Problem A. Takeover

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	512 mebibytes

A new IT company has recently founded their office in the city! Their headquarters are located at the point $(0, 0)$, and the campus is a rectangle with corners at $(0, 0)$ and $(1, 1)$. The campus is very small and has no facilities yet. However, there are many facilities in the neighborhood. Why not to enlarge the campus and have them all inside?

The facilities will be captured and added to the campus one by one. After taking control of each facility, the company has to rebuild the fence around the campus. The fence should be the smallest rectangle with sides parallel to coordinate axes that contains the headquarters and all captured facilities.

It is possible to reuse material from the previous fence, but sometimes the total length of the fence may increase. In this case it is necessary to buy some new material. If the length of the fence before the capture was a meters, and after the capture it became b meters, $b - a$ units of material should be bought.

It is hard to justify the needs for huge buys. Find the order of capturing the facilities such that the maximum increase in the length of the fence after the capture is minimized.

The material required for building the initial fence (of size 4) is not considered an increase, as it is initially present.

Input

In the first line of input, there is an integer n ($1 \le n \le 3 \cdot 10^5$), the number of facilities. In each of the next n lines, there are two integers, x_i and y_i ($1 \le x_i, y_i \le 10^9$), the coordinates of the i-th facility.

It is allowed for two facilities to be located at the same position, and for any facility to be already inside the campus initially.

Output

Print a permutation of numbers from 1 to n: the order in which the facilities should be captured. This order should minimize the maximum increase in the length of the fence after the capture. The facilities are numbered from 1 to n in the order they are given in the input.

If there are multiple answers, print any one of them.

Examples

standard input	standard output

3
1 2
4 4
3 1
	1 3 2

4
1 4
2 3
3 2
4 1
	1 2 3 4

Problem B. Unfair Card Deck

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	512 mebibytes

In yet another card game, there are several types of cards and a deck of 30 cards. For each card type, there is either a single card of that type in the deck or two such cards. Initially, the deck is shuffled. Afterwards, the cards are drawn randomly from the deck one by one until it is empty.

Having played 100 000 games, you noticed that the dealer draws the cards unfairly. Instead of picking the next card randomly, he follows some strange algorithm. The i-th card type is assigned a weight X_i ($0 < X_i \le 1$). On every turn, the probability that the next drawn card would be of type i is $c_i X_i / S$, where c_i is the number of remaining cards of the i-th type and $S = \sum\limits_j c_j X_j$ is the sum of weights of all remaining cards.

In order to be prepared for the next playing session, you want to be able to predict the actions of the dealer. Fortunately, you remember the order in which the cards were drawn in all 100 100 games. Given that information, find the weight assigned to each card type.

Input

In the first line, there are two integers m and n ($m = 100\,000$, $1 \le n \le 30$), the number of games and the number of card types.

In the next line, there are n integers $a_1, \ldots, a_n$ ($1 \le a_1 \le 2$, $\sum\limits_{i=1}^{n} a_i = 30$), denoting how many cards of the i-th type are there in the deck.

Each of the next m lines contains the log of a single game. The log consists of 30 numbers: the types of the cards in the order they were drawn. For each $1 \le i \le n$, the number i occurs in the log exactly a_i times.

Output

Print n numbers $W_1, \ldots, W_n$ ($10^{-300} < W_i \le 1$), the predicted weights of the card types. The answer will be considered correct if, for each pair of types i and j such that $X_i \le X_j$, the following holds:

$$\left| \frac{X_i}{X_i + X_j} - \frac{W_i}{W_i + W_j} \right| < 0.02$$

Note

Small input example (note that it will not occur in the testset, since m is always 100 000):

```
4 3
2 1 1
1 1 2 3
1 2 1 3
1 2 3 1
2 1 1 3
```

Problem C. Diverse Singing

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	512 mebibytes

In the modern world, diversity is the major concern, language diversity in particular. That's why the upcoming talent show is going to be held in multiple languages. However, making the program both diverse and not boring is a tough problem.

There are n singers going to participate in the show and m songs to be performed. Each singer has several songs in their repertoire, and some of them may be sung in different languages. For a program to be complete, each participant should sing at least one song and each song should be sung at least once. For a program to be not boring, each participant should use each language no more than once, and each song should be sung in each language no more than once as well.

Given the repertoire of each singer, make a complete and not boring program of the show, or determine that it is impossible.

Input

In the first line of input, there are three integers n, m, k: the number of singers, songs and possible acts, respectively ($1 \le n, m \le 1000$, $1 \le k \le 10\,000$).

The next k lines describe the repertoires. The i-th line contains three integers p_i, s_i, l_i ($1 \le p_i \le n$, $1 \le s_i \le m$, $1 \le l_i \le k$) denoting that the participant p_i can sing the song s_i in the language l_i.

Output

If it is impossible to make a complete and not boring program, print a single number "`-1`" (without quotes).

Otherwise, on the first line, print an integer t: the number of songs to be performed. On the second line, print t distinct integers between 1 and k: the repertoire entries to include in the program. The entries are numbered from 1 in the order they are given in the input. The repertoire entries can be printed in any order. If there are several possible answers, print any one of them.

Examples

standard input	standard output

`2 2 4`
`1 1 1`
`1 2 1`
`1 2 2`
`2 2 2`
	`2`
	`1 4`

`2 3 5`
`1 1 1`
`1 2 1`
`2 2 2`
`2 3 2`
`2 2 3`
	`3`
	`1 4 5`

`2 3 4`
`1 1 1`
`1 2 1`
`2 2 2`
`2 3 2`
	`-1`

Problem D. Pick Your Own Nim

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	512 mebibytes

Alice and Bob love playing the Nim game (if you do not remember the rules, please refer to the Notes section). They played it so many times that they learned how to determine the winner at a first glance: if there are $a_1, \ldots, a_n$ stones in the heaps, the first player wins if and only if the bitwise xor $a_1 \oplus \ldots \oplus a_n$ is nonzero.

They heard that in some online games players pick their characters before the game, adding a strategic layer. Why not do it with Nim?

They came up with the following version. Alice and Bob each have several boxes with heaps. In the first phase they pick exactly one heap from each box. In the second phase Alice chooses some nonempty subset of those heaps, and the regular Nim game starts on chosen heaps with Bob to move first.

Bob already knows which heaps Alice picked. Help him to perform his picks so that he wins the game no matter which heaps Alice chooses during the second phase.

Input

On the first line, there is a single integer n ($0 \le n \le 60$), the number of heaps picked by Alice.

If $n > 0$, on the next line there are n integers: the sizes of those heaps. Otherwise, this line is omitted.

On the next line there is a single number m ($1 \le m \le 60$), the number of Bob's boxes.

Each of the next m lines contains the description of a box. Each description starts with a number k_i ($1 \le k_i \le 5000$), the number of heaps in the box. Then k_i numbers follow, denoting the sizes of those heaps.

The size of each heap is between 1 and $2^{60} - 1$, inclusive. The total number of heaps in Bob's boxes does not exceed 5000.

Output

If Bob cannot win (that is, no matter what he picks, Alice can make such a choice that the resulting Nim position is losing), print "`-1`" (without quotes). Otherwise, print m integers: the sizes of the heaps Bob should pick from his boxes in the same order in which the boxes are given in the input.

Examples

standard input	standard output

2
1 2
2
2 1 2
3 1 2 3
	-1

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

Note

In the game of Nim, there are several heaps of stones. On each turn, the player selects any heap and takes some positive number of stones from it. The player who takes the last stone wins the game.

Problem E. Permutasino

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	512 mebibytes

At last, the famous spy and secret agent James Bond succeeded in finding his current nemesis, Dr. Zlo. They met at the roulette table of the casino Permutasino, and now 007 tries to guess the latest evil plan of his vicious enemy.

The roulette table consists of $n!$ cells containing all possible permutations of integers between 1 and n. Betting process in this roulette is rather unusual: player puts no more than n bets on some of the cells. Each bet is a non-negative number b_π where π denotes the permutation it corresponds to. The sum of all bets should be equal to 1.

After bets are made, the roulette mechanism picks a random permutation from the distribution defined by the numbers b_π. Formally, the permutation π will be chosen with probability b_π if there is a bet on permutation π, or with probability 0 otherwise.

In order to beat Dr. Zlo in this mind battle and persuade him to disclose his evil plan, 007 has to make bets in such way that the expected value of the resulting permutation is a vector $(x_1, x_2, \ldots, x_n)$. Consider expected value of a random permutation of length n to be defined as a vector of length n, where i-th element is the expected value of the i-th element of the random permutation.

Help James Bond to make appropriate bets, or determine that it is not possible and he has to save the world by using different means (like shooting everyone and blasting every building on his way) this time.

Input

The first line of input contains a single integer n, the length of the permutations appearing at the roulette table ($1 \le n \le 500$).

The second line contains n integers $x_1, x_2, \ldots, x_n$ ($1 \le x_i \le n$), the desired expected value of the resulting permutation.

Output

If there is no way to obtain the vector $(x_1, x_2, \ldots, x_n)$ as the expected value of the game result, print -1.

Otherwise, in the first line, print k ($1 \le k \le n$), the number of bets to be made by James Bond.

In the i-th of the following k lines, print the bet value b_{π_i} ($0 \le b_{\pi_i} \le 1$) and n integers $\pi_{i,1}, \pi_{i,2}, \ldots, \pi_{i,n}$ ($1 \le \pi_{i,j} \le n$, all $\pi_{i,j}$ over all $1 \le j \le n$ are distinct), defining the corresponding permutation π_i.

The sum $b_{\pi_1} + b_{\pi_2} + \ldots + b_{\pi_n}$ should be equal to 1 with an absolute error of no more than 10^{-6}. The maximum value of $|b_{\pi_1}\pi_{1,j} + b_{\pi_2}\pi_{2,j} + \ldots + b_{\pi_k}\pi_{k,j} - x_j|$ over all $1 \le j \le n$ should not exceed 10^{-2}. All calculations to verify these facts will be performed using the double precision floating point data type.

If there are several possible answers, print any one of them.

Examples

standard input	standard output

4
2 2 3 3
	3
	0.5000000000 1 2 3 4
	0.1666666667 1 4 3 2
	0.3333333333 4 1 3 2

2 1 1	-1

Note

In the first sample test, the expected value of the resulting permutation is

$$\left(\frac{1}{2}\cdot 1+\frac{1}{6}\cdot 1+\frac{1}{3}\cdot 4,\ \ \frac{1}{2}\cdot 2+\frac{1}{6}\cdot 4+\frac{1}{3}\cdot 1,\ \ \frac{1}{2}\cdot 3+\frac{1}{6}\cdot 3+\frac{1}{3}\cdot 3,\ \ \frac{1}{2}\cdot 4+\frac{1}{6}\cdot 2+\frac{1}{3}\cdot 2\right) = (2,2,3,3).$$

Problem F. Planar Max Cut

Input file:	*standard input*
Output file:	*standard output*
Time limit:	11 seconds
Memory limit:	512 mebibytes

Some of you may know how to find a Minimum Cut in a graph. Some of you may also have heard about the fact that Maximum Cut problem is NP-complete (more formally, the decision version of finding the cut of value at least k is NP-complete). I bet that some of you even thought about something like: "Hmm, what if I simply negate all costs and transform a Max Cut problem into a Min Cut problem? Where are my Turing Award and billion dollars... Oh, snap, I got it."

It turns out that, for some restricted classes of graphs, problems may become easier than in the general case. Consider the Planar Maximum Cut problem which is formulated as follows. Consider an undirected graph G consisting of n vertices $V = \{v_1, v_2, \ldots, v_n\}$ and m edges $E = \{(v_{a_1}, v_{b_1}), (v_{a_2}, v_{b_2}), \ldots, (v_{a_m}, v_{b_m})\}$. The graph is planar, and you are given its embedding into the plane: besides the description of the graph, you know the coordinates of the vertices such that, if we draw segments corresponding to the edges of the graph, no two edges would share an internal point. There is also an integer cost c_j associated with j-th edge of the graph.

Your task is to partition vertices of the graph into two disjoint sets A and B ($A \cup B = V$) such that the total cost of cut edges is maximum possible. An edge is called a cut edge if exactly one of its endpoints belongs to A and exactly one belongs to B.

Input

The first line of input contains two integers n and m ($1 \le n \le 200$, $1 \le m \le 1000$), the number of vertices of the graph and the number of edges of the graph respectively.

The i-th of the following n lines contains two integers x_i and y_i ($-10^4 \le x_i, y_i \le 10^4$), the coordinates of i-th vertex in the planar embedding. No two points coincide.

The j-th of the following m lines contains three integers a_j, b_j and c_j ($1 \le a_j, b_j \le n$, $a_j \ne b_j$, $0 \le c_j \le 10^5$), the endpoints of the j-th edge and the cost associated with it.

Output

On the first line, print the maximum possible total cost of the cut edges.

On the second line, print n integers $s_1, s_2, \ldots, s_n$ ($s_i \in \{0, 1\}$) where $s_i = 0$ if $i \in A$ and $s_i = 1$ if $i \in B$ in the maximum cut. If there are several possible answers, print any one of them.

Example

standard input	standard output

4 5
0 0
2 0
0 2
2 2
1 2 3
2 4 6
3 4 4
1 3 7
2 3 8
	21
	0 0 1 1

Problem G. Battle Royale

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	512 mebibytes

Have you ever played Battle Royale games? In this kind of games, players are left alone and empty in a large area, and they must collect weapons, ammo, medkits and armor in order to kill each other. The game ends when only one player remains, this player is declared the winner.

To provoke players to fight each other, the safe zone is introduced. Players who are outside the safe zone are getting damage. Safe zone is always getting smaller and smaller, until it becomes a single point, so sooner or later each player has to make a choice: get inside a safe zone and fight other players or stay outside, lose hit points and eventually die.

In one-dimensional world, there is a famous Battle Royale game: BattleUnknown's Playergrounds. The game area is a segment $[L, R]$. Initially, the safe zone is the whole segment $[L, R]$, and in the end of the game it will narrow to a single point M ($L \le M \le R$), where M is chosen equiprobably on the segment $[L, R]$. The safe zone narrows in such a way that $\frac{M-L}{R-M} = const$, and each second its length decreases by 1.

Today we got a match between n extremely passive players: the i-th of them is hiding in the house with the coordinate x_i and is not going to move. Imagine it, they better die outside the safe zone than move out of their houses! Their plan is to outlive all others using the medkits they have collected: the i-th player can stay alive for a_i seconds outside the safe zone.

For each player, determine the probability that this player wins the game.

Input

The first line contains three integers n, L and R: the number of players and the endpoints of the game segment ($1 \le n \le 10^5$, $-10^6 \le L < R \le 10^6$).

The next line contains n integers x_i: coordinates of the houses where players hide ($L < x_i < R$). They are all different and sorted in ascending order.

The next line contains n integers a_i: the time i-th player can stay alive outside the safe zone ($0 \le a_i \le 10^6$).

Output

Output n lines. The i-th line should contain a single real number: the probability that the i-th player wins the game. The absolute error of each number should not exceed 10^{-9}.

Examples

standard input	standard output

2 -5 5
-1 1
3 5
	0.438447187191170
	0.561552812808830

2 0 10
5 7
5 2
	1.000000000000000
	0.000000000000000

3 0 10
3 4 7
3 4 8
	0.109584240176570
	0.121686455414586
	0.768729304408844

Problem H. Jeopardy

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	512 mebibytes

You are playing the final round of an intellectual game. A total of n topics were selected, and each of the n authors prepared a question for each topic. For each topic i and each author j, you know the probability that you will answer the question from the j-th author on the i-th topic.

Before you actually get the question, $n-1$ turns are made. Two actions happen during each turn. First, you discard all questions of any single topic. Second, the moderator discards all questions from any single author. Eventually only one question remains. If you answer it correctly, you win the round.

Of course, you discard columns trying to maximize the probability of your victory. On the contrary, the moderator tries do minimize it.

If you both play optimally, what is the probability that you answer the remaining question?

Input

In the first line, there is a single integer n ($1 \le n \le 500$), the number of topics and authors. Then follow n lines with n integers on each line. The i-th line corresponds to the i-th topic, and the j-th number on it is the probability that you will answer the question by the j-th author.

The probabilities are given as percentages. All integers in the description of topics are between 0 and 100.

Output

Print a single number: the desired probability as a percentage.

Examples

standard input	standard output

2
1 100
99 0
	1

3
0 50 100
100 0 50
50 100 0
	0

Problem I. Slippers

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	512 mebibytes

"Who wakes up first gets the slippers", a Russian proverb says. However, in our campus it is not that easy. You not only have to come early (all slippers will be already taken otherwise), but also are to obey the strict rules of the shoe case.

The shoe case looks like an $n \times m$ grid. Each cell contains a slipper, either left or right. Initially, each slipper is rotated in one of the four directions: left, right, forwards or backwards. It is allowed to take any adjacent pair of slippers (not necessarily distinct) and rotate one of them 90° clockwise and another 90° counterclockwise. When you find a pair of slippers in natural position, you may put them on and finally go away, happy and warm.

A pair of slippers is *in natural position* if:

- their cells share a side;
- they face the same direction;
- if you look at the two cells along that direction, one cell is to the right and another is to the left. Also, there is a right slipper in the right cell and the left slipper in the left cell.

Informally, a normal person can jump into this pair of slippers naturally. Please refer to the "Notes" section for examples.

Assuming everyone is altruistic and performs in an optimal way, find the maximum number of people that can walk away with their pair of slippers put on.

Input

In the first line of input, there are two integers n and m ($1 \le n, m \le 100$), the dimensions of the grid. Each of the next n lines contain m space-separated strings describing the slippers in corresponding cells. Each slipper is described by a string of length 2. The first character is either "L" or "R", denoting left and right slipper respectively. The second character is one of "<", ">", "^" or "v". It means that the slipper initially faces left, right, forwards or backwards, respectively.

Output

Print one integer: the maximum number of pairs of slippers that can be formed.

Examples

standard input	standard output
2 2 R^ L> L< R^	2
3 2 L^ R^ R< L< L< R>	2

Note

Consider the first sample. First, we rotate two left slippers: top counterclockwise, bottom clockwise. Second, we rotate two top slippers: left counterclockwise, right clockwise. After that we have two pairs of slippers in natural position: two in the top row and two in the bottom row.

```
R^ L>      R< L>      Rv Lv
L< R^      L^ R^      L^ R^
```

Here is another possible sequence of actions in this example, leading to a different final picture.

```
R^ L>        R< Lv        R< L>
L< R^        L< R^        L< R>
```

Problem J. The Good, the Bad and the Ugly

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	512 mebibytes

This problem was supposed to have a nice long legend about the Wild Wild West, but the author did not manage to write it in time, so use the power of your imagination!

Consider a number line. A player initially stands at the position $x = p$. At the beginning of each round, you can say either "+" or "-". After that, the player changes position according to what you said. More precisely, if you say t and the player stood at position x, then he moves to position $x' = x + d_t$, where d_+ and d_- are two integer constants.

You do not know the exact values p, d_0 and d_1, but you know that the player is either the Good, the Bad or the Ugly (yeah, imagination!):

- The Good player has $p = m$, $d_+ = 2$, $d_- = -1$;
- The Bad player has $p = -m$, $d_+ = 1$, $d_- = -2$;
- The Ugly player has either $p = m$ or $p = -m$ and either $d_+ = 1$ and $d_- = -1$ or $d_+ = -1$ and $d_- = 1$.

As you can see, the starting position of the player depends on some integer constant m ($1 \le m \le 1000$)... unfortunately, you do not know it too.

After each round, the player tells you if he now stands at $x = 0$ or not.

It appears that, by playing several rounds, you can uniquely determine if the player is Good, Bad or Ugly. Do it in no more than $30m$ rounds.

In each test, the values m, p, d_+ and d_- are chosen according to the above rules. They are fixed in advance and don't change during the checking process.

Interaction Protocol

This is an interactive problem.

If you want to play a round, print either "+" or "-" on a separate line. In response, you will get a line containing either 1 if the player arrived at position $x = 0$, or 0 if the player stands somewhere else.

If you are ready to guess the type of the player, print a line containing the character "`!`", a space and one of the words "`good`", "`bad`" or "`ugly`". After that, your program must terminate.

If after playing $30m$ rounds you do not provide the answer, your solution will get a "`Wrong Answer`" outcome.

To prevent output buffering, flush the output buffer after each printed line: this can be done by using, for example, `fflush (stdout)` in C or C++, `System.out.flush ()` in Java, `flush (output)` in Pascal, or `sys.stdout.flush ()` in Python.

Example

standard input	standard output
	-
0	
	-
1	
	! good