Problem A. Accurate Driver

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

Mr. Doitsafe, a.k.a. Mr. D, is famous for his thoroughly safe driving. Not only does he always drive the car at exactly the maximum allowed speed, but also he immediately stops the car if a traffic light turns red from green when he just enters a crossing, and he immediately starts the car at exactly the maximum allowed speed when a traffic light just turns green from red.

Mr. D's next driving course is a straight road, L units in length, and the maximum allowed speed is 1 unit per second. Mr. D will start his drive at time 0. The road has N traffic lights numbered 1 through N. The traffic light i is at a distance of x_i units from the starting point. At time 0, all the N traffic lights just turned green from red. The i-th traffic light turns red from green after g_i seconds, then turns green from red after r_i seconds, then again turns red from green after g_i seconds, then once more turns green from red after r_i seconds, and so on.

In this situation, Mr. D will start from the starting point and run the car at the speed of 1 unit per second. If the i-th traffic light is green or just turns green from red (but not just turns red from green) when Mr. D reaches x_i, Mr. D won't stop and will go through the crossing at the speed of 1 unit per second. If the i-th traffic light is red or just turns red from green (but not just turns green from red) when Mr. D reaches x_i, Mr. D will stop until the i-th traffic light turns green.

Your task is, given the descriptions of N traffic lights, to compute the time in seconds when Mr. D reaches point L.

Input

The first line of the input consists of two integers, the number N ($1 \leq N \leq 100\,000$) of traffic lights on the road and the length L ($1 \leq L \leq 10^9$) of the road.

The i-th of the following N lines has three integers, x_i, g_i, and r_i, where x_i ($1 \leq x_i < L$) is the position of the i-th traffic light from the start point, g_i ($1 \leq g_i \leq 10^9$) is the duration the i-th traffic light is green, and r_i ($1 \leq r_i \leq 10^9$) is the duration the i-th traffic light is red.

You can assume all the positions of the traffic lights are different. In other words, $x_i \neq x_j$ holds for all $i \neq j$.

Output

Output a line with a single integer which is the time in seconds when Mr. D reaches point L.

Examples

standard input	*standard output*

3 10
3 3 3
6 2 2
9 3 6
	19

1 101 50 900 1	101

Problem B. Board Game With Cards

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

You got $H \cdot W$ cards, where $H \cdot W$ is even. Each card has a number 1 through $H \cdot W/2$ on the face, and each such number is on the face of exactly two cards. You considered what types of card games you could play with the cards, and decided to play the following board game.

You first align $H \cdot W$ cards face down on an $H \times W$ rectangular field. Your goal is to remove all the cards from the field by repeating turns. On each turn, you must flip exactly two cards. If the two flipped cards have the same number on the face, you remove the two cards from the field. If not, you flip the two cards again to make them face down. You have perfect memory, so you remember the positions and numbers of every card you flipped. So, on each turn, you will act as follows.

1. If you already know the positions of two cards with the same number, flip any such pair of cards and remove them from the field.
2. If not, flip a card you haven't flipped yet which has the highest precedence. We define the precedence of cards later.
3. If you have seen the number on the flipped card on another card already, flip this other card and remove the two cards from the field.
4. If not, flip another card you haven't flipped yet which has the highest precedence.
5. If the first and the second cards you flipped on this turn luckily have the same number, remove the two cards from the field.
6. If not, put the two flipped cards face down to prepare for the next turn.

Let us number the rows 1 through H from the top. The card at the topmost row among the remaining cards has the highest precedence. If there are multiple cards at the topmost row, if the row is initially an odd-numbered row, the leftmost card has the highest precedence. If the row is initially an even-numbered row, the rightmost card has the highest precedence.

After you played the game, you noticed you forgot to count how many turns you took to remove all the cards from the field. Fortunately, you remember the initial placement of the cards. So you decided to write a program to compute the turns you made to remove all the cards for a given initial placement.

Input

The first line contains two integers H ($1 \leq H \leq 100$) and W ($1 \leq W \leq 100$). You can assume that $H \times W$ is even.

Each of the following H lines has exactly W integers. The j-th integer of the i-th row represents the number on the face of the card at the i-th row from the top and the j-th column from the left. You can assume all the integers in the H lines are between 1 and $H \cdot W/2$, and each such integer appears exactly twice.

Output

Print a single integer which is the number of turns you take to remove all the cards.

Examples

standard input	*standard output*

2 6
1 1 5 4 4 5
3 2 6 2 6 3
	9

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

1 10 5 4 3 2 1 1 2 3 4 5	7

Problem C. Count the Permutations

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

You are given an undirected tree T with N vertices numbered 1 through N. An edge between vertices u and v in T is denoted as $\{u, v\}$.

Let $P = (p_1, p_2, \ldots, p_N)$ be a permutation of $(1, 2, \ldots, N)$. A permutation is *fitting* when, for each edge $\{u, v\}$ of the tree T, the edge $\{p_u, p_v\}$ also belongs to the tree T.

Compute the number of fitting permutations among the $N!$ possible permutations. Since the answer may be very large, find it modulo 998 244 353.

Input

The first line of the input contains an integer N, the number of vertices in the tree T ($1 \le N \le 100\,000$).

The i-th of the following $N - 1$ lines contains two integers u_i and v_i which mean that there is an edge $\{u_i, v_i\}$ in the tree T ($1 \le u_i, v_i \le N$).

It is guaranteed that the given graph is a tree.

Output

Print a line with a single integer: the answer modulo 998 244 353.

Examples

standard input	*standard output*

4
1 3
4 1
2 1
	6

4
3 4
1 2
2 3
	2

6
6 4
1 3
4 5
2 3
4 3
	8

Problem D. Decompress and Sort

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	1024 mebibytes

Your task is to sort the given N strings $s_1, s_2, \ldots, s_N$ which are given in compressed form.

Compression works as follows. An m-times repetition of a non-empty character sequence *seq* can be compressed into "m(*seq*)", where $m \geq 2$ is an integer. When *seq* consists of just one letter `c`, we may omit the parentheses and write "m`c`". Formally, a compressed string is defined by the following BNF grammar:

```
<String> ::= <Letter> | <Number> <Letter> | <Number> '(' <String> ')' | <String> <String>
<Letter> ::= 'A' | 'B' | ... | 'Z'
<Number> ::= <Digit> | <Number> '0' | <Number> <Digit>
<Digit> ::= '1' | '2' | ... | '9'
```

For example, `ACMACMACMICPCICPCACMACMACMICPCICPCXXXXXXXXXX` can be compressed into:

`3(ACM)ICPCICPCACMACMACMICPCICPCXXXXXXXXXX`

by replacing the first occurrence of `ACMACMACM` with its compressed form. Similarly, by replacing the following repetitions of `ICPC`, `ACM`, and `X`, we get:

`3(ACM)2(ICPC)3(ACM)2(ICPC)10X`

Since `X` is a single letter, parentheses are omitted in this compressed representation. Finally, we have:

`2(3(ACM)2(ICPC))10X`

by compressing the repetitions of `3(ACM)2(ICPC)`. As you may notice from this example, parentheses can be nested.

Input

The first line contains an integer N, the number of compressed strings ($1 \leq N \leq 50$).

The i-th of the following N lines contains the compressed string s_i. The length of the given compressed form is between 1 and 2000 characters, inclusive. Each decompressed string consists of uppercase English letters, and its length is at most 10^{10}.

Output

Print N lines. If the decompressed s_x is the i-th lexicographically smallest string, print x on the i-th line. Equal strings must be sorted in the order in which they appear in the input.

Examples

standard input	*standard output*
4 3(IC)PC I3(CI) ICICICPC 2(5I)	2 1 3 4
5 2(2(2(2X))) 4(4X) 16X 8X8X 8XX	5 1 2 3 4

Problem E. Expand the Logical Formula

Input file:	`standard input`
Output file:	`standard output`
Time limit:	13 seconds
Memory limit:	1024 mebibytes

This problem is about the special case of the Satisfiability Problem (SAT). Let us introduce the definition of SAT first.

A *SAT instance* is a boolean logic formula consisting of several boolean variables combined by AND ($\wedge$), OR ($\vee$), and NOT ($\neg$) operators and parentheses. An *assignment* is a mapping from variables to boolean values. An assignment is *satisfying* a formula if and only if the formula is evaluated to be true with this assignment. A *literal* is either a variable or its negation. A *clause* is a list of literals concatenated with OR. A formula is in *Conjunctive Normal Form* (CNF) if it consists of clauses concatenated with AND. In the following, we only consider CNF formulae as SAT inputs because every formula can be converted to an equivalent CNF formula.

2-SAT is a special case of SAT where the length of clauses is limited to 2. For example, $(x \vee y) \wedge (\neg x \vee z)$ is a 2-SAT instance consisting of 3 variables and 2 clauses. The assignment $x = \texttt{false}$, $y = \texttt{true}$, $z = \texttt{true}$ is one of the satisfying assignments for this formula.

You are given a 2-SAT instance in CNF with N variables and M clauses. The i-th variable is denoted by x_i and this i is called its index. In all clauses, the difference between the indices of the two variables is less than or equal to 2.

Let C_k be the number of satisfying assignments where exactly k variables are true. Your task is to write a program that calculates C_k for all k from 0 to N. Since the answers may be huge, find them modulo 998 244 353.

Input

The first line of the input consists of two integers, the number of variables N ($1 \le N \le 100\,000$) and the number of clauses M ($1 \le M \le 100\,000$).

The following M lines represent the clauses in the 2-SAT instance. The i-th of them corresponds to the i-th clause and contains two integers A_i and B_i representing the literals in this clause. They satisfy $1 \le |A_i|, |B_i| \le N$ and $||A_i| - |B_i|| \le 2$, and each of them has the following meaning: If it is a positive integer a, the literal is x_a (without negation). If it is a negative integer b, the literal is $\neg x_{-b}$ (with negation).

Output

Output $N + 1$ lines. The i-th line must contain a single integer: the value $C_{i-1} \bmod 998\,244\,353$.

Examples

standard input	standard output

4 2
1 -3
2 2
	0
	1
	2
	2
	1

3 6
1 2
2 3
2 -1
1 3
2 -3
-2 3
	0
	0
	1
	1

Problem F. Find The Length

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	1024 mebibytes

You are given an integer sequence $A = (a_1, a_2, \ldots, a_N)$. Find the length of the longest sequence $B = (b_1, b_2, \ldots, b_M)$ which satisfies the following two conditions:

- B is a subsequence of A
- $b_i < b_{i+2}$ for all i $(1 \le i \le M - 2)$

A subsequence of a sequence is a sequence obtained by removing zero or more elements from the original sequence and then concatenating the remaining elements without changing the order.

Input

The first line contains an integer N, the number of elements in A $(1 \le N \le 5000)$.

The second line consists of N integers between 1 and N, inclusive. For each i $(1 \le i \le N)$, a_i represents the i-th element of A $(1 \le a_i \le N)$.

Output

Print one integer: the answer to the problem.

Examples

standard input	*standard output*
8 1 5 7 8 6 3 4 2	4
8 1 4 2 8 5 7 1 4	5
2 1 2	2
6 2 2 3 3 5 5	6

Problem G. Goodness

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	1024 mebibytes

You are given two integer sequences $A = (a_1, \ldots, a_N)$ and $B = (b_1, \ldots, b_M)$ such that $N \geq M$, that is, A is always longer than or has the same length as B.

Each element is a positive integer less than or equal to L. The *goodness* of the pair of sequences (A, B) is defined to be the number of bijections (that is, one-to-one invertible functions) f from $1, \ldots, L$ to $1, \ldots, L$ such that B is a contiguous subsequence of $(f(a_1), \ldots, f(a_N))$.

A contiguous subsequence of a sequence is a sequence obtained by removing zero or more elements from the beginning and the end of the original sequence.

Given two sequences A and B, calculate the goodness of the pair (A, B). Since the answer may be huge, find it modulo $998\,244\,353$.

Input

The first line of the input consists of three integers: N, the length of the sequence A ($1 \leq N \leq 300\,000$), M, the length of the sequence B ($1 \leq M \leq N$), and L, the largest integer allowed ($1 \leq L \leq 300\,000$).

The second line consists of N positive integers from 1 to L, inclusive, that represent the sequence A.

The third line consists of M positive integers from 1 to L, inclusive, that represent the sequence B.

Output

Print the goodness of the pair of sequences (A, B) modulo $998\,244\,353$.

Examples

standard input	*standard output*

`9 3 3`
`1 1 2 3 2 1 1 2 1`
`1 1 2`
	`1`

`1 1 6`
`2`
`2`
	`120`

`8 3 3`
`3 2 3 2 3 1 3 1`
`1 2 1`
	`4`

Problem H. Hardtown

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	1024 mebibytes

Hardtown is a city consisting of N towns numbered from 1 to N. The previous mayor of this city constructed a single road between each pair of towns. However, every road is not wide enough and hence is one-directional. In other words, for any two different towns i and j, there is a single road that you can pass either from i to j or from j to i, but not both.

Because of the sloppy city planning, you suspect that there may be two different towns such that you cannot travel from one town to the other by passing through one or more roads. If so, as a new mayor of this city, you have to resolve this problem. Unfortunately, there is not enough space to make each road bidirectional nor construct new roads. Therefore, you instead decided to reverse the directions of some roads.

For each pair of towns, you are given the initial direction of the road between these two towns and the cost to reverse the direction. You can reverse the directions of zero or more roads. After that, you must be able to travel from any town to any other town by passing through roads. Your task is to calculate the minimum total cost to achieve it. Under the constraints of this problem, it can be proven that a solution always exists.

Input

The first line consists of an integer N between 3 and 3000, inclusive. This represents the number of towns in this city.

The i-th of the following $N-1$ lines consists of $N-i$ non-zero integers $c_{i,i+1}, c_{i,i+2}, \ldots, c_{i,N}$, each between -10^9 and 10^9, inclusive. For each i and j ($1 \leq i < j \leq N$), $c_{i,j}$ represents the information about the road between towns i and j. If $c_{i,j}$ is positive, then you can initially pass through this road from i to j only. Otherwise, you can initially pass through this road from j to i only. In either case, the absolute value $|c_{i,j}|$ is the cost to reverse the direction of this road.

Output

Output a line with a single integer: the minimum total cost of the roads which can be reversed so that you can travel from any town to any other town.

Examples

standard input	*standard output*

7
-17 -76 -46 -94 83 -22
53 -59 95 42 82
-31 66 26 12
71 96 56
65 -29
-23
	57

7
-17 -76 -46 -94 83 -22
53 -59 95 42 82
31 66 -26 12
71 96 56
65 -29
-23
	0

Problem I. Island and Checkpoints

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	1024 mebibytes

Consider a rectangular island on the plane with sides parallel to coordinate axes. The bottom left and top right corners of the island are located at $(0, 0)$ and (W, H), respectively.

There are N checkpoints on the island. The i-th checkpoint is located at (x_i, y_i). Your task is to find a point on the island that maximizes the (Euclidean) distance to the nearest checkpoint. Find the distance from such a point to its nearest checkpoint.

In other words, calculate the value of

$$\max_{\substack{0\le x\le W\\0\le y\le H}} \min_i \sqrt{(x-x_i)^2+(y-y_i)^2}.$$

Input

The first line contains three integers: the number N ($1 \le N \le 2000$) of stations, the width W and the height H ($1 \le W, H \le 1000$) of the island.

The i-th of the following N lines contains two integers x_i and y_i ($0 \le x_i \le W$, $0 \le y_i \le H$) which represent the coordinates of the i-th checkpoint. The coordinates of the checkpoints are distinct: $(x_i, y_i) \ne (x_j, y_j)$ for any i, j ($i \ne j$).

Output

Print a line with a single real number: the answer to the problem. The answer will be considered correct if its absolute or relative error is at most 10^{-6}.

Examples

standard input	*standard output*
1 2 2 0 0	2.8284271247
1 6 6 3 3	4.2426406871

2 7 7
3 1
3 5
	4.4721359550

4 11 11
1 1
1 10
10 1
10 10
	6.3639610307

Problem J. Joy of Tracking

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

You are a big fan of a particular feature of an online map service: route tracking. You enjoy drawing pictures on the online map: first, design a route looking like the desired image, and then actually trace the route with your mobile device.

One day, you noticed the town has a grid-shaped train network. On the map, there are $H \times W$ stations on a grid with H horizontal lines and W vertical lines. Each crossing point has exactly one station, and each station is connected to all the stations adjacent to it vertically or horizontally (but not diagonally). Connections can have different fees, but the fee to move from station A to station B is always the same as the fee to move from B to A. If you use a connection multiple times, you have to pay the fee each time you use it.

You plan to draw a complete grid on the map by going through all the connections on the train network at least once. You have to start and finish the route at the same station. Under these constraints, you want to minimize the total cost of travel. As you are also good at programming, you decided to write a program to calculate the minimum cost when you design an optimal route.

Input

The first line contains two integers H ($2 \le H \le 100$) and W ($2 \le W \le 100$) which represent that the train network grid consists of H rows and W columns. Let (i, j) be the crossing at the i-th row from the top and the j-th column from the left.

The i-th of the following $H-1$ lines contains W integers, where the j-th integer is the fee to move between the station at (i, j) and the station at $(i+1, j)$.

The i-th of the following H lines contains $W-1$ integers, where the j-th integer is the fee to move between the station at (i, j) and the station at $(i, j+1)$.

All the fees are at least 0 and at most 10^9.

Output

Output a line with a single integer: the minimum cost to draw a complete grid by taking trains on the train network.

Examples

standard input	*standard output*

`2 4`
`2 2 2 2`
`1 1 1`
`1 1 1`
	`16`

`4 3`
`3 2 0`
`6 1 0`
`7 1 6`
`7 5`
`8 1`
`8 3`
`3 5`
	`76`

Problem K. Klingon Attack

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

You are developing a video game. The goal of the game is to avoid attacks from the famous alien race called the Klingons. The player initially stands at the origin of an infinite two-dimensional plane. The player can move in arbitrary directions at speeds up to the speed limit specified by the game.

There will be N attacks from the Klingon ships. The i-th attack occurs at time T_i and bombs every point of the plane except for the safety zone. The shape of the safety zone may be different for each attack, but it is always a convex polygon on the plane. To avoid the bombing when it occurs, the player must be inside this safety zone. The border of the safety zone is also considered safe. The player clears this game if they successfully avoid all the N attacks. The player knows the information about all the N attacks prior to the beginning of this game.

You have already determined the shapes of the safety zones of all the N attacks. The remaining task is to set the speed limit for the player. Of course, the faster the player can move, the easier the game is. On the other hand, if the limit is too low, it may become impossible to clear this game. In order to adjust the difficulty of this game, please calculate the minimum speed limit that makes it possible to clear the game.

Input

The first line consists of an integer C ($1 \le C \le 20$) representing the number of test cases in the input.

Each test case starts with a line containing a single integer N: the number of Klingon attacks in this test case ($1 \le N \le 20$). The descriptions of the attacks follow.

The description of the i-th attack starts with a line consisting of two integers: T_i, the time of the attack ($1 \le T_i \le 100$), and M_i, the number of vertices of the polygon which is the safety zone for the i-th attack ($3 \le M_i \le 20$). Then M_i lines follow, each containing two integers $(x_{i,j}, y_{i,j})$ representing the coordinates of the j-th vertex of the i-th polygon ($-100 \le x_{i,j} \le 100$, $-100 \le y_{i,j} \le 100$). The vertices of polygons are given in counter-clockwise order.

It is guaranteed that all the given polygons are convex. No three vertices of a single polygon are on the same line. You may assume that $T_1 < T_2 < \ldots < T_N$.

Output

For each test case, print a line with a single real number which is the minimum speed limit that makes it possible to clear the game. In particular, if the player can avoid all the bombings even without moving, the answer is 0. Each answer is considered correct if its absolute or relative error is at most 10^{-5}.

Examples

standard input	*standard output*

2
2
10 3
3 5
7 5
3 9
15 4
0 9
8 1
16 9
8 17
3
10 3
1 0
2 0
1 1
15 3
2 0
2 1
1 1
30 3
3 2
5 2
5 3
	0.5830951894904501
	0.1201850425186421

1
3
1 4
0 0
5 0
5 5
0 5
2 4
0 0
-5 0
-5 -5
0 -5
3 4
5 5
-5 5
-5 -5
5 -5
	0.0000000000009095

Problem L. Liar Game

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

You are playing the game “Liar, Liar”. In this game, N people say facts about some topic: $N-1$ people tell true facts, and the remaining person tells a lie. You win if you identify who is the liar from the facts they said.

This time, the topic of the game is the heights of K mountains, numbered 1 through K. Among N people, the i-th person says “mountain a_i is x_i meters higher than mountain b_i”. If you exclude the liar from N people, the facts that the other $N-1$ people say have no contradiction. On the other hand, if you exclude a person who is not a liar, the facts that the other $N-1$ people say must contradict.

Your task is to write a program to identify who is the liar among N people from the facts they said.

Input

The first line consists of two integers, the number N ($2 \le N \le 200\,000$) of people and the number K ($3 \le K \le 200\,000$) of mountains.

The following N lines represent facts that N people say. The i-th line contains three integers a_i ($1 \le a_i \le K$), b_i ($1 \le b_i \le K$), and x_i ($1 \le x_i \le 10^9$), which mean that the i-th person says “mountain a_i is x_i meters higher than mountain b_i”. You can assume $a_i \ne b_i$. Also, you can assume $(a_i, b_i) \ne (a_j, b_j)$ and $(a_i, b_i) \ne (b_j, a_j)$ hold for all $1 \le i < j \le N$. The input is consistent with the situation where there is exactly one liar.

Output

Output a line with a single integer i which means the i-th person is the liar.

Examples

standard input	standard output

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

```
8 5
1 3 4
3 2 1
4 2 2
1 4 3
1 5 1
4 5 7
5 3 3
5 2 4
```
	`6`