Problem A. Stop & Go

- Time Limit: 2 sec

Problem Statement

Mr. Drive, a.k.a. Mr. D, is famous for his thorough safe driving. Not only he always drives a car at an exact legal speed, but also he immediately stops a car if a traffic light turns red from green when he just enters a crossing, and he immediately starts a car at an exact legal speed when a traffic light just turns green from red.

Mr. D's next driving course is a simple straight road with length L and the legal speed limit 1 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 from the start point. At time 0, all the N traffic lights are green. The i-th traffic light turns red from green after g_i seconds, then turns green from red after r_i, and then turns red from green after g_i seconds, then turns green from red after r_i, and so on.

In this situation, Mr. D will start from the start point and run a car at speed 1 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 go through the crossing at speed 1 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 to compute the time in seconds when Mr. D reaches point L, for given N traffic lights.

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

Sample Input 1

```
3 10
3 2 2
6 1 1
9 2 5
```

Sample Output 1

```
15
```

Sample Input 2

```
1 100
50 1000 1
```

Sample Output 2

```
100
```

Sample Input 3

```
3 100
70 10 50
20 10 15
50 50 10
```

Sample Output 3

```
150
```

Problem B. 1-Player Concentration

- Time Limit: 2 sec

Problem Statement

You got HW cards, where HW is even. Each card has a number 1 through $HW/2$ on the face, and exactly two cards have the same number. You considered what types of card games you could play with the cards, and decided to play 1-player concentration.

In the 1-player concentration, you first align HW cards face down on the $H \times W$ rectangle field. Your goal is to remove all the cards from the field by repeating turns. For 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. But because you have perfect memory, you can remember which card has which number and where it's placed if you flipped the card once. So for each turn, you will act as follows.

1. If it's the second or later turn and you already know where two cards having the same number are placed according to previous turns, flip the two cards and remove them from the field.
2. If not, flip a card you haven't flipped yet and with the highest precedence (We define the precedence of cards later). If you have seen the number on the card on another already flipped card, flip the other card and remove the two cards from the field.
3. If not, flip another card you haven't flipped yet and with the highest precedence. If the first and the second cards you flipped in this turn (luckily) have the same number, remove the two cards from the field.
4. If not, make the two flipped cards face down to prepare for the next turn.

Let's 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 take 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 $H \times W$ 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 a card at the i-th from the top and the j-th from the left. You can assume all the integers in the H lines are no less than 1, no more than $HW/2$, and the same integer appears exactly twice.

Output

Output in a line a single integer, which is the number of turns you take to remove all the cards.

Sample Input 1

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

Sample Output 1

```
9
```

Sample Input 2

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

Sample Output 2

```
6
```

Sample Input 3

```
1 10
1 2 3 4 5 5 4 3 2 1
```

Sample Output 3

```
7
```

Problem C. Track Train Trail

- Time Limit: 2 sec

Problem Statement

You are a big fan of a feature of an online map service: route tracking. You enjoy drawing a picture on the online map by designing a route looking like a target figure beforehand and actually tracing the route with your mobile device.

One day, you noticed a 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, where 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 from B to A.

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 a route at the same station. Under these constraints, you want to minimize the total cost of train fees. 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 ($2 \leq H \leq 100$) and W ($2 \leq W \leq 100$) 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 in a line a single integer, which is the minimum cost to draw a complete grid by taking trains on the train network.

Sample Input 1

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

Sample Output 1

```
18
```

Sample Input 2

```
4 3
4 3 1
7 2 1
8 2 7
8 6
9 2
9 4
4 6
```

Sample Output 2

```
98
```

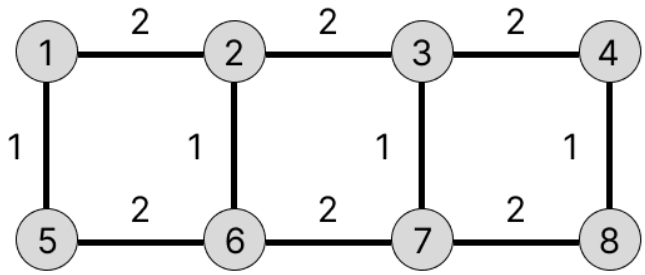

Figure C.1. Train network of the first example input

The train network corresponding to the first example is illustrated in Figure C.1. Let's denote a route on which you visit station numbered a_i in the figure as the i-th station by $[a_1, a_2, \ldots, a_k]$. For example, the minimum cost 18 is achieved by a route $[2, 1, 5, 6, 2, 3, 7, 8, 4, 3, 7, 6, 2]$.

Problem D. 2-LIS

- Time Limit: 2 sec

Problem Statement

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 $(1 \leq i \leq M - 2)$

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

N
a_1 a_2 $\ldots$ a_N

The first line consists of an integer N between 1 and 5,000, inclusive. This represents the number of elements in A.

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

Output

Print the answer in a line.

Sample Input 1	Sample Output 1
7 1 5 7 6 3 4 2	4

Sample Input 2	Sample Output 2
7 6 2 1 4 7 5 3	5

Sample Input 3	Sample Output 3
2 2 1	2

Sample Input 4	Sample Output 4
6 1 1 2 2 3 3	6

Problem E. Sharp 2-SAT

- Time Limit: 8 sec

Problem Statement

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

A SAT instance is a boolean logic formula consisting of several boolean variables combined by AND, OR, and NOT operators and parentheses. An assignment is a mapping from variable to boolean value. An assignment is satisfying a formula if and only if the formula is evaluated to be true with the given 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 the equivalent CNF formula. 2-SAT is a special case of SAT where the length of clauses are 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. $x = \text{false}, y = \text{true}, z = \text{true}$ 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 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 ks from 0 to N.

Since the answer may be huge, print the answer modulo $998{,}244{,}353$.

Input

The input consists of a single test case with the following format.

$$
\begin{aligned}
&N\ M \\
&A_1\ B_1 \\
&\vdots \\
&A_M\ B_M
\end{aligned}
$$

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

The following M lines represent the clauses in the 2-SAT instance. Each line corresponds to one of the clauses. These lines contain 2 integers A_i and B_i representing the literals in the clause. They satisfy $1 \leq |A_i|, |B_i| \leq N$ and $||A_i| - |B_i|| \leq 2$ 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 contains a single integer C_{i-1} modulo $998{,}244{,}353$.

Sample Input 1	Sample Output 1
4 2 2 2 1 -3	0 1 2 2 1

Sample Input 2	Sample Output 2
2 4 -2 2 2 2 2 1 -1 2	0 1 1

Sample Input 3	Sample Output 3
40 10 13 -12 6 4 -19 -17 -6 4 -10 -10 16 17 32 34 -3 -2 -11 -12 37 -36	0 0 0 4 130 2052 20958 155678 896220 4160798 16004412 51999948 144776190 349181040 735692490 364558777 233997014 242620671 184707628 808271668 924963778 496489648 654440271 639886064 690035227 954668130 474278220 205649340 77168754 24784920 6715462 1505398 271728 37950 3848 252 8 0 0 0 0

Problem F. Seimei Handan 999.0

- Time Limit: 2 sec

Problem Statement

Mankind has removed every inefficiency from the world, not only from the earth but also from the entire universe. One of the biggest achievements is the reformation of alphabet systems. As a result, human names are represented by a sequence of integers, not even Unicode codepoints. People call each other using integer sequences. How efficient it is!

Some things change, and some things stay the same. The love that parents have for their children is one thing that has remained unchanged. When parents have new twins, they give their name that maximizes the Efficiency Number so that these babies have a wonderful and efficient life.

Let $A = (a_1, \ldots, a_N)$ and $B = (b_1, \ldots, b_M)$ be the names of twins. For simplicity and efficiency, we assume 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 Efficiency Number is defined to be the number of bijections, i.e. one-to-one invertible functions, from $1, \ldots, L$ to $1, \ldots, L$ that satisfies the following condition.

Condition: Let f denote the bijection. 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.

This procedure is called Seimei Handan 999.0, which is named after Seimei Handan, an unscientific fortune telling in Japan. In ancient Japan, there were programs that did Seimei Handan and some of them were even available via the Internet (ancient computer network system of the era).

You, a programmer, have got an idea to implement Seimei Handan 999.0 as a program and provide it via the EfficientNet (modern computer network system) to make the naming process more efficient (and get advertising revenue). Again, mankind has removed every inefficiency from the world, but your task is to write a program that calculates Efficiency Number. Glory to mankind!

Since the answer may be huge, print the answer modulo $998{,}244{,}353$.

Input

The first line consists of 3 integers, the length of longer name N ($1 \leq N \leq 300{,}000$), the length of shorter name M ($1 \leq M \leq N$), and the biggest integer allowed to use in names L ($1 \leq L \leq 300{,}000$). The second line consists of N positive integers that represent the longer name A. The third line consists of M positive integers that represent the shorter name B.

Output

Output Efficiency Number between A and B modulo $998{,}244{,}353$ in a line.

Sample Input 1

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

Sample Output 1

```
2
```

Sample Input 2

```
1 1 5
1
1
```

Sample Output 2

```
24
```

Sample Input 3

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

Sample Output 3

```
4
```

Sample Input 4	Sample Output 4
22 2 15 2 9 1 2 7 11 5 12 7 11 9 10 6 13 13 4 2 14 7 15 7 8 1 2	520561546

Problem G. Avoid bombings

- Time Limit: 2 sec

Problem Statement

You are developing a video game of avoiding bombings. A player initially stands at the origin of an infinite two-dimensional plane, and the player can move in arbitrary directions at up to the speed limit specified in this game. Then there will be bombings N times. The i-th bombing occurs at time T_i and attacks everywhere on the plane except for a *safety zone*. The shape of a safety zone may be different per bombing but is always a convex polygon on the plane. To avoid a bombing, when it occurs the player must be inside its safety zone. Here, the border of a safety zone is also considered as a part of the safety zone. The player clears this game if they successfully avoid all the N bombings. The player can know the information of all the N bombings prior to the beginning of this game.

You have already decided the shape of the safety zones of all the N bombings. The remaining task is to decide the limit of the speed that a player can move at. Of course, the faster the player can move, the easier the game is. On the other hand, the limit of the speed is too slow, it may become impossible to clear this game. In order to adjust the difficulty of this game, please calculate the minimum limit of the player's speed that the player requires to clear the game.

Input

The input consists of multiple independent test cases given in the following format.

C
$\mathtt{case}_1$
$\mathtt{case}_2$
$\vdots$
$\mathtt{case}_C$

The first line consists of an integer C ($1 \leq C \leq 20$) representing the number of test cases in the input. Then each of the C test cases follows in the following format.

N
T_1 M_1
$x_{1,1}$ $y_{1,1}$
$x_{1,2}$ $y_{1,2}$
$\vdots$
x_{1,M_1} y_{1,M_1}
T_2 M_2
$x_{2,1}$ $y_{2,1}$
$x_{2,2}$ $y_{2,2}$
$\vdots$
x_{2,M_2} y_{2,M_2}
$\vdots$
T_N M_N
$x_{N,1}$ $y_{N,1}$
$x_{N,2}$ $y_{N,2}$
$\vdots$
x_{N,M_N} y_{N,M_N}

Here, N is the number of bombings in this test case ($1 \leq N \leq 20$). The information of the i-th bombing consists of T_i representing the time that the bombing occurs ($1 \leq T_i \leq 100$), M_i representing the number of vertices of the polygon as the safety zone for the i-th bombing ($3 \leq M_i \leq 20$), and $(x_{i,j}, y_{i,j})$ representing the coordinates of the j-th vertex of the i-th polygon ($-100 \leq x_{i,j} \leq 100$, $-100 \leq y_{i,j} \leq 100$). The vertices of a polygon 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 $T_1 < T_2 < \ldots < T_N$.

All values in the input are integers.

Output

For each test case, output in a line a single number, which is the minimum limit of the player's speed that the player requires to clear the game. In particular, if the player can avoid all the bombings even without moving, the answer is 0. The output must not contain an error greater than 10^{-5}.

Sample Input	Sample Output
3 2 10 3 3 4 7 4 3 8 15 4 0 8 8 0 16 8 8 16 3 1 4 0 0 10 0 10 10 0 10 2 4 0 0 -10 0 -10 -10 0 -10 3 4 10 10 -10 10 -10 -10 10 -10 3 10 3 1 1 2 1 1 2 15 3 2 1 2 2 1 2 30 3 3 3 5 3 5 4	0.5 0 0.141421356

Problem H. Sloppy city planning

- Time Limit: 2 sec

Problem Statement

Insane Connection Plan City 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 where 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 minimal total cost to achieve it. Under the constraints of this problem, it can be proven that a solution always exists.

Input

The input consists of a single test case of the following format.

$$
\begin{array}{l}
N \\
c_{1,2}\ c_{1,3}\ \cdots\ c_{1,N} \\
c_{2,3}\ c_{2,4}\ \cdots\ c_{2,N} \\
\vdots \\
c_{N-1,N}
\end{array}
$$

The first line consists of an integer N between 3 and 3,000, 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 between -10^9 and 10^9, inclusive. For each i, j $(1 \leq i < j \leq N)$, $c_{i,j}$ represents the information of the road between towns i and j. Here, 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 in a line a single integer, which is the minimum total cost of roads of which you have to reverse the directions.

Sample Input 1

```
7
-18 -77 -47 -95 84 -23
54 -60 96 43 83
-32 67 27 13
72 97 57
66 -30
-24
```

Sample Output 1

```
59
```

Sample Input 2

```
7
-18 -77 -47 -95 84 -23
54 -60 96 43 83
32 67 -27 13
72 97 57
66 -30
-24
```

Sample Output 2

```
0
```

In the first example, the optional solution is to reverse the directions of a road between towns 3 and 4 and another

road between towns 3 and 6, where the total cost is $|c_{3,4}| + |c_{3,6}| = |-32| + |27| = 59$. After reversing the directions of these two roads, the direction of every road becomes the same as the second example.

Problem I. $N-1$ Truths and a Lie

- Time Limit: 2 sec

Problem Statement

You are playing "$N-1$ Truths and a Lie" game. In this game, N people say facts about some topic. Each of the $N-1$ people says a true fact but the other person says a lie. You win if you identify who is a 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 a specific 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 a liar among N people from the facts they said.

Input

The first line consists of two integers, the number N ($2 \leq N \leq 200{,}000$) of people and the number K ($3 \leq K \leq 200{,}000$) of mountains. The following N lines represent facts that N people say. The i-th line contains three integers a_i ($1 \leq a_i \leq K$), b_i ($1 \leq b_i \leq K$), and x_i ($1 \leq x_i \leq 10^9$), which represent the i-th person says mountain a_i is x meters higher than mountain b_i. You can assume $a_i \neq b_i$. Also, you can assume $(a_i, b_i) \neq (a_j, b_j)$ and $(a_i, b_i) \neq (b_j, a_j)$ hold for all $1 \leq i < j \leq N$. The input is consistent with the situation where there is exactly one liar.

Output

Output in a line a single integer i, where the i-th person is a liar.

Sample Input 1

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

Sample Output 1

```
3
```

Sample Input 2

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

Sample Output 2

```
6
```

Problem J. Most distant point from the stations

- Time Limit: 2 sec

Problem Statement

JAG-island is a rectangular island on the xy-plane. The 4 coastlines of JAG-island are parallel to the x-axis or the y-axis. The left-bottom and right-top corners of JAG-island are located at $(0, 0)$ and (W, H), respectively.

There are N stations in JAG-island. The i-th station is located in (x_i, y_i). You are interested in the coordinates that maximize the (Euclidean) distance to the nearest station. Find the distance from such a coordinate to its nearest station. In other words, calculate the value of

$$\max_{0 \leq x \leq W} \max_{0 \leq y \leq H} \min_{i} \sqrt{(x - x_i)^2 + (y - y_i)^2}.$$

Input

N W H
x_1 y_1
$\vdots$
x_N y_N

The first line of the input consists of three integers, the number N ($1 \leq N \leq 2{,}000$) of stations, the width W and the height H ($1 \leq W, H \leq 1{,}000$) of JAG-island.

The i-th of the following N lines has two integers x_i and y_i ($0 \leq x_i \leq W, 0 \leq y_i \leq H$), which represent the coordinate of the i-th station. The coordinates of stations are distinct, i.e. $(x_i, y_i) \neq (x_j, y_j)$ for any $i, j (i \neq j)$.

Output

Print the answer in a line. Absolute or relative errors less than 10^{-6} are permissible.

Sample Input 1	Sample Output 1
1 1 1 0 0	1.41421356237

Sample Input 2	Sample Output 2
1 4 4 2 2	2.82842712475

Sample Input 3	Sample Output 3
2 6 6 3 1 3 5	3.60555127546

Sample Input 4	Sample Output 4
4 10 10 1 1 1 9 9 1 9 9	5.65685424949

Sample Input 5	Sample Output 5
9 10 10 1 1 1 5 1 9 5 1 5 5 5 9 9 1 9 5 9 9	2.82842712475

Problem K. Sort Compressed Strings

- Time Limit: 4 sec

Problem Statement

There are N strings $s_1, s_2, \ldots, s_N$ which are compressed by rules below. Decompress these strings, and sort them in lexicographical order. Since the length of the decompressed string is very long, you have to answer the permutation $P = (p_1, p_2, \ldots, p_N)$, which is the permutation of $(1, 2, \ldots, N)$ and satisfies $s_{p_1} \leq s_{p_2} \leq \cdots \leq s_{p_N}$. If two decompressed strings s_i, s_j $(i < j)$ are the same, i must appear before j in P.

We compress m-times repetitions of a non-empty sequence `'seq'` into `'m(seq)'`, where m is a natural number greater than or equal to two and the length of `'seq'` is at least one. When `'seq'` consists of just one letter `'c'`, we may omit parentheses and write `'mc'`. Formally, a compressed string is defined by following BNF.

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

For example,

```
'JAGJAGJAGICPCICPCJAGJAGJAGICPCICPCXXXXXXXXXX'
```

can be compressed into:

```
'3(JAG)ICPCICPCJAGJAGJAGICPCICPCXXXXXXXXXX'
```

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

```
'3(JAG)2(ICPC)3(JAG)2(ICPC)10X'
```

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

```
'2(3(JAG)2(ICPC))10X'
```

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

Input

The input consists of a single test case of the following format.

$$
\begin{aligned}
&N \\
&s_1 \\
&s_2 \\
&\vdots \\
&s_N
\end{aligned}
$$

The first line consists of an integer N between 1 and 50, inclusive. This represents the number of compressed strings.

The i-th of the following N lines consists of a compressed string s_i. The length of s_i is at least 1, and less than or equal to 2,000. For each s_i, the length of the decompressed string is less than or equal to 10^{10}.

Output

Output N lines. If the decompressed s_x is the i-th lexicographically smallest string, print x in the i-th line.

Sample Input 1	Sample Output 1
4 3(AB)C A3(BA) ABABABC 2(5A)	4 2 1 3

Sample Input 2	Sample Output 2
5 2(2(2(2A))) 4(4A) 16A 8A8A 8AA	5 1 2 3 4

Problem L. Tree Automorphism

- Time Limit: 2 sec

Problem Statement

There is an undirected tree named T with N vertices numbered 1 to N. The edge between vertices u and v in T is denoted by $\{u, v\}$.

Consider $P = (p_1, p_2, \ldots, p_N)$ as a permutation of $(1, 2, \ldots, N)$. Among the $N!$ possible permutations, compute the number of Ps which satisfy the condition below.

- If T has an edge $\{u, v\}$, then T also has an edge $\{p_u, p_v\}$.

Since the answer may be very large, print the answer modulo $998{,}244{,}353$.

Input

The input consists of a single test case of the following format.

$$
\begin{aligned}
&N \\
&u_1\ v_1 \\
&u_2\ v_2 \\
&\vdots \\
&u_{N-1}\ v_{N-1}
\end{aligned}
$$

The first line consists of an integer N between 1 and 100,000. This represents the number of vertices in the tree T.

The i-th of the following $N-1$ lines consists of two integers between 1 and N. This represents that there is an edge $\{u_i, v_i\}$ in the tree T.

It is guaranteed that the given graph is a tree.

Output

Output the answer modulo $998{,}244{,}353$ in a line.

Sample Input 1	Sample Output 1
4 1 2 1 3 1 4	6

Sample Input 2	Sample Output 2
4 1 2 2 3 3 4	2

Sample Input 3	Sample Output 3
6 1 3 2 3 3 4 4 5 4 6	8