Problem A: Adherence to the Oath

As a contestant of The 2025 ICPC Vietnam National Contest, you are required to take the ICPC contestant's oath:

> ICPC contestant's oath
>
> I, as a contestant of the International Collegiate Programming Contest, and on behalf of my team, do hereby pledge:
>
> 1. `Uphold integrity and ethics throughout the contest.`
> 2. `Do not seek or receive external help from people, platforms, tools or AI.`
> 3. `Follow all ICPC rules and guidelines, accept decisions made by organizers and judges as final.`
> 4. `Show good sportmanship and treat competitors, volunteers, staff and judges with respect.`
> 5. `Compete with creativity and teamwork, honor the contest spirit and pursue excellence.`
>
> We make this pledge to honor our university, our team, and the global community of competitive programmers.

In this problem, you get a single integer p, you need to state the p-th item in the ICPC contestant's oath.

Input

The input consists of a single integer p ($1 \leq p \leq 5$).

Output

Print out the p-th item in the ICPC contestant's oath.

Sample Input 1	Sample Output 1
1	`Uphold integrity and ethics throughout the contest.`

This page is intentionally left blank.

Problem B: Blowing Balloons

In the International Collegiate Programming Contest (ICPC), each problem has its own balloon color: problem A might be red, B blue, C yellow, and so on. Whenever a team solves a problem, a volunteer goes to the balloon area to grab that color and rush it to the team's table. Today the balloon area is coordinated by Paul. His station already has a mix of balloons – some partly inflated from earlier prep, others still flat and waiting for air. There are n balloons, numbered 1 to n. Balloon i initially contains v_i liters of air. From experience, Paul can estimate when each balloon will be needed: balloon i will be dispatched t_i seconds after the contest starts.

The contest has just begun, and Paul may start blowing now. At any second, he can blow exactly 1 liter of air into exactly one balloon. As balloon i is dispatched t_i seconds after the contest starts, it may be blown only at times strictly before t_i seconds. Once a balloon is taken, it cannot be blown anymore.

Paul wants big balloons because they make better photos and happier teams. He defines *beauty score* of a balloon be the square of its final amount of air (in liters). In other words, if a balloon's final amount of air is x liters, its beauty score equals x^2. For each balloon, the final amount of air is its initial air plus the amount Paul blows into it before it is dispatched. The total beauty score is the sum of the beauty scores of all balloons.

Your task is to help Paul decide how to allocate his blowing over time so that the total beauty score is as large as possible.

Input

Each test contains multiple test cases. The first line contains the number of test cases τ ($1 \leq \tau \leq 10^4$). The description of the test cases follows.

- The first line contains an integer n ($1 \leq n \leq 2 \cdot 10^5$) — the number of balloons.
- The i-th line of the next n lines contains two integers t_i and v_i ($1 \leq t_i \leq 10^9; 0 \leq v_i \leq 10^6$) — the dispatch time (seconds after the contest starts) and the initial liters of air for balloon i.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, print a single integer — the maximum total beauty score.

Sample Input 1	Sample Output 1
2 3 3 2 1 4 2 3 2 5 0 1 3	50 34

Sample Explanation

In the first test case, one optimal schedule is:

- blow balloon 2 for 1 second, adding 1 liter;
- blow balloon 1 for 2 seconds, adding 2 liters.

The final amounts of air in all balloons are $[4, 5, 3]$ (liters), so the total beauty score is $4^2 + 5^2 + 3^2 = 16 + 25 + 9 = 50$.

It can be shown that no other schedule yields a larger total beauty score.

Problem C: Counting Contours

Please refer to the judging system for the colored illustrations.

A recursive island, is an island that lies within a lake (which may itself be inside an island). A recursive lake, is a lake that lies within an island (which may itself be inside a lake).

Bash is fascinated with this idea. He wants to create some drawings with islands and lakes, where some can potentially be recursive.

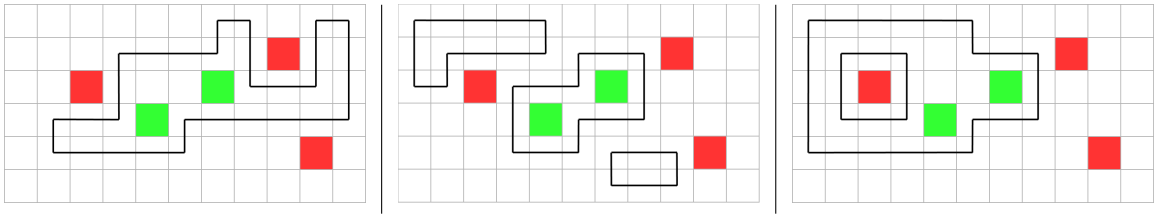
Bash has a grid with R rows and C columns, divided into $R \times C$ cells. Bash then colors some cells red, and some cells green.

Now bash wants to draw some non-intersecting *contours* on this grid. A contour is a border between land and water, and must satisfy the following conditions:

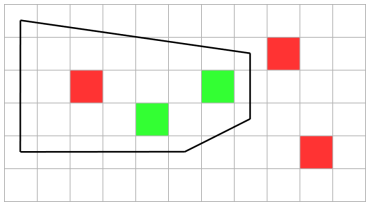
- It must be a polygon, with all vertices being centers of some non-colored grid cells.
- All edges are axis-aligned.
- It doesn't self intersect.
- No edge intersects with any colored cells.

Note that it is possible for one contour to be completely inside another. However, no point can lie on the borders of more than one contours.

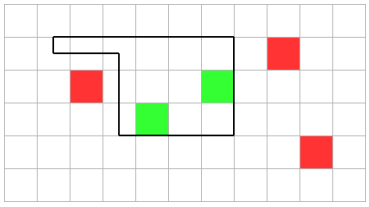
Here are some examples of **valid** way to draw contours:



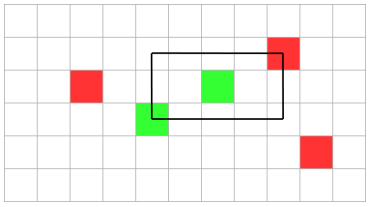
Here are some examples of **invalid** way to draw contours:



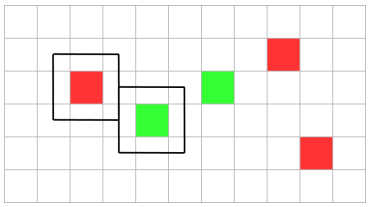
Some edges are not axis-aligned.



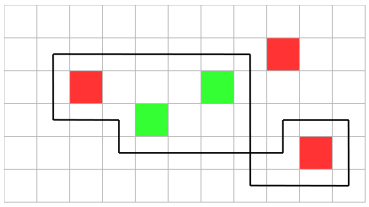
Some vertices are not grid cells' centers.



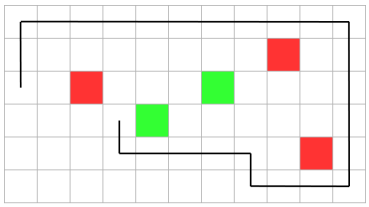
A contour touches colored cells.



Two contours intersect each others.



A contour intersects itself.



This is not a polygon.

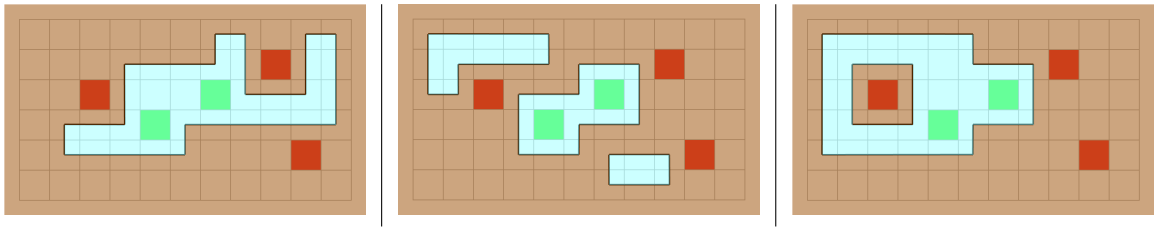
After drawing all contours, Bash wants to draw islands and lakes in between the contours such that:

- For each contour, the parts inside and outside of it must be different (one must be water and the other must be land).
- The parts touching the border of the grid must be land.

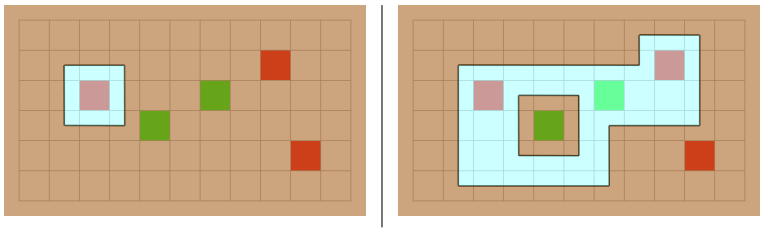
A drawing is considered **valid** iff:

- All red cells are land.
- All green cells are water.

Here are some **valid** drawings after filling in land and water:



Here are some **invalid** drawings after filling in land and water:



Please count the number of **valid** drawings. Two drawings are considered different iff there exists some part of the grid that is land in one way, and is water in the other.

Input

Each test contains multiple test cases. The first line contains the number of test cases τ ($1 \leq \tau \leq 10^4$). The description of the test cases follows.

- The first line contains three integers: R, C and k ($1 \leq R \leq 12$; $1 \leq C \leq 100$; $0 \leq k \leq R \cdot C$), where k is the number of colored cells.
- In the next k lines, each line contains two integers r, c and a string s ($1 \leq r \leq R$; $1 \leq c \leq C$, s is either `red` or `green`) – meaning that the cell (r, c) is colored by s. It is guaranteed that all k cells are distinct.

It is guaranteed that the sum of $R \cdot C$ over all test cases does not exceed 1200.

Output

For each test case, print one line containing the number of valid drawings, modulo $10^9 + 7$.

Sample Input 1	Sample Output 1
2 3 3 1 2 2 green 3 3 1 2 2 red	1 1

Problem D: Difficulty Distance

Mimi, a rhythm game released in 201x, has been attracting more and more players - from young teenagers to adults. As a social-based game, it requires players to join in pairs and choose the same song to play together (note that in this fictional version of the game, each song only has only one level of difficulty).

At a certain arcade, there is a club of n Mimi players who are about to participate in a practice session. The i-th player has a preferred song difficulty level a_i, which also represents their **skill level**. No two players have the same skill level.

The team leader wants to select a subset of players and organize them into pairs such that the difference in skill levels between the two players in each pair is at most 1. Formally, suppose that $x_0(i)$, $x_1(i)$ are the indices of the players in the i-th pair; then it must hold that $\left|a_{x_0(i)} - a_{x_1(i)}\right| \leq 1$.

The team leader aims to form as many pairs of players as possible for the practice session. Your task is to count the number of ways to form the maximum number of pairs. Two ways are considered different iff there exists at least one pairs of players that appear in one way but not in the other.

Input

Each test contains multiple test cases. The first line contains the number of test cases τ ($1 \leq \tau \leq 2 \cdot 10^5$). The description of the test cases follows.

- The first line contains an integer n $(1 \leq n \leq 2 \cdot 10^5)$ — the number of players in the club.
- The second line contains n integers $a_1, a_2, \ldots, a_n$ $(1 \leq a_i \leq 10^9)$ — the skill level of the players. It is guaranteed that all skill levels are distinct.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output a single integer — the number of ways to form the maximum number of pairs, modulo $998\,244\,353$.

Sample Input 1	Sample Output 1
3 6 2 4 5 7 9 10 6 1 3 4 9 8 5 2 1 100	1 2 1

Sample Explanation

In the first test case, one can select players with skill levels 4, 5, 9 and 10 and form two pairs: $(4, 5)$ and $(9, 10)$. It can be shown that this is the only way to form two pairs, and it is impossible to form more than two pairs.

In the second test case, there are 2 ways to form two pairs of players:

- $[(3, 4), (8, 9)]$
- $[(4, 5), (8, 9)]$

In the third test case, it is impossible to form any pair of players, and the number of ways to form zero pairs is 1.

Problem E: Extended Fermat Theorem

In number theory, Fermat's Last Theorem (sometimes called Fermat's conjecture) states that no three positive integers a, b and c satisfy the equation $a^n + b^n = c^n$ for any integer value of n greater than 2. This problem had been unsolved for 358 years until the first successful proof was released in 1994 by Andrew Wiles.

In this problem, we are going to solve an extended version of the Fermat's Last Theorem:

Given a sequence of $n + 1$ **distinct primes** $a_1, a_2, \ldots, a_n, a_{n+1}$ and an integer p. You need to find $n + 1$ **positive** integers $x_1, x_2, \ldots, x_n, x_{n+1}$ such that:

- $x_i < p$,
- $\sum_{i=1}^{n} x_i^{a_i} \equiv x_{n+1}^{a_{n+1}} \pmod{p}$.

Input

Each test contains multiple test cases. The first line contains the number of test cases τ ($1 \leq \tau \leq 100$). The description of the test cases follows.

- The first line contains two integers n and p ($1 \leq n \leq 10^5$, $10^9 \leq p \leq 10^{18}$).
- The second line contains $n + 1$ integers $a_1, a_2, \ldots, a_n, a_{n+1}$ ($1 \leq a_i \leq 10^9$).

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

Output

For each test case, print one line containing $n + 1$ integers $x_1, x_2, \ldots, x_n, x_{n+1}$ ($0 < x_i < p$).

If there is no solution, print `NO SOLUTION` instead.

If there are multiple solutions, you can output any of them.

Sample Input 1	Sample Output 1
1 4 1000000009 3 5 7 11 2	1 1 1 1 2

Sample Explanation

$1^3 + 1^5 + 1^7 + 1^{11} \equiv 2^2 \pmod{1000000009}$

This page is intentionally left blank.

Problem F: Feast of Delicacy

After the great exodus from the Tower of God, the maidens' party wants to celebrate with a huge feast. And the one in charge of it is none other than the party's self-proclaimed leader — Hameln the Hero!

In a hurry, Hameln has found n pieces of spice and put them into a conveyor belt in order, which leads straight to the pot. The i-th spice has a deliciousness of a_i. Hameln takes so long being proud of her handful skill to notice one thing only at the last second — some spices can worsen the feast! In other words, a_i can be negative.

Having only a second to spare, Hameln has devised a brilliant plan to save the celebration:

- First, she chooses 2 numbers l and r ($1 \le l \le r \le n$) and **reverse** the segment in the conveyor belt containing spices $a_l, a_{l+1}, \ldots, a_r$.
- Second, she chooses a number k ($0 \le k \le n$). She only uses k first spices and discard the rest of them. The deliciousness of the feast is the total deliciousness of these k spices used. Note that if Hameln uses 0 spices, the deliciousness is 0.

The plan is almost as sweet as Gretel's cakes, but Hameln's mind is not made for these decisions! Can you help her find the maximum deliciousness of the feast if she proceeds with her plan optimally?

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

The first line contains an integer n ($2 \le n \le 2 \cdot 10^5$) — the number of spices in the conveyor belt.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($-10^9 \le a_i \le 10^9$).

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, print a single number that denotes the maximum deliciousness Hameln can get for the feast.

Sample Input 1	Sample Output 1
3 5 1 3 5 -7 9 4 -7 -1 -4 -6 10 1 -3 5 -7 9 -11 13 -15 17 -19	18 0 24

Sample Explanation

In the first test case, Hameln reverses the segment $[-7, 9]$. The conveyor belt now looks like this: $[1, 3, 5, 9, -7]$. Then she takes the first 4 spices. Thus, the deliciousness is $1+3+5+9 = 18$. It can be proven that the best possible deliciousness is 18.

In the second test case, since all spices have negative deliciousness, she discards all spices. Thus, the deliciousness of the feast is, unfortunately, 0.

Problem G: Goodifying Strings

FWMC receives a string s of length n, consisting of characters $s_0, s_1, \ldots, s_{n-1}$, as a birthday gift. Initially, every character in s is the letter '`A`'.

FWMC then starts playing around with the string. The process can be described as a sequence of q queries, each is one of the following two types:

- `1 l r c` — FWMC assigns all positions i in the range $[l, r]$ to the character c (that is, set $s_i = c$ for all $l \le i \le r$).
- `2 k` — FWMC dislikes long sequences of consecutive identical characters. FWMV considers a string **good** if there is no consecutive substring[1] of more than k identical characters. Considering the current string s, FWMC wants to determine the minimum number of characters that must be deleted to make s a **good** string.

  Note that FWMC just wants to know the minimum number of deleted characters, but does not make any changes to s.

Your task is to help FWMC determine the answer to each query of the second type.

Input

The first line contains two integers n and q ($1 \le n, q \le 300\,000$) — the length of the string and the number of queries.

Each of the next q lines describes a query. The first integer is t ($1 \le t \le 2$) —— the type of the query.

- If $t = 1$, two integers l and r ($0 \le l \le r < n$) and an uppercase Latin letter c are given.
- If $t = 2$, an integer k ($1 \le k \le n$) is given.

Output

For each query of the second type, output a single integer — the minimum number of characters that need to be deleted to make the string **good**.

[1] A string α is a substring of a string β if α can be obtained from β by the deletion of several (possibly, zero or all) characters from the beginning and several (possibly, zero or all) characters from the end.

Sample Input 1	Sample Output 1
11 10 1 0 1 F 1 2 10 C 1 3 8 W 1 9 9 M 1 4 6 M 2 2 2 1 2 5 1 5 7 C 2 1	1 4 0 3

Sample Input 2	Sample Output 2
12 5 1 3 7 G 1 7 11 G 1 4 5 C 1 0 6 C 2 2	8

Sample Explanation

The following is the explanation for the first test case.
In the first five queries, the string changes as follows:

$$\texttt{AAAAAAAAAA} \to \texttt{FFAAAAAAAA}$$
$$\texttt{FFAAAAAAAA} \to \texttt{FFCCCCCCCC}$$
$$\texttt{FFCCCCCCCC} \to \texttt{FFCWWWWWWC}$$
$$\texttt{FFCWWWWWWC} \to \texttt{FFCWWWWWWMC}$$
$$\texttt{FFCWWWWWWMC} \to \texttt{FFCWMMMWWMC}$$

In the sixth query, it is sufficient to delete one character:

$$\texttt{FFCWMMMWWMC} \to \texttt{FFCWMMWWMC}$$

The groups of consecutive identical characters are FF | C | W | M | MM | WW | M | C, none of which contain more than $k = 2$ identical characters, thus the final string is **good**.
In the seventh query, with $k = 1$, one possible solution to delete 4 characters is:

$$\texttt{FFCWMMMWWMC} \to \texttt{FCWMWMC}$$

It can be shown that there exists no solution that delete fewer characters.
In the eighth query, with $k = 5$, the current string s is already a good string, so no deletions are needed.
In the ninth query, the string changes as follows:

$$\texttt{FFCWMMMWWMC} \to \texttt{FFCWMCCCWMC}$$

In the tenth query, with $k = 1$, one possible solution to delete 3 characters is:

$$\texttt{FFCWMCCCWMC} \to \texttt{FCWMCWMC}$$

It can be shown that there exists no solution that delete fewer characters.

Problem H: Harmonic Strings

Today, professor Awk teaches Bash about binary strings and their various beautiful properties. A binary string of length n consists of exactly n characters, which are either 0 or 1.

Given a binary string, we can always decompose it into `runs`, each run is a **maximal** contiguous block of equal characters. For example, the binary string `1110011011000` can be decomposed into 6 runs: `111`, `00`, `11`, `0`, `11`, and `000`. We call a run with only 0 a zero-run, and a run with only 1 a one-run.

For a binary string w, professor Awk is interested in the two following values:

- L_0: the length of the longest zero-run of w,
- S_1: the length of the shortest one-run of w.

Note that if w doesn't have any zero-runs, L_0 is set to 0. If w doesn't have any one-runs, S_1 is set to 0.

Professor Awk then asks Bash to help him find a Harmonic String w, which is a binary string that satisfies all the following conditions:

- The number of 1 in w must be exactly k.
- The value $L_0 \cdot (n+1) - S_1$ is minimal.

Input

The first line of the input contains τ ($1 \le \tau \le 10^5$) – the number of test cases. τ test cases follow, each is presented by a single line with two integers n and k ($1 \le n \le 10^5, 0 \le k \le n$).

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

Output

For each test case, print a Harmonic string. If there are multiple optimal solutions, you can print any of them.

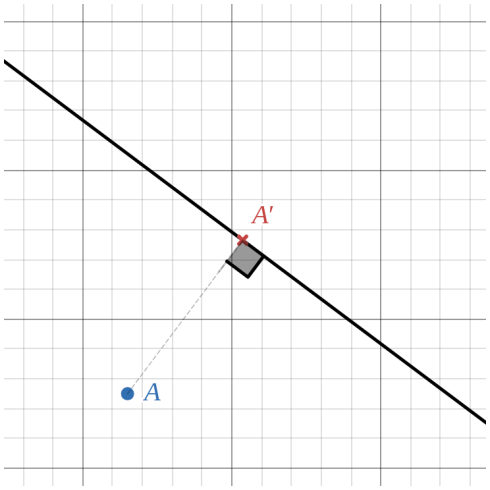
Sample Input 1	Sample Output 1
2 3 2 4 2	110 0110

This page is intentionally left blank.

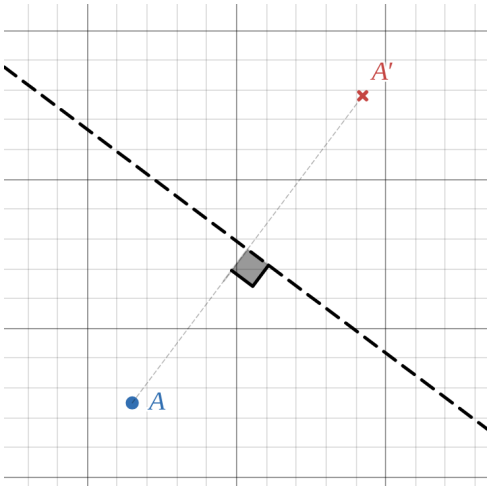
Problem I: Impulsive Particle Experiment

The scientists have discovered a new kind of particle that can revolutionize transportation. In the ideal environment, if there is a nearby flat object, the particle will shoot itself in the **direction to the closest point** on the object, almost instantly. After that, the particle will be stabilized and rest. The scientists call the flat object *reflector*, and based on the material of the *reflector*, the particle can rest at a different position:

- if the *reflector* is **gold**, the particle's resting position will rest on the *reflector* itself;
- if the *reflector* is **diamond**, the particle's resting position will be directly on the opposite side of the *reflector*, at the same perpendicular distance from it.



Shoot the particle at point A to a **gold** *reflector*



Shoot the particle at point A to a **diamond** *reflector*

The scientists realizes that this can make teleportation become a reality. However, more testing is required to ensure everyone's safety and also to optimize the cost for such a travelling method. So several experiments are conducted.

The experiment's environment can be seen as an Oxy plane. The scientists have prepared n *reflectors*, the i-th *reflector* is made of the material m_i (which is either **gold** or **diamond**) and it is model as a line of form $a_i x + b_i y + c_i = 0$. Then the scientists perform q actions of the following types:

- $1\ j\ m\ a\ b\ c$ – replace the j-th *reflector* with another one, made of the material m, on the line $ax + by + c = 0$.
- $2\ x\ y\ l\ r$ – the scientists place a particle at the position (x, y). For each i from l to r, the scientists shoot the particle to the i-th reflector, moving the particle to a new resting position.

For each action of the second type, please help the scientists find the final resting position of the particle.

Input

The first line contains two integers n and q ($1 \le n, q \le 2 \cdot 10^5$) – the number of *reflectors* and the number of actions to perform.

The i-th of the next n lines contains a character m_i and three integers a_i, b_i, c_i ($m_i \in \{\texttt{G}, \texttt{D}\}$, $|a_i|, |b_i|, |c_i| \le 10^4$, $a_i^2 + b_i^2 > 0$) – the material of the i-th *reflector* (`G` for **gold**, and `D` for **diamond**) and the model line coefficients of the i-th *reflector*.

The i-th of the next q lines describes the actions with the following form:

- `1` j m a b c ($1 \le j \le n$, $m \in \{\texttt{G}, \texttt{D}\}$, $|a|, |b|, |c| \le 10^4$, $a^2 + b^2 > 0$) – description of the action type `1`.
- `2` x y l r ($|x|, |y| \le 10^4$, $1 \le l \le r \le n$) – description of the action type `2`.

Output

For each action of the second type, print two integers x', y' ($0 \le x', y' < 998\,244\,353$) – the coordinates of particle's final resting position, modulo $998\,244\,353$.

Formally, let $M = 998,244,353$. It can be shown that every coordinate can be expressed as an irreducible fraction $\frac{\alpha}{\beta}$, where α and β are integers and $\beta \not\equiv 0 \pmod{M}$. Output the integer equal to $\alpha \cdot \beta^{-1} \bmod M$. In other words, output such an integer χ that $0 \le \chi < M$ and $\chi \cdot \beta \equiv \alpha \pmod{M}$.

Sample Input 1	Sample Output 1
2 5 G 1 0 -1 D 0 1 -1 2 3 5 1 1 2 3 5 1 2 1 2 D 1 1 0 2 3 5 1 2 2 6 9 2 2	1 5 1 998244350 998244348 998244352 998244344 998244347

Sample Explanation

In the below illustraion, the connected line represents the **gold** *reflector*, while the dashed-line represents the **diamond** *reflector*.

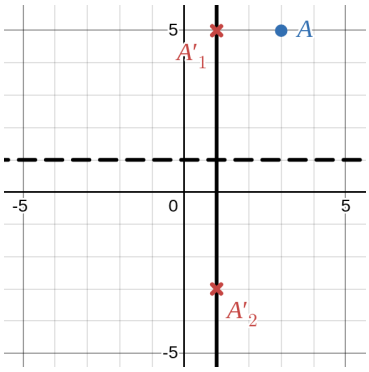


Illustration for the actions 1 and 2.

- The point $A'_1 = (1, 5)$ is the answer for the first action.
- The point $A'_2 = (1, -3)$ is the answer for the second action.

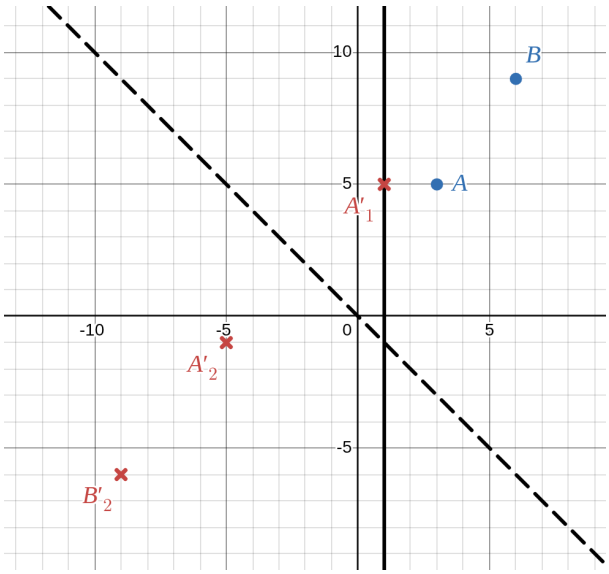


Illustration for the actions 3, 4 and 5.

- The second *reflector* is changed after the 3-rd action.
- The point $A'_2 = (-5, -1)$ is the answer for the 4-th action.
- The point $B'_2 = (-9, -6)$ is the answer for the 5-th action.

This page is intentionally left blank.

Problem J: Jeopardizing Justice

The Shell is a great prison where the members of Adwent were imprisoned. It is said to be somewhere deep underground within this world, but its exact location is unknown.

At the core of The Shell, there are n prison cells. Each cell is equipped with a surveillance camera that must be constantly monitored. There are n monitoring rooms, the i-th room is connected to the cameras of a specific subset of cells. The monitoring rooms and the cells are numbered from 0 to $n-1$.

For security reasons, each room can monitor **exactly one** of the camera it is connected to. The Shell is considered *secure* if and only if there exists a way to assign one camera to each room such that **all** cameras are monitored. It is guaranteed that The Shell is **secure** at the beginning.

After escaping from The Shell, the members of Adwent are now planning to sabotage the security of The Shell. However, due to limited resources, they can only perform at most 50 sabotage actions. In each sabotage action, they may choose a monitoring room i and a cell j such that room i is currently connected to cell j's camera, and delete that connection.

The leader of Adwent wants to determine whether it is possible to perform a sequence of at most 50 sabotage actions such that The Shell is no longer **secure**. If it is possible, you must output one valid sequence of sabotage actions.

Input

Each test contains multiple test cases. The first line contains the number of test cases τ ($1 \leq \tau \leq 50$). The description of the test cases follows.

- The first line contains an integer n ($1 \leq n \leq 50$) — the number of monitoring rooms and cells.
- The i-th of the next n lines contains n integers $a_{i,0}, a_{i,1}, \ldots, a_{i,n-1}$ ($0 \leq a_{i,j} \leq 1$). Here, $a_{i,j} = 1$ means that the i-th monitoring room is connected to the j-th cell's camera; otherwise, $a_{i,j} = 0$.

It is guaranteed that The Shell is **secure** in the initial configuration.

Output

For each test case, if it is impossible to make The Shell insecure using at most 50 sabotage actions, output a single integer -1. Otherwise, output the following:

- The first line contains an integer m ($0 \leq m \leq 50$) — the number of sabotage actions.
- The i-th of the next m lines contains two integers r_i and c_i ($0 \leq r_i, c_i < n$) — the indices of the chosen room and cell for the i-th sabotage action. The connection between the selected room and cell's camera must exist before the action is performed.

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

Sample Explanation

In the first test case, after the sabotage actions, it can be shown that no valid assignment exists that allows all cameras to be monitored simultaneously. Therefore, The Shell is no longer **secure**.

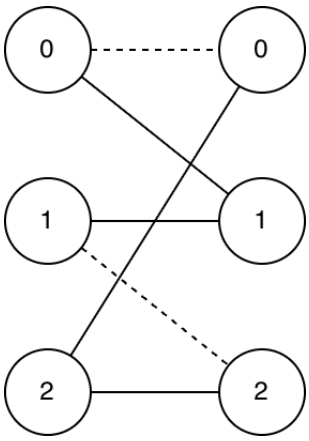


Figure J.1: The following is the illustration for the first test case. Vertices on the left and the right represent monitoring rooms and cells, respectively. Lines represent existing connections, and dashed lines represent deleted connections.

In the second test case, note that the number of sabotage actions does **not** need to be minimized.

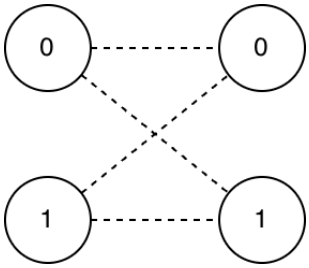


Figure J.2: Illustration of the second test case.

Problem K: Kilo Kilter Kerfuffle

This is an interactive problem.

It is well-known that the precise strength of Earth's gravity varies with location. Alice and Bob live in different countries, and they want to do experiments to verify this fact. First, the friends decide to meet up in Vietnam, and each of them will buy the same scale there. Then they fly back to their countries and prepare some weights, making sure that the scale shows exactly 1KG for each of the weights. Finally, they fly to Vietnam again, and this time compare their weights. Sure enough, when putting one of Bob's weights and one of Alice's weights onto different sides of a balance scale, the scale is not balanced!

After doing the experiment all day, the friends decide to take a break. However, since they do not clean up, when going back, Alice and Bob do not know which weights are whose. They also do not remember the amount of weights each of them bring either, as there are a lot of weights. Now the friends need to clean up and bring home their weights. Here is the summary of what is known by Alice and Bob:

- There are exactly 2^n weights, numbered from 1 to 2^n, indistinguishable from each other;
- Each of the friends brings at least one weight;
- All weights brought by Alice have the same weight;
- All weights brought by Bob have the same weight;
- Alice's weights' weight are different from Bob's weights' weight;
- The number of Alice's weights is **less than or equal to** the number of Bob's weights.

They want to start by figuring out the number of weights each of them bring. To do so, they use the same balance scale. In one operation, they can put some of the weights on each side of the balance scale and observe one of the following results:

- The left side is heavier;
- The right side is heavier;
- The scale is balanced.

Please help the friends find the number of weights brought by Alice, using no more than n^2 operations.

Interaction Protocol

Each test contains multiple test cases. Your program read the number of test cases τ ($1 \le \tau \le 1000$). For each test case, your program interact with the jury's program as follows.

First, your program read the integer n ($1 \le n \le 10$) meaning that the total number of weights the friends have is 2^n.

Then, your program can use the scale as follows:

- On the first line, print ? cnt_l cnt_r ($1 \le cnt_l, cnt_r$; $cnt_l + cnt_r \le 2^n$) – the number of weights to put onto the left side and the right side of the balance scale, respectively;

- On the next line, print cnt_l numbers $l_1, l_2, \ldots, l_{cnt_l}$ ($1 \le l_i \le 2^n$) – the indices of the weights to put on the left side of the balance scale;
- On the next line, print cnt_r numbers $r_1, r_2, \ldots, r_{cnt_r}$ ($1 \le r_i \le 2^n$) – the indices of the weights to put on the right side of the balance scale;

  All $cnt_l + cnt_r$ integers $l_1, l_2, \ldots, l_{cnt_l}, r_1, r_2, \ldots, r_{cnt_r}$ must be distinct.
- Then read a character c ($c \in \{>, =, <\}$) – the balance scale result:
  - If $>$ is read, the left side is heavier;
  - If $<$ is read, the right side is heavier;
  - If $=$ is read, the scale is balanced.

When your program knows k – the number of weights that Alice brings, print ! k, then continue with the next test case, or terminate the program when there are no more test cases. Printing this does not count towards the number of operations limit.

It is guaranteed that the sum of 2^n over all test cases does not exceed 5000.

The jury's program is **not adaptive**. The weight and the owner of each weight is known by the jury's program before the interaction and will not be changed during the interaction process.

After printing each query do not forget to output the end of line and flush[2] the output.

[2]To flush, use:

- `fflush(stdout)` or `cout.flush()` in C++;
- `sys.stdout.flush()` in Python;
- see the documentation for other languages.

Sample Interaction

Stdin	Stdout	Comment
2		The number of test cases
		Test case 1
1		$n = 1$, so there are $2^n = 2^1 = 2$ weights in total
	? 1 1	Put 1 weight onto the left side, and 1 weight onto the right side of the scale
	1	Put the 1-st weight onto the left side.
	2	Put the 2-nd weight onto the right side.
<		The right side is heavier.
	! 1	Answer that Alice brings $k = 1$ weights
		Test case 2
2		$n = 2$, so there are $2^n = 2^2 = 4$ weights in total
	? 2 2	Put 2 weights onto the left side, and 2 weights onto the right side of the scale
	1 4	Put the 1-st and 4-th weights onto the left side.
	2 3	Put the 2-nd and 3-rd weights onto the right side.
<		The right side is heavier.
	? 1 1	Put 1 weight onto the left side, and 1 weight onto the right side of the scale
	1	Put the 1-st weight onto the left side.
	3	Put the 3-rd weight onto the right side.
=		Both sides have the same weight.
	! 1	Answer that Alice brings $k = 1$ weights

The example interaction only shows how your program and the jury's program interact with each other, and does not show how the answer is deduced.

In the first test case, there are 2 weights. Since each friend bring at least 1 weight, Alice must have brought 1 weight.

In the second test case, suppose that one of Alice's weights is 1.01 KG, and one of Bob's weights is about 0.99 KG. Alice brings the 2-nd weight, while Bob brings the 1-st, 3-rd, and 4-th weights:

- In the first operation, the sum of the 1-st and 4-th weights ($0.99 + 0.99 = 1.98$ (KG)) is less than the sum of the 2-nd and 3-rd weights ($1.01 + 0.99 = 2$ (KG)).
- In the second operation, the 1-st weight and the 3-rd weight have the same weight (0.99 (KG)).

The observations result is consistent with the fact that Alice brings only the 2-nd weight.

This page is intentionally left blank.

Problem L: Laning Phase

The year is 2030. Kiaya, one of the most decorated *Leaf of Lemons* players in the world, queues up a ranked match to prepare for the next tournament. On the loading screen, a familiar name appears. Kiaya cannot believe his eyes. Facing him in the top lane is Zeros the GOAT[†], his former teammate, once considered the most promising prospect in the game, who was sadly banned from competitions years ago due to numerous scandals.

[†] *GOAT stands for **G**lobally banned fr**O**m **A**ll compe**T**itions. Not quite what you expected...*

"Let's have a good match, old man" — murmurs Kiaya as he charges his champion into the top lane. Zeros is within his sight, flashing taunting dances as he always did when they were young.

In *Leaf of Lemons*, the first phase of the game is called the *laning phase*, where players go up against their opponent in one of three lanes: top, middle, and bottom. For simplicity, the top lane where Kiaya and Zeros fight can be described as a strip of n squares numbered from 1 to n. Currently, Kiaya's and Zeros' champions are positioned at square x and square y, respectively ($1 \le x < y \le n$). The players take turns alternatively to move, with Kiaya going first.

On his turn, a player at position p must dash his champion by choosing a **non-zero** integer k such that:

- $-a \le k \le a$ for Kiaya, $-b \le k \le b$ for Zeros;
- $1 \le p + k \le n$

After the dash, the mover's new position is $p + k$.

Moreover, since they are both supremely skilled players, if a player dashes his champion into or over his opponent, he will be slain immediately. Therefore, the player who can force his opponent to dash into or over his champion wins the laning phase. More precisely, a player dashes his champion into or over his opponent iff after his turn, Kiaya's current position is greater than or equal to Zeros's.

Given the current state of the top lane, Kiaya wants to know who, if any, will win the laning phase, assuming both players play optimally. Please help him!

Input

The first line of the input contains τ ($1 \le \tau \le 10^5$) – the number of test cases. τ test cases follow, each is presented by a single line with five integers n, x, y, a, b ($2 \le n \le 10^9$, $1 \le x < y \le n$, $1 \le a, b < n$) – the number of squares on the lane, the starting position of Kiaya and Zeros, and maximum number of steps of Kiaya and Zeros in one turn, respectively.

Output

For each test case:

- if Kiaya has a winning strategy, print `Kiaya`.
- if Zeros has a winning strategy, print `Zeros`.
- if neither player has a winning strategy, print `Draw`.

Sample Input 1	Sample Output 1
3 3 2 3 2 1 4 3 4 2 1 999 1 999 1 1	Zeros Kiaya Kiaya

Sample Explanation

In the first test case, Kiaya has to move back one step to square 1. Zeros then wins by moving to square 2, forcing Kiaya to dash into or over him next turn.

In the second test case, Kiaya wins as follows:

- Kiaya moves to square 1.
- Zeros has to move to square 3.
- Kiaya moves to square 2.
- Zeros has to move to square 4.
- Kiaya moves to square 3, forcing Zeros to dash into him next turn.

In the third test case, it can be shown that Kiaya can win by keeping moving forward one step, no matter what Zeros does.