Problem A. Hitoshizuku

Input file: `standard input`
Output file: `standard output`

Even a drifting cloud keeps on going
Without knowing what lies one second away
Into the tomorrow that dulls and fogs over from my anxiety as well.

There are $3n$ girls in a school who wish to form n bands, with each band consisting of three girls. The i-th girl has a charm value a_i, which is a comprehensive value that takes into account various aspects, such as piano skills, stress levels, or even poetry recitation abilities.

To ensure the bands are envy-free and stress-free, each girl reported a value b_i representing the maximum charm value she is willing to share a song with. Specifically, the i-th girl does not want to be in the same band with girls having charm values greater than b_i. Since no girl would envy herself, it is guaranteed that $b_i \geq a_i$.

Can these girls be divided into n bands such that everyone is satisfied with her assignment? Only the charm value should be taken into consideration, as guitar skills can be practiced, and singing abilities can be developed; as long as everyone works together, there are no obstacles.

Input

The input consists of multiple test cases. The first line contains an integer T ($1 \leq T \leq 10^5$), the number of test cases. For each test case:

- The first line contains an integer n ($1 \leq n \leq 10^5$).
- Then $3n$ lines follow, each of which contains two integers a_i and b_i ($1 \leq a_i \leq b_i \leq 10^9$), representing the charm value and the reported maximum charm value limit of the i-th girl, respectively.

It is guaranteed that the sum of n across all test cases does not exceed 10^5.

Output

For each test case:

- If there is a way to divide the girls into n bands, output `Yes` in the first line. In the following n lines, output three numbers indicating the indices of the girls in the same band. Ensure that the output forms a partition of $\{1, 2, \ldots, 3n\}$, with each part containing exactly 3 elements.
- Otherwise, output `No` in one line.

Example

standard input	standard output
2 2 1 2 2 2 2 3 3 5 4 4 4 5 1 1 1 1 1000000000 1000000000 1000000000	Yes 1 2 3 5 4 6 No

Problem B. Guess the Polygon 2

Input file: `standard input`
Output file: `standard output`

This is an interactive problem.

You are given a simple polygon, but before giving you the n vertices of the polygon, Little Q has shuffled them.

You can ask Little Q no more than $n - 2$ queries, each of which consists of three integers a, b ($-10^{15} \le a, b \le 10^{15}$, $a^2 + b^2 > 0$), and c ($-2 \times 10^{18} \le c \le 2 \times 10^{18}$). He will tell you the total length of the points on the line $ax + by + c = 0$ that lie inside or on the boundary of the polygon, which can be represented as $r\sqrt{a^2 + b^2}/s$ with two integers r and s ($r \ge 0, s \ge 1, \gcd(r, s) = 1$). Here $\gcd(r, s)$ is the greatest common divisor of r and s.

You need to find the polygon and then output the vertices in counter-clockwise order. You may start from any vertex.

The polygon and the shuffled order of the vertices are determined before the start of the interaction and do not depend on your queries. In other words, the interactor is not adaptive.

Input

The first line of the input contains an integer T ($1 \le T \le 200$), indicating the number of test cases. For each test case:

The first line contains an integer n ($3 \le n \le 200$), indicating the number of vertices of the polygon.

Then n lines follow, each containing two integers x and y ($0 \le x, y \le 1000$) that give the coordinates (x, y) of the vertices of the polygon in shuffled order.

The polygon is simple, i.e., its vertices are distinct and no two edges of the polygon intersect or touch, except that consecutive edges touch at their common vertex. In addition, no two consecutive edges are collinear.

It is guaranteed that the sum of n over all test cases does not exceed 200.

Interaction Protocol

If you want to ask a query, output one line. First output `?` followed by a space, then output three integers a, b ($-10^{15} \le a, b \le 10^{15}$, $a^2 + b^2 > 0$), and c ($-2 \times 10^{18} \le c \le 2 \times 10^{18}$). After flushing your output, your program should read two integers r and s ($r \ge 0, s \ge 1, \gcd(r, s) = 1$), which means the answer to your query is $r\sqrt{a^2 + b^2}/s$.

If you want to guess the polygon, output $n + 1$ lines. First output `!` in one line, then output n lines, each containing two integers x and y that give the coordinates (x, y) of the vertices of the polygon in counter-clockwise order. You may start from any vertex. After flushing your output, your program should continue processing the next test case, or exit immediately if there are no more test cases. Note that your guess does not count as a query.

To flush your output, you can use:

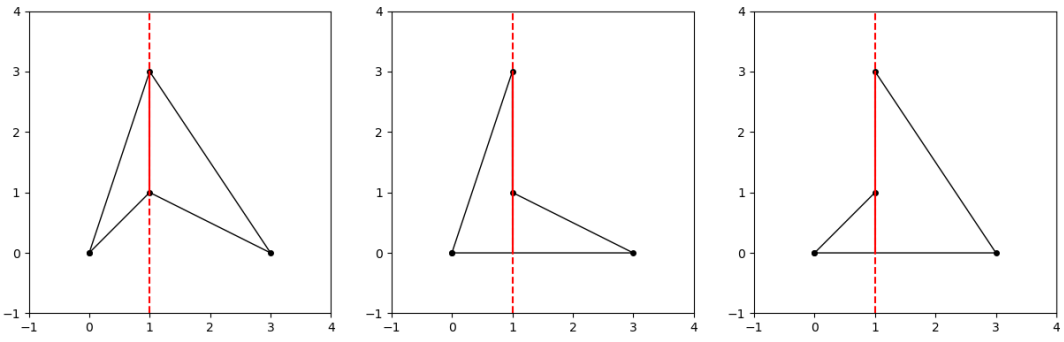
- `fflush(stdout)` (if you use `printf`) or `cout.flush()` (if you use `cout`) in C and C++.
- `System.out.flush()` in Java and Kotlin.
- `sys.stdout.flush()` in Python.

Example

standard input	standard output
2 4 0 0 1 3 1 1 3 0 2 1 3 0 0 999 1000 1000 999 1 1000000000000000000000000	 ? 1 0 -1 ! 1 1 3 0 1 3 0 0 ? 1000000000000 1000000000000 -1999 ! 0 0 1000 999 999 1000

Note

For the first sample case, there are three candidate polygons, shown in the figure below. Only the leftmost one meets the answer 2 to the query $1 \cdot x + 0 \cdot y + (-1) = 0$. The other two have the answer 3 to the query $1 \cdot x + 0 \cdot y + (-1) = 0$.



For the second sample case, the answer to the query $10^{12} \cdot x + 10^{12} \cdot y + (-1999) = 0$ is $\sqrt{2}/10^{12}$, and thus $r = 1$ and $s = 10^{24}$. There is only one candidate polygon.

Note that the order of the vertices shown in the sample case may not be consistent with the actual provided.

The blank lines in the sample case are added for readability. In your output, extra spaces or blank lines will be ignored.

Problem C. Norte da Universidade

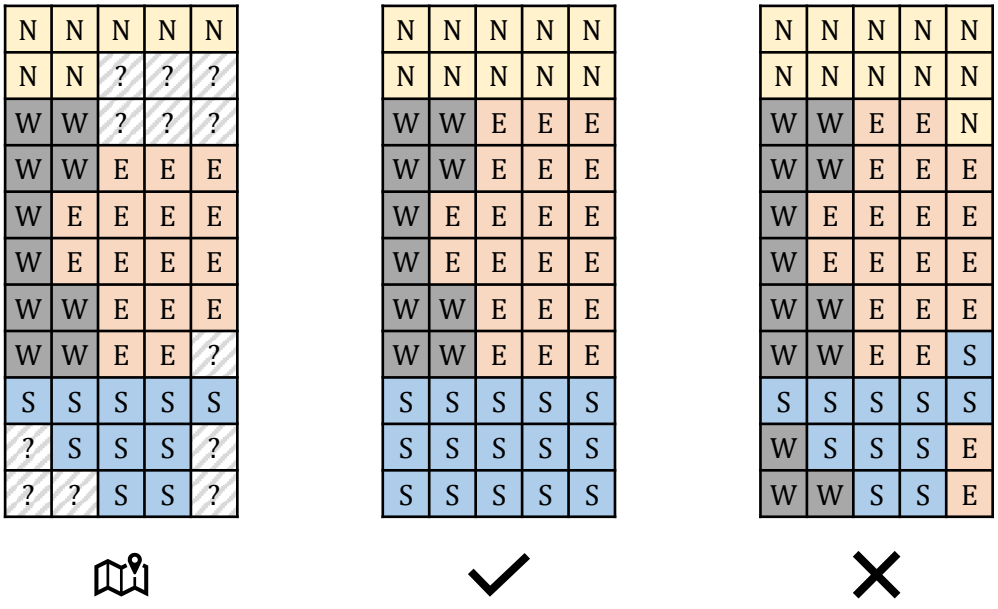
Input file: `standard input`
Output file: `standard output`

Some universities use a coordinate system to name the buildings on their campus: divide the campus into north, south, west, and east regions, then assign an ordinal number to each building to enumerate and identify them. Due to historical or geographical reasons, some regions may not be connected or may not exist at all. The only rules to ensure the system's consistency are:

- To the due north of each building in the north region (N), every building should belong to the north region.
- To the due south of each building in the south region (S), every building should belong to the south region.
- To the due east of each building in the east region (E), every building should belong to the east region.
- To the due west of each building in the west region (W), every building should belong to the west region.

Today, Aki visited a university that applies the same naming rules. After wandering around the campus in the afternoon, he discovered that the buildings in the university are aligned in an $n \times m$ grid, with exactly one building located at each cell. Aki visited some of the buildings, while the others remain unknown. Aki is curious about the possible number of layouts of the university based on his current knowledge. Two layouts are considered different if there exists a building belonging to different regions.

See the following figure illustrating the first sample. For the puzzle shown on the left, the layout in the middle is a valid solution, while the layout on the right is invalid because there are cells violating the rules. For example, the last three cells in the third row violate the third rule, which states that all buildings to the due east of a building in the east region should belong to the east region.



Solve Aki's task. As the answer may be large, only the answer modulo 998244353 is desired.

Input

The input consists of multiple test cases. The first line contains an integer T ($1 \le T \le 100$), the number of test cases. For each test case:

- The first line contains two integers n and m ($1 \le n, m \le 1000$), the sizes of the map. The map is n rows from north to south, and m columns from west to east. The upper left corner is the north west corner.
- The next n lines each contain a string of length m. Each character should be `N`, `S`, `W`, `E`, or `?`.

It's guaranteed that the sum of $n \times m$ does not exceed 2×10^6.

Output

For each test case, print the number, modulo 998244353, of possible layouts in a line.

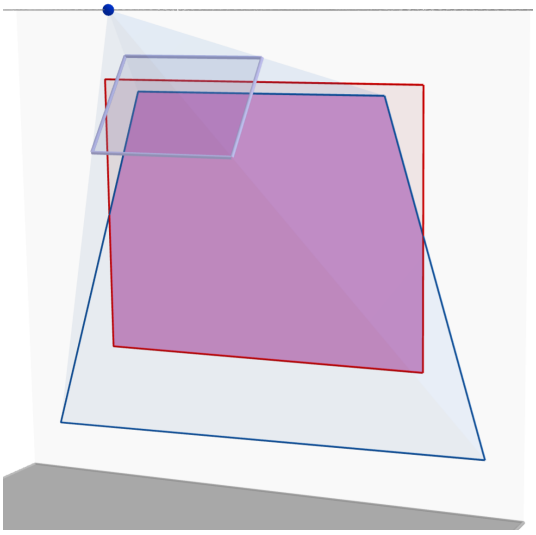
Example

standard input	standard output
5 11 5 NNNNN NN??? WW??? WWEEE WEEEE WEEEE WWEEE WWEE? SSSSS ?SSS? ??SS? 2 7 ??S?N?? ??S?N?? 3 4 W??E WEEE ?E?? 2 2 ?? ?? 3 3 ??? ??? ???	26 1587 18 56 1112

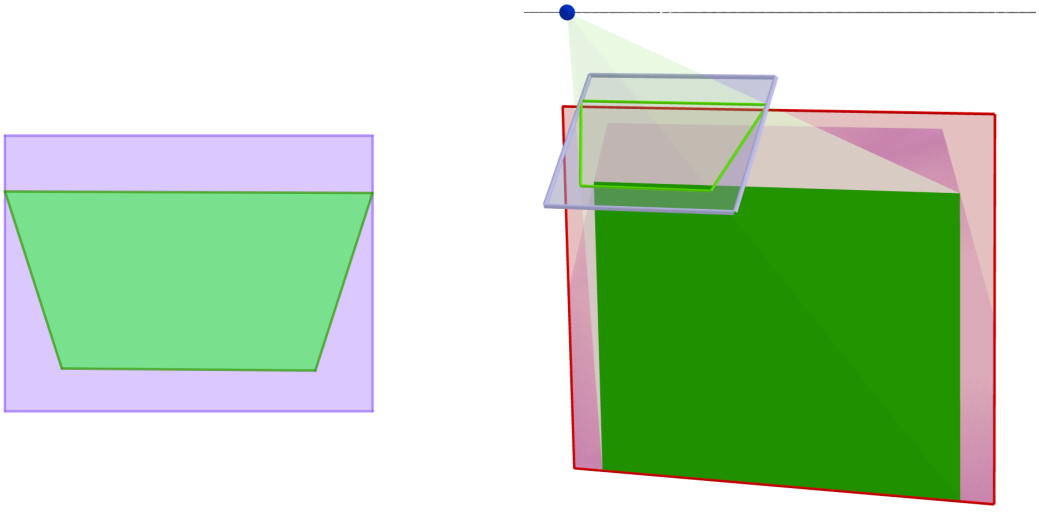
Problem D. Keystone Correction

Input file: `standard input`
Output file: `standard output`

As a huge fan of anime, Haru brought a projector recently to enjoy the home theater with a big screen on the wall. However, the projector is, well, cost-effective: it costs very little and functions like…"just works." It provides limited adjustment functionalities and is screwed to a fixed stand; hence, misalignment often occurs. Fortunately, there is a manual keystone correction feature that can align the corners of the image to the given points, ensuring that the output is as close to the original aspect ratio as possible.



An example of a misaligned projector.



The keystone correction and the final result. This is Haru's solution for the first sample case.

The projector is described as a point light source, and four corners of the lens rectangle of width w and height h in the lens plane. The lens rectangle defines the aspect ratio of the desired picture, denoted as $w : h$ – for example, the aspect ratio after simplification is usually $4 : 3$ or $16 : 9$ in practice. The light beam is bounded by the lens rectangle and finally projects to the screen, which is a rectangle on the wall plane.

Haru's challenge is to obtain four anchor points in the lens rectangle for the keystone correction, so that the corrected projection is a rectangle that keeps the aspect ratio of the lens rectangle with the bottom (with the smallest maximum z coordinate) edge parallel to the xOy plane and that falls entirely inside the screen.

Tired of manually adjusting silhouettes on the wall, Haru decided to write a program to implement a keystone correction algorithm to calculate the maximum area of a good projection rectangle.

Input

The input consists of multiple test cases. The first line contains an integer T ($1 \le T \le 1000$), the number of test cases. For each test case, there are eight lines:

- In the first four lines, the i-th line contains integers x_i, y_i, z_i ($|x_i|, |z_i| \le 100$, $1 \le y_i \le 100$, $\max\{x_1, x_4\} < \min\{x_2, x_3\}$, $\max\{z_1, z_2\} < \min\{z_3, z_4\}$) representing a corner (x_i, y_i, z_i) of the lens rectangle.
- In the next four lines, the i-th line contains integers u_i, v_i, w_i ($|u_i|, |w_i| \le 1000$, $(\max_{j=1}^{4} y_j) < v_i \le 1000$, $u_1 = u_4 < u_2 = u_3$, $v_1 = v_4 < v_2 = v_3$, $w_1 = w_2 < w_3 = w_4$) representing a corner (u_i, v_i, w_i) of the screen rectangle.

Each four points form a rectangle and are listed in the order of the bottom-left, bottom-right, top-right, and top-left corner (see the above constraints for the x and z coordinates). The light source is at $(0, 0, 0)$. Let the distance between the bottom-left and bottom-right corner of the lens be w, and the distance between the bottom-left and top-left corner of the lens be h. Then, $1 \le \frac{w}{h} \le 4$. There are some extra constraints to ensure that the projection is done *roughly* along the front direction (the positive y axis):

- The wall plane is perpendicular to xOy, and the bottom edge of the screen is parallel to xOy.
- The lens rectangle and the point $(0, 0, 0)$ are both on the same side of the wall and are at least 1 unit away from the wall. The distance from the image plane to $(0, 0, 0)$ is at least 1 unit.
- The angle between the bottom edge of the screen and the x axis will not exceed $\frac{\pi}{18} = 10^\circ$. The angle between the bottom edge of the lens and the x axis will not exceed $\frac{\pi}{18} = 10^\circ$.
- The beam projects through the lens to the wall plane, lighting up an area of at most 10^6 square units. The intersection of the screen and the uncorrected projection is at least 1 square unit.

Output

Output the answer as a decimal real number on a line for each test case. The answer will be considered correct if it has an absolute or relative error not exceeding 10^{-6}.

Example

standard input	standard output
2 -20 23 -36 20 23 -36 20 41 -12 -20 41 -12 -50 80 -100 50 79 -100 50 79 -20 -50 80 -20 -1 2 4 1 2 4 1 3 5 -1 3 5 -20 6 0 20 6 0 20 6 20 -20 6 20	5450.0656584647 5.6568542495

Problem E. Corrupted Scoreboard Log

Input file: `standard input`
Output file: `standard output`

You are doing volunteer work for an ancient programming contest. Unfortunately, the contest directors are having difficulty revealing the final standings, as they recorded the final scoreboard on a stone tablet that has been worn away by dirt and sand! Fortunately, there is a rubbing of the stone tablet that survives. It states that there are n teams and m problems in the contest, and each line of text (belonging to the same team) on the scoreboard is preserved. However, all the spaces in the records are missing, and the teams are possibly shuffled during the rescue process — meaning that the teams may not even be listed according to rank.

RANK	TEAM	SCORE	A	B	C	D	E	F	G	H	I	J	K	L
1	Qfljoh Vojwfstjuz	9 935	158 3 tries	26 1 try	31 2 tries	23 1 try	4 tries	43 1 try		241 2 tries	55 1 try	99 1 try		179 1 try
2	Nptdpx Jotujuvuf pg Qiztjdt boe Ufdiopmphz	9 1212	148 2 tries	57 2 tries	39 2 tries	82 1 try	4 tries	43 1 try			52 1 try	249 2 tries	184 2 tries	218 3 tries
3	Utjohivb Vojwfstjuz	9 1218	108 2 tries	14 1 try	54 2 tries	92 2 tries	6 tries	30 2 tries		6 tries	50 2 tries	244 1 try	195 6 tries	171 4 tries
4	Uplzp Jotujuvuf pg Ufdiopmphz	9 1322	124 1 try	26 1 try	54 2 tries	133 1 try	200 2 tries	62 1 try	2 tries		81 1 try	240 1 try		282 5 tries

QfljohVojwfstjuz99351583tries261try312tries231try4tries431try2412tries551try991try1791try
NptdpxJotujuvufpgQiztjdtboeUfdiopmphz912121482tries572tries392tries821try4tries431try521try2492tries1842tries2183tries
UtjohivbVojwfstjuz912181082tries141try542tries922tries6tries302tries6tries502tries2441try1956tries1714tries
UplzpJotujuvufpgUfdiopmphz913221241try261try542tries1331try2002tries621try2tries811try2401try2825tries

An (electronic) scoreboard, along with a possible rubbing of a stone tablet, is shown above. However, the team names will not appear in the input.

More formally, each record consists of $m + 2$ parts. The first two parts represent the total number of problems solved by the team and their total penalty time. The next m parts detail the status of each problem. For each problem:

- If the team did not make any valid submissions for this problem, this part is empty.
- If the team submitted this problem and solved it on the x-th valid attempt in y minutes, it is displayed as "y x `try`". If $x \geq 2$, then `try` becomes `tries`. The penalty for this problem is $y+20(x-1)$.
- If the team made x valid submissions but did not solve it, it is displayed as "x `try`". If $x \geq 2$, then `try` becomes `tries`.

The total penalty time of a team is the sum of the penalty time of each problem they solve. Since this is not an abnormal contest, we can further make the following assumptions:

- The contest lasts for 300 minutes. Teams can solve problems from the 0-th minute to the 299-th minute, but not the 300-th minute.
- There are no more than 13 problems in the contest, and no more than 500 teams are participating.
- Each team will not make more than 100 submissions for any single problem.

You have been assigned the task to cover up the mistake by restoring each line of input text into any possible interpretation of the correct scoreboard by adding spaces. The teams are already waiting in the hall, and the contest directors are counting on you to resolve this crisis. Be quick!

Input

The first line contains two integers n ($1 \le n \le 500$) and m ($1 \le m \le 13$), the number of teams in the contest, and the number of problems in the contest.

The following n lines each contain the record of a team with all spaces removed. It is guaranteed that each record is valid, i.e., it satisfies the conditions described above and has a correct interpretation.

Output

For each team, output one line representing their record. First, output the number of problems they solved and their corresponding penalty time. For each problem:

- If there were no submissions, do not output any characters.
- If there were submissions, output "y x `try(tries)`" or "x `try(tries)`" accordingly.

Separate each part with **a single space**, and **do not print extra spaces**. Ensure that after removing spaces, your record exactly matches the input. If there are multiple solutions, print any.

Examples

standard input
4 12 99351583tries261try312tries231try4tries431try2412tries551try991try1791try 912121482tries572tries392tries821try4tries431try521try2492tries1842tries2183tries 912181082tries141try542tries922tries6tries302tries6tries502tries2441try1956tries1714tries 913221241try261try542tries1331try2002tries621try2tries811try2401try2825tries
standard output

9 935 158 3 tries 26 1 try 31 2 tries 23 1 try 4 tries 43 1 try 241 2 tries 55 1 try
  99 1 try 179 1 try
9 1212 148 2 tries 57 2 tries 39 2 tries 82 1 try 4 tries 43 1 try 52 1 try 249 2 tries
  184 2 tries 218 3 tries
9 1218 108 2 tries 14 1 try 54 2 tries 92 2 tries 6 tries 30 2 tries 6 tries 50 2 tries
  244 1 try 195 6 tries 171 4 tries
9 1322 124 1 try 26 1 try 54 2 tries 133 1 try 200 2 tries 62 1 try 2 tries 81 1 try
  240 1 try 282 5 tries

standard input
5 2 0022tries22tries 12222tries22tries 24422tries22tries 284222tries222tries 2844222tries222tries
standard output
0 0 22 tries 22 tries 1 22 2 2 tries 22 tries 2 44 2 2 tries 2 2 tries 2 84 22 2 tries 22 2 tries 2 844 2 22 tries 2 22 tries

Note

Lines starting with two spaces are considered part of the same line as the previous line. They are only used for presentation purposes in the sample and should not appear in the real output.

Problem F. Boolean Function Reconstruction

Input file: `standard input`
Output file: `standard output`

Given the truth table of a boolean function with n boolean variables as input, construct an expression that satisfies this function. In the expression, you are only allowed to use the logical `and` (`&`) and logical `or` (`|`) operators.

Specifically, a truth table of a boolean function with n boolean variables gives all the 2^n outputs corresponding to the possible values of n input variables. A boolean expression `<expr>` has the following forms:

- `T`, `F`: Represents `True` and `False`.
- `a`, `b`, ..., `z`: Represents one of the variables. The i-th variable is represented by the i-th lowercase letter in alphabetical order.
- `(<expr>&<expr>)`: Represents the logical `and` operation applied to the results of two expressions.
- `(<expr>|<expr>)`: Represents the logical `or` operation applied to the results of two expressions.

The logical `and` operation and the logical `or` operation are defined as two boolean functions below that take two boolean values.

x_1	x_2	x_1&x_2	x_1|x_2
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

Determine whether an expression exists that satisfies the conditions. If such an expression exists, ensure that the number of binary operators (`&` and `|`) does not exceed $2^{n-1} + 10$, and the depth of parentheses nesting does not exceed 100 layers.

It can be proven that if a solution exists, there is always one that meets the constraints of the problem.

Input

The input consists of multiple test cases. The first line contains an integer T ($1 \le T \le 2^{16}$), the number of test cases. For each test case, there are two lines:

- The first line contains an integer n ($1 \le n \le 15$).
- The second line contains a binary string s with length 2^n, indicating the truth table of the given function.

To interpret the input binary string, suppose the i-th variable has a value of x_i. Then, the corresponding function value, $f(x_1, x_2, \ldots, x_n)$, is equal to the $\left(\sum_{i=1}^{n} x_i \cdot 2^{i-1} + 1\right)$-th bit of the string s.

It is guaranteed that the sum of 2^{2n} over all test cases will not exceed 2^{30}.

Output

For each test case:

- Output `Yes` or `No` on the first line to indicate whether an expression satisfying the conditions exists.

- If an expression exists, output the expression on the second line. The expression must strictly adhere to the format given in the problem description, **without adding or omitting parentheses, and without adding extra spaces**.

Example

standard input	standard output
7 2 0001 2 0111 2 1111 3 00010111 1 10 2 0101 5 00000000000000000000000000000001	Yes (a&b) Yes (a|b) Yes T Yes ((a&(b|c))|(b&c)) No Yes a Yes (a&(b&(c&(d&e))))

Note

Below is the truth table interpretation for the fourth sample.

x_3	x_2	x_1	$f(x_1, x_2, x_3)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Problem G. Collatz Conjecture

Input file: `standard input`
Output file: `standard output`

Lynk is studying the Collatz conjecture recently. The Collatz conjecture is a mathematical conjecture that states the following:

- Start with any positive integer n. Then, perform the following steps:
  - If n is even, divide it by 2: $n \to \frac{n}{2}$.
  - If n is odd, multiply it by 3 and add 1: $n \to 3n + 1$.
- Repeat the process indefinitely. The conjecture claims that regardless of the initial value of n, the sequence will eventually reach 1.

Lynk quickly solved the Collatz conjecture and decided to study a variant. In this problem, there are two coprime parameters: A and B.

- Start with any positive integer n. Then, perform the following steps:
  - If n is a multiple of A, divide it by A: $n \to \frac{n}{A}$;
  - Otherwise, add B to it: $n \to n + B$.

He discovered that in this problem, not all numbers will eventually become 1. In his experiments, some numbers grow quickly in the first few steps, and some eventually enter a cycle. Based on his observations, he conjectured that certain integers satisfying specific conditions would return to themselves. For a given positive integer n, he sought to determine whether it would eventually return to itself after a finite number of steps.

Input

The input consists of multiple test cases. The first line contains an integer T ($1 \le T \le 10^5$), the number of test cases. For each test case:

The first line contains three integers A, B, n ($2 \le A \le 10^9, 1 \le B \le 10^9, 1 \le n \le 10^{18}$). It's guaranteed that A and B are coprime.

Output

For each test case:

- Output `Yes` if n will return to itself after a finite number of steps.
- Otherwise, output `No` in a single line.

Example

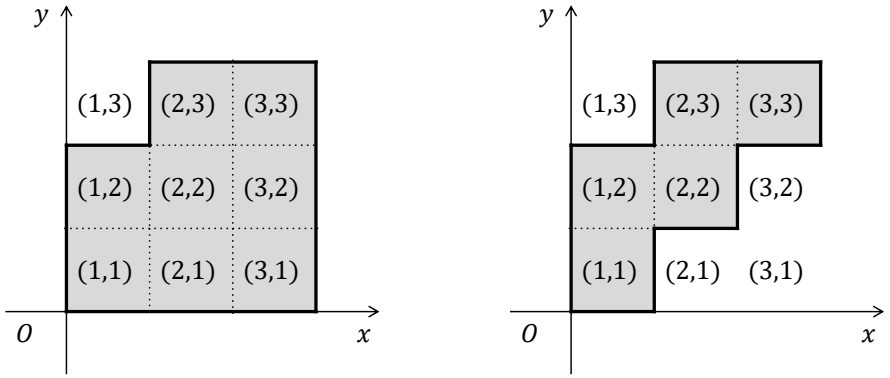
standard input	standard output
7 2 1 1 2 1 2 2 1 3 2 1 100 314 159 265 314 159 2653 314 159 26535	Yes Yes No No Yes Yes No

This page is intentionally left blank.

Problem H. Staircase Museum

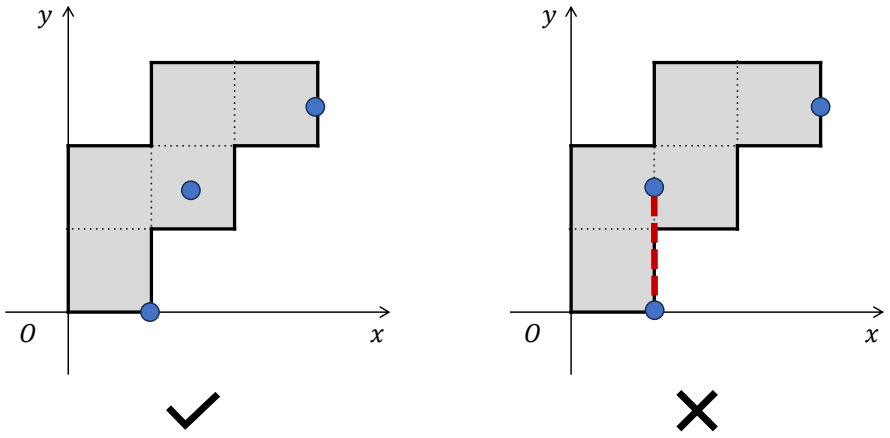
Input file: `standard input`
Output file: `standard output`

A "staircase" museum has been constructed for exhibiting abstract art works. To describe the shape of the museum – a staircase, let the ground be represented as a plane with infinitely many unit square cells with edges aligned to the x and y axes: for every pair of integers (x, y), there is a cell (x, y) that has its bottom-left corner at coordinate $(x-1, y-1)$ and its top-right corner at coordinate (x, y). Then, two given sequences $l_1, l_2, \ldots, l_n$ and $r_1, r_2, \ldots, r_n$, all cells (x, y) that satisfy $1 \le x \le n$ and $l_x \le y \le r_x$ form the interior of the museum.



The staircase museums in the first two sample cases.

A group of abstract artists will enter the museum, and each will find a fixed position either in the interior or on the boundary of the museum. However, none of them would like to see other artists while appreciating the artworks in the museum, as it is already challenging enough to understand abstractions, and seeing others would be too distracting. Here, two artists can see each other if and only if every point of the straight line segment connecting them lies entirely within the interior or on the boundary of the museum.



The left solution shows a possible positioning for the second sample. The right solution is invalid, as two artists can see each other.

As the manager of the museum, it is your responsibility to ensure that the artists can enjoy their visit without being distracted by each other. You need to determine the maximum number of artists such that no pair of artists can see each other within the museum.

Input

The input consists of multiple test cases. The first line contains an integer T ($1 \le T \le 10^5$), the number

of test cases. For each test case:

- The first line contains an integer n ($1 \le n \le 5 \times 10^5$).
- Then n lines follow, the i-th of which contains two integers l_i and r_i ($1 \le l_i \le r_i \le 10^9$). For each $i = 1, 2, \ldots, n-1$, it holds that $l_i \le l_{i+1} \le r_i \le r_{i+1}$, ensuring the museum is a connected staircase.

It is guaranteed that the sum of n over all test cases does not exceed 5×10^5.

Output

For each test case, output a line containing a single integer, indicating the maximum number of artists so that every two artists cannot see each other in the museum.

Example

standard input	standard output
4 3 1 2 1 3 1 3 3 1 2 2 3 3 3 3 1 1 1 3 3 3 4 1 2 2 3 3 4 4 5	2 3 3 4

Problem I. Color-Balanced Tree

Input file: `standard input`
Output file: `standard output`

There are various types of balanced trees, such as height-balanced trees and weight-balanced trees. There are also trees where each vertex is assigned a color, such as red-black trees. How about a color-balanced tree?

A tree containing an even number of vertices is called a color-balanced tree if it satisfies the following properties:

- Each vertex is assigned a color, cyan or white.
- There are an equal number of cyan and white vertices.
- For every simple path in the tree, the absolute difference between the numbers of cyan and white vertices must not exceed 3.

To illustrate these properties, you need to color each vertex either cyan or white to make a given tree a color-balanced tree, if possible. Of course, the total number of vertices of the tree would be even, denoted by $2n$.

Input

The input consists of multiple test cases. The first line contains an integer T ($1 \leq T \leq 10^4$), the number of test cases. For each test case:

- The first line contains an integer n ($1 \leq n \leq 10^5$), denoting that there are $2n$ vertices.
- Then $2n-1$ lines follow, each of which contains two integers u and v ($1 \leq u, v \leq 2n$), indicating an edge connecting the u-th and the v-th vertices.

It is guaranteed that the given edges form a tree, and the sum of n over all test cases does not exceed 10^5.

Output

For each test case,

- If there is a solution, output a line containing $2n$ integers $c_1, c_2, \ldots c_{2n}$ ($c_i \in \{0, 1\}$), separated by single spaces. If $c_i = 0$, the i-th vertex is colored cyan, or otherwise the i-th vertex is colored white.
- Otherwise, print -1 in a line.

Example

standard input	standard output
4 3 1 2 2 3 2 4 4 5 4 6 3 1 2 1 3 1 4 1 5 1 6 4 1 2 2 3 3 4 4 5 5 6 6 7 7 8 5 1 2 1 4 4 3 4 5 5 6 5 8 8 7 8 9 9 10	0 0 0 1 1 1 1 1 1 0 0 0 1 1 1 0 0 0 1 0 1 1 1 0 0 0 1 0 1 0

Problem J. The Mysterious Shop

Input file: standard input
Output file: standard output

There is a mysterious shop outside the Valoran continent. No one knows who operates it, but at the very beginning of every day, the items in the shop are regenerated uniformly and randomly: the shop contains items with weights $1, 2, \ldots, n$. For each weight, there is a probability of $\frac{1}{2}$ that exactly one item of that weight will be in the shop; otherwise, there will be exactly zero items of that weight — in which case, even if an item was present the previous day, it will mysteriously disappear.

Kevin, as a newcomer, is ready to make a big impact here, but first, he needs to buy good equipment. Kevin decides to take a pocket with a capacity of weight n to the mysterious shop every day to be the first customer. He employs the following strategy: examine the items in the shop from heaviest to lightest. For each item, if it fits into the pocket, i.e., the sum of the weights will not exceed n, he adds it to his collection, continuing to the next item until all have been considered. Due to the peculiar payment method in Valoran, it can be assumed that Kevin can always afford to buy all the items he puts into the pocket.

After several days of purchasing, Kevin notices that his daily haul varies. He wants to know whether he is having good or bad luck each day, so he asks you to calculate the possibility that he takes items of total weight i in a day for each possible i.

Input

Input contains an integer n ($1 \le n \le 2 \times 10^5$) in a line, denoting the maximum weight of the items as well as Kevin's pocket.

Output

Print $n + 1$ integers in a line. The i-th integer contains the probability that Kevin takes items of total weight $i - 1$ in a day, multiplied by 2^n, and then taken modulo 998244353. It can be proved that the results before taking modulo are integers.

Examples

standard input	standard output
1	1 1
2	1 1 2
3	1 1 1 5
10	1 1 1 2 2 3 5 13 45 180 771

Note

For the second sample test, there are below four possibilities.

- $\{\}$: Take $\{\}$. The total weight is 0.
- $\{1\}$: Take $\{1\}$. The total weight is 1.
- $\{2\}$: Take $\{2\}$. The total weight is 2.
- $\{1, 2\}$: Take $\{2\}$. The total weight is 2.

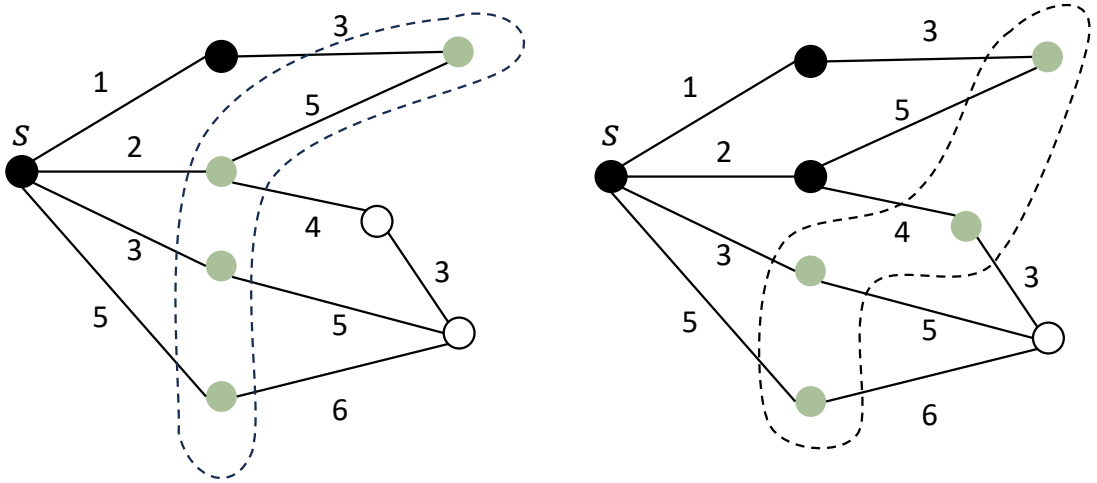
The probabilities are $\frac{1}{4}, \frac{1}{4}, \frac{1}{2}$. After multiplying $2^n = 4$, the output should be $1, 1, 2 \pmod{998244353}$.

This page is intentionally left blank.

Problem K. Exploration Boundary

Input file: `standard input`
Output file: `standard output`

Recently, a paper "Universal Optimality of Dijkstra via Beyond-Worst-Case Heaps" on FOCS raised Fuyu's attention, which talks about the good old shortest-path algorithm achieving optimality on every type of graph by the heap with the working set property. To describe the optimality, there is a concept called *exploration boundary*, that is: pause the Dijkstra's algorithm at some time, the exploration boundary is the frontier of exploration on the graph, or, namely, the vertices that have been updated but with distances not yet determined.



An example of exploration boundaries, where the right boundary is based on the left boundary after determining the vertex with distance 2 from s and updating its neighbors.

To define it more formally, consider the following description of Dijkstra's algorithm. A non-empty set of vertices is called an *exploration boundary* if it can be obtained by extracting the vertices in Q when the algorithm is running to line 6.

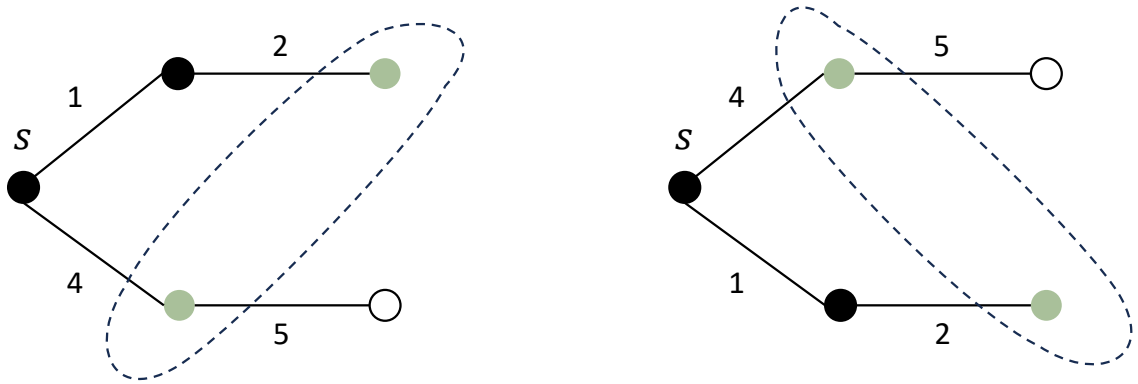
Algorithm 1: Dijkstra's Algorithm

Input: A graph $G = (V, E)$, weights w, a source vertex $s \in V$
Output: Distances D from s

```
1  S ← {} // A set of explored vertices, initially empty.
2  Q ← {} // Priority queue storing (distance, vertex) pairs, sorted by distance.  Initially empty.
3  P ← [∅, ..., ∅] // For each vertex v, a pointer into Q, or ∅, if v has not been inserted yet.
4  D ← [+∞, ..., +∞] // For each vertex, the current best distance from s.
5  P[s] ← Q.Insert((0, s)), D[s] ← 0
6  while |Q| ≠ ∅ do
7      (d_u, u) ← Q.DeleteMin()
8      S ← S ∪ {u}
9      forall (u, v) ∈ E where v ∉ S do
10         if P[v] = ∅ then
11             P[v] ← Q.Insert((+∞, v)) // Insert a dummy value first.
12         D[v] ← min(D[v], d_u + w(u, v)) // Use d_u to update the current best distance to v.
13         Q.DecreaseKey(P[v], (D[v], v)) // For simplicity, call even if D[v] did not change.
14 return D
```

For an undirected graph G, there are many different sets of exploration boundaries if we assign different positive weights w. The paper splits the graph into several boundaries to prove the best complexity of the distance ordering problem on a certain fixed graph is the same as the time complexity of Dijkstra's

algorithm using a working set heap. However, Fuyu noticed that some boundaries cannot be obtained at the same time with the same weight, even if they are disjoint. For example, the following two exploration boundaries are possible for different weights, but they cannot both appear in a single run of Dijkstra's algorithm.



An example of two exploration boundaries that can be obtained by different weights, but cannot both appear in a single run of Dijkstra's algorithm.

Given an undirected graph G and some desired exploration boundaries, let the source vertex $s = 1$, your task is to construct a weight for each edge if it's possible, such that:

- Each edge has a positive integer weight no greater than 10^9.
- No two vertices have the same distance from 1, so that the exploration boundaries can be uniquely determined during the execution of Dijkstra's algorithm.
- All exploration boundaries appeared during the run of Dijkstra's algorithm from 1.

Input

The input consists of multiple test cases. The first line contains an integer T ($1 \le T \le 10^5$), the number of test cases. For each test case:

- The first line contains two integers n and m ($2 \le n \le 2 \times 10^5, 1 \le m \le 2 \times 10^5$), the number of vertices and edges, respectively.
- The next m lines each contain two integers u and v ($1 \le u, v \le n$), describing an undirected edge between vertices u and v.
- The following line contains an integer k ($1 \le k \le 2 \times 10^5$), the number of desired exploration boundaries.
- Each of the next k lines describes a boundary: Each line starts with an integer b_i ($1 \le b_i \le n$), the number of vertices in the i-th boundary, followed by b_i integers denoting the vertices in the boundary. It is guaranteed that b_i vertices are pairwise distinct.

It's guaranteed that the input graph is connected with no self-loops and no duplicate edges, and two boundaries in the same test will not coincide. The sum of n, the sum of m, and the sum of b_i across all test cases do not exceed 10^6.

Output

For each test case:

- Output `Yes` in a single line if it is possible to construct such a weight assignment. In this case, on the next line, output m integers representing the weights for the edges in the order they were provided in the input. Each weight must be a positive integer within $[1, 10^9]$.
- Otherwise, output `No` in a single line.

Example

standard input	standard output
2 8 10 1 2 1 3 1 4 1 5 2 6 3 6 3 7 4 8 5 8 7 8 2 4 3 4 5 6 4 4 5 6 7 5 4 1 2 1 3 2 4 2 5 2 2 3 4 2 2 5	Yes 1 2 3 5 3 5 4 5 6 3 No

This page is intentionally left blank.

Problem L. Shiori

Input file: `standard input`
Output file: `standard output`

What does it mean to be "normal" or "ordinary"?
This ruler in my hand doesn't measure very well at all.

Tomorin, a penguin, has a huge stone collection! To appreciate the stones, she puts some of them on the ground, and there are n piles of stones. Initially, the i-th pile has a_i stones, where a_i is a non-negative number. Tomorin is going to move her stones, and your task is to help her move the stones according to her instructions.

- `1 l r v` : Tomorin rebuilds each stone pile from l to r, with each having v stones. More formally, for every i that $l \leq i \leq r$, $a_i \leftarrow v$.
- `2 l r` : Tomorin watches the piles from l to r. As a careful penguin with a taste for natural numbers, she does not like numbers to be left behind. To solve the problem fairly, let w be the minimum natural number that no piles in $a[l \dots r]$ have exactly w stones; then Tomorin puts an extra w stone from her collections to each of the piles. More formally, let $w = \operatorname{mex}(a_l, a_{l+1}, \cdots, a_r)$; for all $l \leq i \leq r$, $a_i \leftarrow a_i + w$. Here, $\operatorname{mex}(S) = \min(\{0, 1, 2, 3, \dots\} \setminus S)$.
- `3 l r` : Rikki, the enthusiastic breeder, asks for the sum of the number of stones from pile l to r to check if you are ignoring Tomorin. More formally, calculate $\sum_{i=l}^{r} a_i$.

Input

The first line contains two integers n and m ($1 \leq n \leq 5 \times 10^5$, $1 \leq m \leq 5 \times 10^5$), indicating the length of the sequence and the number of operations.

The second line contains n integers $a_1, a_2, \cdots, a_n$ ($0 \leq a_i \leq 5 \times 10^5$), which denote the sequence a.

For the following m lines, the i-th line first contains an integer op_i ($op_i \in \{1, 2, 3\}$) specifying the type of the i-th operation.

- If $op_i = 1$, then three integers l, r, and v follow in the same line ($1 \leq l \leq r \leq n$, $0 \leq v \leq 5 \times 10^5$);
- Otherwise, two integers l and r follow in the same line ($1 \leq l \leq r \leq n$).

Output

For each operation of the third kind, output one line containing the answer.

Example

standard input	standard output
5 8 0 7 2 1 0 1 2 4 0 2 1 3 2 3 4 3 1 3 1 2 3 4 3 1 4 2 1 5 3 2 5	5 11 22