle that involves in a number of square tiles. They all look the same in their size, but actually there are three different kinds of the tiles, A, B and X. The kind of a tile can be distinguishable by its mass. All the tiles of the same kind have exactly the same mass. The mass of type A tiles is within the range $[mA1, mA2]$. The mass of type B tiles is similarly within the range $[mB1, mB2]$. You don't know the exact numbers for type A and B. The mass of type X tiles is exactly mX.

You have just got one big object consists of the tiles. The tiles are arranged in an $H \times W$ grid. All the adjacent tiles are glued together on the edges. Some cells in the $H \times W$ grid can be empty.

You wanted to balance the object on a pole. You started to wonder the center of gravity of the object, then. You cannot put the object on a pole if the center of gravity is on an empty cell. The center of gravity of each single square tile is at the center of the square. The center of gravity of an object which combines two objects with masses m_1 and m_2 is the point dividing the line segment between those centers in the ratio $m_2 : m_1$ internally.

Your task is to write a program that computes the probability that the center of gravity of the big object is actually on the object, not on a hole in the object. Although the exact mass is unknown for tile A and B, the probability follows the continuous uniform distribution within the range mentioned above. You can assume the distribution is independent between A and B.

Input

The first line of the input contains two integers H and W ($1 \leqslant H, W \leqslant 50$) separated by a space, where H and W are the numbers of rows and columns of given matrix.

The second line of the input contains five integers $mA1$, $mA2$, $mB1$, $mB2$ and mX ($1 \leqslant mA1 < mA2 \leqslant 100$, $1 \leqslant mB1 < mB2 \leqslant 100$ and $1 \leqslant mX \leqslant 100$) separated by a space.

The following H lines, each consisting of W characters, denote given matrix. In those H lines, '`A`', '`B`' and '`X`' denote a piece of type A, type B and type X respectively, and '`.`' denotes empty cell to place no piece. There are no other characters in those H lines.

Of the cell at i-th row in j-th column, the coordinate of the left top corner is (i, j), and the coordinate of the right bottom corner is $(i + 1, j + 1)$.

You may assume that given matrix has at least one '`A`', '`B`' and '`X`' each and all pieces are connected on at least one edge. Also, you may assume that the probability that the x-coordinate of the center of gravity of the object is an integer is equal to zero and the probability that the y-coordinate of the center of gravity of the object is an integer is equal to zero.

Output

Print the probability that the center of gravity of the object is on the object. The output should not contain an error greater than 10^{-8}.

Examples

standard input	standard output
3 3 2 4 1 2 1 XAX B.B XAX	0.0000000000000
4 2 1 100 1 100 50 AX XB BA XB	1.0
2 3 1 2 3 4 2 X.B AXX	0.500
10 10 1 100 1 100 1 AXXXXXXXXX X........X X........X X..XXXXXXX X........X XXXXXX...X X........X X......X.X X......X.X XXXXXXXXXB	0.4930639462354
25 38 42 99 40 89 3 ...........X.............X......... ...........XXX.........XXXX......... ...........XXXXX.......XXXX.......... ............XXXXXXXXXXXXXX.......... ............XXXXXXXXXXXXXX.......... ............XXXXXXXXXXXXX........... ............XXXXXXXXXXXXX........... .............XXXXXXXXXXXXX........... ............XXXXXXXXXXXXXX.......... ...........XXXXXXXXXXXXXXXX......... .......X...XXXXXXXXXXXXXXXX...X...... .......XX.XXXXXXXXXXXXXXXXX..XX...... ........XXXXXXXXXXXXXXXXXXX.XX....... ..........XXXXXXXXXXXXXXXXXX......... .........XXXXX..XXXXXXXX..XXXXX....... .......XXXXXX....XXXXX....XXXXXX...... ......XXXXXXXX...XXXXX..XXXXXXXX.X.... ....XXXXXXX..X...XXXXX..X..XXXXXXXX... ..BBBXXXXXX.....XXXXXXX......XXXAAAA.. ...BBBXXXX......XXXXXXX........AAA.... ..BBBB.........XXXXXXXXX........AAA... ....XXXXXXXXX............. ....XXXXXXXXXX............. ....XXXXXXXXXX............. ....XXXXXXXX............. ....XXXXXXXX.............	0.9418222212582

Problem I. Multi Path Story

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

You are a programmer who loves pretty girl games (a sub-genre of dating simulation games). A game, which is titled «Floatable Heart» and was released last Friday, has arrived at your home just now. This game has multiple stories. When you complete all of those stories, you can get a special figure of the main heroine, Megumi. So, you want to hurry to play the game! But, let's calm down a bit and think how to complete all of the stories in the shortest time first.

In fact, you have the special skill that allows you to know the structure of branching points of games. By using the skill, you have found out that there are n branching points in this game and i-th branching point has k_i choices. This game is so complicated that multiple routes get to i-th branching point probably. You also noticed that it takes c_{ij} minutes to proceed from i-th branching point to another branching point (or an ending), if you select j-th choice of i-th branching point. Of course you need to select all the choices on all the branching points and read stories between a branching point and another branching point (or an ending) to complete all of those stories. In addition, you can assume it only takes negligible time to return to the beginning of the game («reset») and to play from the beginning to the first branching point.

The game's manual says that this game has an additional feature called «Quick Save», and the feature allows you to record the point where you are currently playing and return there at any time later. However, this feature is not working for some bug. Thus you have to restart from the first branching point every time, if you reach an ending or quit the game on the way. Any patch to fix this bug has not been yet published. This is an imposed tribulation for the fastest players.

Well, let's estimate how long it will take for completing all of the stories in the shortest time.

Input

The first line of the data set contains one integer n ($2 \leqslant n \leqslant 1,000$), which denotes the number of the branching points in this game. The following n lines describe the branching points. The i-th line describes the branching point of ID number i. The first integer k_i ($0 \leqslant k_i \leqslant 50$) is the number of choices at the i-th branching point. $k_i \geqslant 0$ means that the i-th branching point is an ending. Next $2k_i$ integers t_{ij} ($1 \leqslant t_{ij} \leqslant n$) and c_{ij} ($0 \leqslant c_{ij} \leqslant 300$) are the information of choices. t_{ij} denotes the ID numbers of the next branching points when you select the j-th choice. c_{ij} denotes the time to read the story between the i-th branching point and the t_{ij}-th branching point. The branching point with ID 1 is the first branching point. You may assume all the branching point and endings are reachable from the first branching point. You may also assume that there is no loop in the game, that is, no branching point can be reached more than once without a reset.

Output

Print the shortest time in a line.

Examples

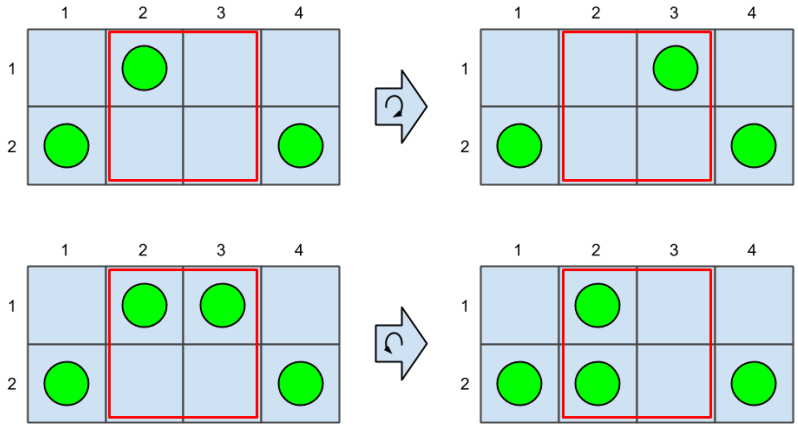
standard input	standard output
2 1 2 2 0	2
6 2 2 1 3 2 2 4 3 5 4 2 5 5 6 6 0 0 0	24
6 3 2 100 3 10 3 10 1 4 100 1 4 10 3 5 1 5 1 6 1 0 0	243

Problem J. Rotation Game

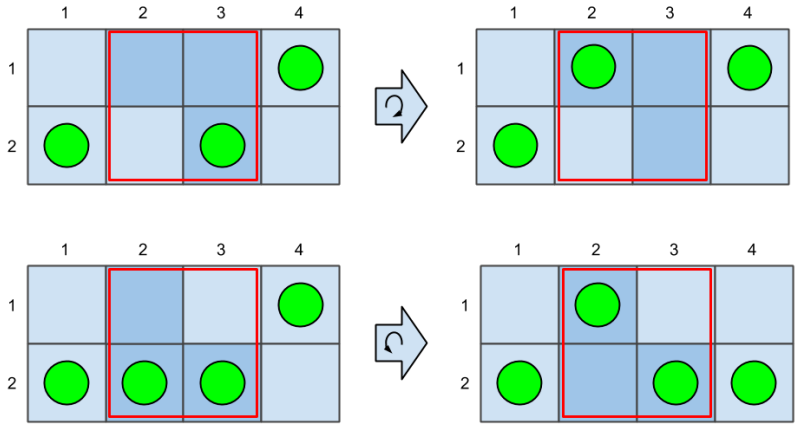
Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	256 mebibytes

You have a board with height two and width W. The board is divided into 1×1 cells. Each row of the board is numbered 1 and 2 from top to bottom and each column is numbered 1 to W from left to right. We denote a cell at the i-th row in the j-th column by (i, j). Initially, some checkers are placed on some cells of the board. Here we assume that each checker looks the same. So we do not distinguish each checker. You are allowed to perform the following operations on the board:

Square rotation: Choose a 2×2 square on the board. Then move checkers in the 2×2 square such that the checkers rotate their position in the square by 90 degrees in either clockwise or counterclockwise direction.



Triangular rotation: Choose a set of cells on the board that form a triangle. Here, we call a set of cells triangle if it is formed by removing exactly one cell from a 2×2 square. In the similar manner with the square rotation, rotate checkers in the chosen triangle in either clockwise or counterclockwise direction.



The objective in this problem is to transform the arrangement of the checkers into desired arrangement by performing a sequence of rotation operations. The information of the initial arrangement and the target arrangement are given as the input. For the desired arrangement, one of the following is specified for each cell (i, j):

- cell (i, j) must contain a checker;
- cell (i, j) must not contain a checker, or
- there is no constraint for cell (i, j).

Compute the minimum required number of operations to move the checkers so the arrangement satisfies the constraints and output it.

Input

The first line of the input contains an integer W ($2 \leqslant W \leqslant 2,000$), the width of the board. The following two lines contain the information of the initial arrangement. If a cell (i, j) contains a checker in the initial arrangement, s_{ij} is '`o`'. Otherwise, s_{ij} is '`.`'. Then an empty line follows. The following two lines contain the information of the desired arrangement. Similarly, if a cell (i, j) must contain a checker at last, t_{ij} is '`o`'. If a cell (i, j) must not contain a checker at last, t_{ij} is '`.`'. If there is no condition for a cell (i, j), t_{ij} is '`*`'. You may assume that a solution always exists.

Output

Output the minimum required number of operations in one line.

Examples

standard input	standard output
3 .oo o.. o.o o..	1
5 .o.o. o.o.o o.o.o .o.o.	3
4 o.o. o.o. .*.o .o.*	4
20 oooo.....ooo......oo .o..o....o..oooo...o ...o*.o.oo..*o*.*o.. .*.o.o.*o*o*..*.o...	25
30 ...o...oo.o..o...o.o........o. .o.oo..o.oo...o.....o........o **o.*....*...*.**...**....o... **..o...*...**..o..o.*........	32

Problem K. Before first zero

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

Consider a sequence B_i defined as follows: First two elements (B_1 and B_2) are natural numbers whose decimal representations consist of N and M "1" digits correspondingly. Following terms are computed according to the formula: $B_i = |B_{i-2} - B_{i-1}|$.

Your program should find such an element B_k that $B_{k+1} = 0$ and $B_j \neq 0$ for all $1 \leqslant j \leqslant k$.

In the second sample: $B_1 = 11$, $B_2 = 1111$, $B_3 = 1100$, $B_4 = 11$, $B_5 = 1089$, $B_6 = 1078, \ldots$, $B_{151} = 11$, $B_{152} = 11$, $B_{153} = 0, \ldots$.

Input

Input file contains two positive integers N and M ($1 \leqslant N, M \leqslant 10^6$).

Output

Output file should contain a single integer — B_k or 0 if there's no zero term in B series. Note that the value of B_k may be very large.

Examples

standard input	standard output
1 1	1
2 4	11

Problem L. Optimal alpha beta pruning

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

Fox Ciel is developing an artificial intelligence (AI) for a game. This game is described as a game tree T with n vertices. Each node in the game has an evaluation value which shows how good a situation is. This value is the same as maximum value of child nodes' values multiplied by -1. Values on leaf nodes are evaluated with Ciel's special function – which is a bit heavy. So, she will use alpha-beta pruning for getting root node's evaluation value to decrease the number of leaf nodes to be calculated.

By the way, changing evaluation order of child nodes affects the number of calculation on the leaf nodes. Therefore, Ciel wants to know the minimum and maximum number of times to calculate in leaf nodes when she could evaluate child node in arbitrary order. She asked you to calculate minimum evaluation number of times and maximum evaluation number of times in leaf nodes.

Ciel uses following algotithm:

```
function negamax(node, a, b)
    if node is a terminal node
        return value of leaf node
    else
        foreach child of node
            val := -negamax(child, -b, -a)
            if val >= b
                return val
            if val > a
                a := val
        return a
```

Input

The first line contains an integer n, which means the number of vertices in game tree T. The second line contains n integers p_i, which means the evaluation value of vertex i. Then, next n lines which contain the information of game tree T. k_i is the number of child nodes of vertex i, and t_{ij} is the indices of the child node of vertex i. Input follows following constraints: $2 \leqslant n \leqslant 100$, $-10^4 \leqslant p_i \leqslant 10^4$, $0 \leqslant k_i \leqslant 5$, $2 \leqslant t_{ij} \leqslant n$. Index of root node is 1.

Evaluation value except leaf node is always 0. This does not mean the evaluation values of non-leaf nodes are 0. You have to calculate them if necessary. Leaf node sometimes have evaluation value of 0. Game tree T is tree structure.

Output

Print the minimum evaluation number of times and the maximum evaluation number of times in leaf node. Please put the whitespace between minimum and maximum.

Examples

standard input	standard output
3 0 1 1 2 2 3 0 0	2 2
8 0 0 100 100 0 -100 -100 -100 2 2 5 2 3 4 0 0 3 6 7 8 0 0 0	3 5
8 0 0 100 100 0 100 100 100 2 2 5 2 3 4 0 0 3 6 7 8 0 0 0	3 4
19 0 100 0 100 0 0 0 0 0 1 2 3 4 5 6 7 8 9 10 2 2 3 0 2 4 5 0 3 6 7 8 3 9 10 11 3 12 13 14 3 15 16 17 2 18 19 0 0 0 0 0 0 0 0 0 0	7 12