The 4th Universal Cup

Stage 15: Grand Prix of China

February 7-8, 2026

This problem set should contain 12 problems on 27 numbered pages.

Based on

International Collegiate Programming Contest (ICPC)

Hosted by

Prepared by

Problem A
Outstanding Outlines
Time limit: 2 seconds

Cai, the outstanding billionaire, has a very particular obsession: he only feels comfortable when his feet are planted firmly on an edge or vertex of the tiling. Recently, he acquired a collection of identical, high-end triangular floor tiles. He plans to use these tiles to tile a long, straight corridor in his new mansion.

The corridor is an infinite strip on a two-dimensional plane. To ensure the design meets his supreme standards, Cai has dictated a strict tiling rule: every single triangular tile must be placed such that it spans the entire width of the corridor, touching both parallel boundary lines. This creates a continuous, single-row *tessellation* of the strip. One of these tiles must exactly match his favorite reference triangle, and all other tiles must be congruent to it.

Cai is currently standing at a fixed location in the hallway. He is exhausted after a long day of terraforming work and, quite frankly, will not move at all. He wants the tiles to grow automatically across the floor starting from his reference triangle, as long as the resulting pattern follows his rules. Of course, it is only a success if an edge or vertex of the tiles coincides with his current position!

Your task is to determine if it is possible to tile the corridor such that Cai's current position, point D, lies exactly on an edge or a vertex of the *tessellation* containing $\triangle ABC$. The corridor's boundaries are defined as the two unique parallel lines l_1 and l_2 such that l_1 passes through A and B, and l_2 passes through C. A *tessellation* of the strip formed by (l_1, l_2) is a tiling of the entire strip using congruent copies of triangle $\triangle ABC$ (allowing translation, rotation, and reflection), with no overlaps and no gaps.

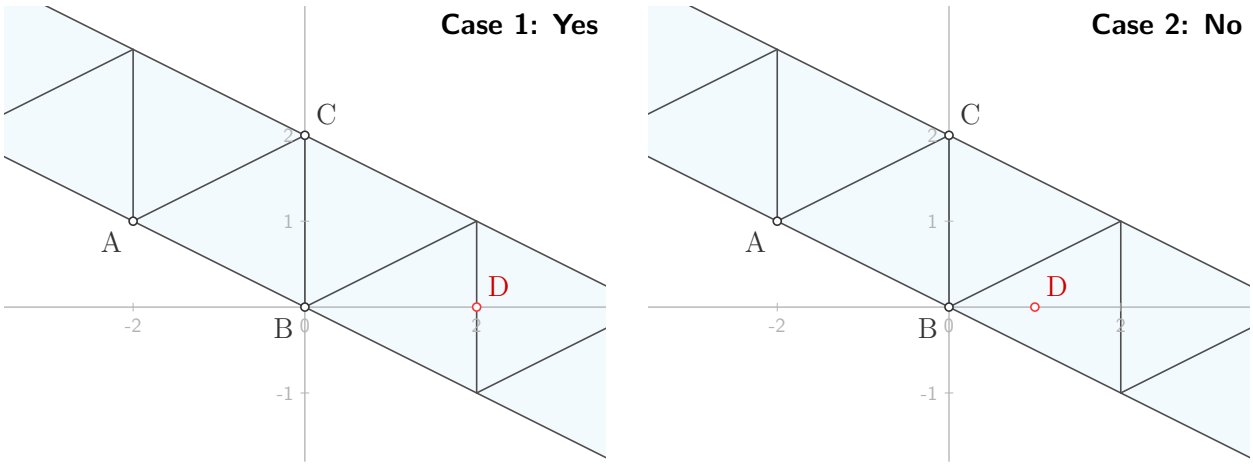


Illustration of the first two sample test cases.

Input

There are multiple test cases. The first line of the input contains an integer T ($1 \le T \le 10^3$) indicating the number of test cases. For each test case:

The first line contains two integers x_a and y_a ($-100 \le x_a, y_a \le 100$), the coordinates of point A.

The second line contains two integers x_b and y_b ($-100 \le x_b, y_b \le 100$), the coordinates of point B.

The third line contains two integers x_c and y_c ($-100 \le x_c, y_c \le 100$), the coordinates of point C.

The fourth line contains two fractions x_{d_1}/x_{d_2} and y_{d_1}/y_{d_2}, where $x_{d_1}, x_{d_2}, y_{d_1}, y_{d_2}$ are integers satisfying $-10^7 < x_{d_1}, y_{d_1} < 10^7$ and $0 < x_{d_2}, y_{d_2} < 10^7$, representing the coordinates of point D.

It is guaranteed that A, B, and C form a non-degenerate triangle.

Output

For each test case, output "Yes" in a line if there exists a valid tessellation such that point D lies on an edge or a vertex of at least one triangle in the pattern. Otherwise, output "No".

Sample Input 1

```
4
-2 1
0 0
0 2
2/1 0/1
-2 1
0 0
0 2
1/1 0/1
0 1
-2 1
0 2
1/2 7/4
0 1
-2 1
0 2
-3/2 3/4
```

Sample Output 1

```
Yes
No
Yes
No
```

Problem B
Batching Badges
Time limit: 2 seconds

It is the year 2075. While organizing your archives, you rediscovered a collection of badges from the 2025 ICPC Regional contests, originally distributed by the Universal Cup organization.

A complete UCup badge set.
In this problem, however, we only consider a subset of three colors.

Your badges come in three distinct colors only: Ultramarine (`u`), Crimson (`c`), and Periwinkle (`p`). A complete souvenir pack is defined as four badges appearing in the exact order: Ultramarine, then Crimson, then Ultramarine, then Periwinkle (`ucup`).

Your collection consists of $4n$ badges arranged in a row. You vividly recall that in 2025, this arrangement was a **perfect batch**:

- The sequence could be partitioned into n disjoint subsequences (i.e., using each badge in exactly one subsequence), such that each subsequence formed a complete souvenir pack (`ucup`).

However, over the last 50 years, some badges have faded, making their colors indistinguishable. These positions are represented by `?`.

Given the current state of the badge sequence S, calculate the number of ways to restore the faded colors (`?`) to one of `u`, `c`, or `p`, such that the resulting sequence is a perfect batch. As the answer may be large, output it modulo $998\,244\,353$.

Note that a subsequence can be derived from a string by removing zero or more characters without changing the order of the remaining characters. For example, `ucp`, `c`, and `ucup` are subsequences of the string `ucup`, while `uucp` is not a subsequence of the string `ucup`.

Input

The first line contains an integer n ($1 \le n \le 50$).

The second line contains a string S of length $4n$ consisting of characters `u`, `c`, `p`, and `?`.

Output

Output the number of ways to replace every `?` in S with one of `u`, `c`, or `p`, so that the resulting string represents a perfect batch, taken modulo $998\,244\,353$.

Sample Input 1

```
1
????
```

Sample Output 1

```
1
```

Sample Input 2

```
1
ucup
```

Sample Output 2

```
1
```

Sample Input 3

```
1
uucp
```

Sample Output 3

```
0
```

Sample Input 4

```
1
uu?u
```

Sample Output 4

```
0
```

Sample Input 5

```
2
????????
```

Sample Output 5

```
11
```

Sample Input 6

```
4
u???c???u???p???
```

Sample Output 6

```
437
```

Problem C
Potential Peak
Time limit: 6 seconds

A running club is recruiting members and growing rapidly, with exactly one new member joining each day. Every member comes with a specific initial skill level, and they are assigned a unique ID based on the order of their arrival.

At the end of every day, the Coach likes to calculate a metric he calls the "total potential" of the club. This value is simply the sum of the individual potentials of all current members. To keep the members motivated, the potential of a member is calculated through a hypothetical "what-if" scenario involving a single, intense workout.

In this imaginary workout, a member can choose to put in a certain amount of effort to temporarily boost their skill level. For every unit of effort they expend, their skill level increases by exactly one unit. In return, they gain one unit of "happiness" for every other member currently in the club who:

- joined the club strictly after they did; and
- was originally strictly more skilled than they were but has now been caught or surpassed thanks to the workout.

Note that during this calculation, only that member's skill level is increased, while the skill levels of all other members remain fixed at their original values.

The potential of a single member is defined as the maximum possible value of their net gain (happiness minus effort) they could achieve by choosing their effort level optimally. These workouts are entirely theoretical; they are used only for the Coach's daily report, and the members' actual skill levels remain unchanged from day to day.

Your task is to help the Coach calculate the total potential of the club at the end of each day of the recruiting period.

Input

The first line of input contains an integer n ($1 \leq n \leq 10^5$), representing the number of days the club is recruiting members.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq n$), where a_i is the initial skill level of the member who joins on day i.

Output

Output n integers in a single line. The k-th integer should be the total potential of the club at the end of day k.

Sample Input 1	Sample Output 1
8 1 2 3 1 2 3 2 3	0 0 0 0 1 3 5 9

Explanation of Sample 1: On day 5, there are 5 members with skill levels $[1, 2, 3, 1, 2]$. For member 1: There are 3 members joining later with strictly higher skills: member 2, member 3, and member 5. The maximum potential for member 1 is 1, achieved by either of the following strategies:

- If member 1 pays 1 unit of effort, their skill becomes 2. They will catch or surpass member 2 and member 5. The happiness is 2 units, and the gain is $2 - 1 = 1$ unit.
- If member 1 pays 2 units of effort, their skill becomes 3. They will catch or surpass members 2, 3, and 5. The happiness is 3 units, and the gain is $3 - 2 = 1$ unit.

The other members have maximum potential 0. Hence, the total potential for day 5 is 1.

On day 8, there are 8 members with skill levels $[1, 2, 3, 1, 2, 3, 2, 3]$. The optimal efforts are $[2, 1, 0, 2, 1, 0, 1, 0]$ and the corresponding maximum potentials are $[4, 2, 0, 2, 1, 0, 0, 0]$.

Sample Input 2	Sample Output 2
1 1	0

Problem D
Strategic Stones
Time limit: 2 seconds

Architecture is often the result of friction between two brilliant but opposing minds. Kotori and Umi, lead architects for a new metropolitan promenade, find themselves in a deadlock regarding the pavement's aesthetic direction. The promenade is modeled as a sequence of n cells in a single row.

Some cells have already been paved: some contain Kotori's signature white marble, while others contain Umi's preferred black granite. Many cells remain empty (marked as "`?`"). Kotori and Umi must now fill the remaining cells one by one.

To ensure total fairness in their collaboration and to prevent any single philosophy from dominating without a struggle, they have agreed to a formal design protocol:

- Kotori and Umi take turns placing a single stone into an empty cell, with Kotori going first.
- Kotori always places a white marble stone, as she believes the beauty of the promenade is measured by its **longest contiguous segment** of white marbles. She naturally seeks to maximize this length.
- Umi, seeking to provide structural contrast and rhythm, believes that long, uninterrupted white segments are monotonous. She always places a black granite stone, aiming to minimize that same maximum length.

Once a stone is placed, it cannot be moved or replaced. The process continues until all empty cells are filled. Given the initial state of the promenade, your task is to determine the final "Aesthetic Score" — the length of the longest contiguous segment of white marbles — assuming both architects play optimally to achieve their respective aesthetic goals.

Input

There are multiple test cases. The first line of the input contains an integer T ($1 \le T \le 10^4$) indicating the number of test cases. For each test case:

The first and only line contains a single string $s_1 s_2 \cdots s_n$ of length n ($1 \le n \le 10^5$), where: `0` represents a cell already containing Kotori's white marble; `1` represents a cell already containing Umi's black granite; `?` represents a currently empty cell.

It is guaranteed that the sum of n over all test cases does not exceed 10^6.

Output

For each test case, output one line that contains an integer indicating the final score of the game.

Sample Input 1	Sample Output 1
6 ?? 000 111 0??0??0 0?0?0?0 00?00?100?0	1 3 0 3 5 5

This page is intentionally left blank.

Problem E
Efficient Express
Time limit: 3 seconds

His dream, together with the Express Train in City C, roared through the station he was waiting at, leaving him the only one behind.

City C opened a new subway line with n stations. The i-th station has level $1 \le a_i \le k$, which represents the importance of the station. In general, the higher the level of a station, the more trains stop there.

City C plans to open m train services on the new subway line. The i-th service has a switching point p_i, and two level parameters x_i and y_i, indicating that the trains of that service will stop at all the stations with level at least x_i from the 1-st station to the p_i-th station, and all the stations with level at least y_i from the $(p_i + 1)$-th station to the n-th station. More formally, they stop at all stations j satisfying at least one of the following:

- $j \le p_i$ and $a_j \ge x_i$;
- $j > p_i$ and $a_j \ge y_i$.

Let's consider some examples.

- If $x_i = y_i = 1$, the trains will stop at every station. This kind of train service is also known as *Local Service*.
- If $x_i = y_i = k$, the trains will only stop at stations with level k. This kind of train service is also known as *Limited Express* or *Benchmark Train*.
- If $x_i = 1, y_i = k$, the service is a *Local Service* between 1 and p_i, and a *Limited Express* between $(p_i + 1)$ and n. This could be the *Through Service* that stops at every station in the downtown and speeds up in the suburb.

Two distinct stations are directly reachable if at least one train service stops at both stations. We call the subway line *efficient* if all pairs (i, j) of distinct stations with the same level $(a_i = a_j)$ are directly reachable.

You are given $p_1, \ldots, p_m$ and $x_1, \ldots, x_m$. Please calculate the number of possible layouts of $y_1, \ldots, y_m$, consisting of positive integers between 1 and k, such that the subway line is *efficient*. As the number might be large, output the answer modulo 998 244 353.

Input

The first line contains three integers n $(2 \le n \le 10^3)$, m $(1 \le m \le 10^3)$, and k $(1 \le k \le 500)$, indicating the number of stations, the number of train services, and the upper limit of the levels of stations. It is NOT guaranteed that there is at least one station with each level.

The second line contains n integers $a_1, a_2, \ldots, a_n$ $(1 \le a_i \le k)$, indicating the level of each station.

The following m lines describe the train services. The i-th of these lines contains two integers p_i $(1 \le p_i < n)$ and x_i $(1 \le x_i \le k)$, indicating two of the parameters of the i-th train service.

Output

Output an integer representing the number of possible arrays $y_1, \ldots, y_m$, modulo 998 244 353.

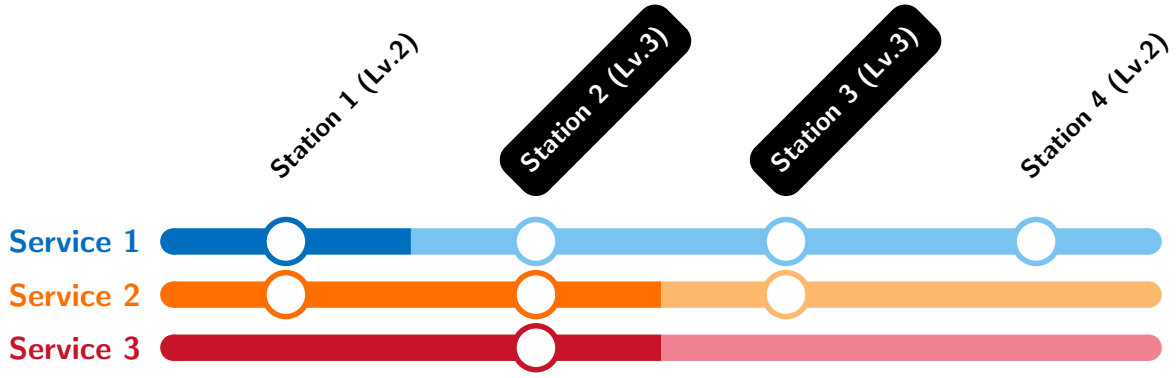
Sample Input 1

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

Sample Output 1

```
48
```

Explanation of Sample 1: Below is the illustration for a possible configuration, $y = [2, 3, 4]$:



- Service 1: $p_1 = 1, x_1 = 2, y_1 = 2$;
- Service 2: $p_2 = 2, x_2 = 2, y_2 = 3$;
- Service 3: $p_3 = 2, x_3 = 3, y_3 = 4$.

Sample Input 2

```
7 3 5
2 4 3 2 4 2 3
2 2
4 3
5 2
```

Sample Output 2

```
80
```

Sample Input 3

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

Sample Output 3

```
100
```

Problem F
Set of Intervals 2
Time limit: 6 seconds

Little Cyan Fish is conducting research on a set of intervals.

In this problem, we use $[l, r]$ to denote the interval $\{x \mid l \leq x \leq r\}$. For a set S containing intervals $[l, r]$, it is said to be *Cyanic* if and only if the following three conditions hold:

1. $S \subseteq \{[l, r] \mid l, r \in \mathbb{Z}, 0 \leq l \leq r \leq n\}$. That is, every element of S is a closed interval $[l, r]$ with integer endpoints contained in $[0, n]$.
2. For all integers $i \in [0, n]$, the interval $[i, i] \in S$.
3. For any two intervals $[l_0, r_0] \in S$ and $[l_1, r_1] \in S$, if $l_0 \leq l_1 < r_0 \leq r_1$, then the intervals $[l_0, l_1]$ and $[r_0, r_1]$ must also belong to S.

To help you understand, Little Cyan Fish considers the following scenarios for $n = 3$.

- The set $\{[0, 0], [2, 2]\}$ is **not** a Cyanic set. The interval $[1, 1] \notin S$, so the second condition is violated.
- The set $\{[0, 0], [1, 1], [2, 2], [3, 3], [0, 2], [1, 3]\}$ is **not** a Cyanic set. When we choose $[l_0, r_0] = [0, 2]$ and $[l_1, r_1] = [1, 3]$, the interval $[l_0, l_1] = [0, 1] \notin S$, so the third condition is violated.
- The set $\{[0, 0], [1, 1], [2, 2], [3, 3], [0, 2], [0, 1], [1, 2]\}$ is a Cyanic set.
- The set $\{[0, 0], [1, 1], [2, 2], [3, 3], [0, 3], [1, 2]\}$ is also a Cyanic set.

Of course, there are many, many Cyanic sets, and Little Cyan Fish loves all of them. Let $\mathcal{C}$ be the set of all Cyanic sets. For a given integer q, Little Cyan Fish wants to know the sum:

$$\sum_{S \in \mathcal{C}} q^{|S|}$$

As the sum can be huge, Little Cyan Fish asks you to calculate the sum modulo $998\,244\,353$.

Input

The first line of the input contains two integers n ($1 \leq n \leq 10^5$) and q ($1 \leq q < 998\,244\,353$).

Output

Output a single line containing a single integer indicating the answer.

Sample Input 1	Sample Output 1
1 2	12

Explanation of Sample 1: There are two Cyanic sets $\{[0, 0], [1, 1]\}$ and $\{[0, 0], [1, 1], [0, 1]\}$, so the answer is $2^2 + 2^3 = 12$.

Sample Input 2	Sample Output 2
3 1	22

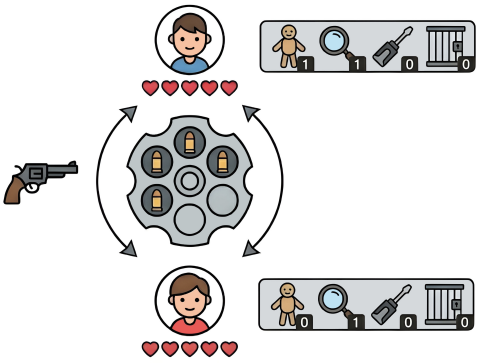
Sample Input 3	Sample Output 3
10 3	855061512

This page is intentionally left blank.

Problem G
Revolver Roulette
Time limit: 15 seconds

This is an interactive problem. Remember to flush the output buffer after every print. To flush your output, you can use:

- `fflush(stdout)` or `cout.flush()` in C/C++;
- `System.out.flush()` in Java and Kotlin;
- `sys.stdout.flush()` in Python.



The concept of the game, illustrated by Nano Banana Pro.

You are participating in a strategic turn-based game known as *Revolver Roulette*.

The game begins with a revolver loaded with exactly 6 bullets. Each bullet is either *live* or *blank*, independently with equal probability. Both players always know the current counts of live and blank bullets remaining in the cylinder, but the exact order is unknown.

Each player starts with 5 health points (HP) and 2 items. If a player's HP drops to 0 or below, they are eliminated, and the other player wins. Players take turns performing actions.

Turn Procedure

During their turn, the active player may perform any number of the following actions:

- Use any item.
- Perform any shooting action.

The turn ends only when a shooting action causes the turn to pass to the other player, or some player is eliminated.

Shooting Actions

The active player must choose one of the following shooting actions:

- `Fire at Opponent`: The active player fires the next bullet at the other player. If it is live, the other player loses 1 HP. Regardless of the bullet type, the turn passes to the other player.
- `Fire at Self`: The active player fires the next bullet at themselves. If it is live, the active player loses 1 HP, and the turn passes to the other player. Otherwise, if it is a blank, the active player retains the turn and continues performing actions.

Items

When an item is used, it is consumed, and the active player retains the turn, regardless of the item type. Items are visible to both players. When a player obtains a new item (including each of the initial items), it is chosen uniformly at random from the following four types:

- `Dummy`: Fire the next bullet at a training dummy. This consumes the bullet and is not considered a shooting action.
- `Magnifier`: Peek at the next bullet to reveal whether the next bullet is live or blank. Both players learn the result.

- `Converter`: Flip the type of the next bullet (live ↔ blank). Both players observe the updated counts of live and blank bullets.
- `Cage`: Gain a one-time effect that prevents the turn from ending on the next shooting action that would normally end it. Specifically:
  - If the player later performs `Fire at Opponent` (regardless of whether the bullet is live or blank), the effect applies and the turn does not end.
  - If the player later performs `Fire at Self` and the bullet is live, the effect applies and the turn does not end (the player still loses 1 HP, but retains the turn).

  The effect is consumed immediately after it applies once. At most one `Cage` item may be used per turn.

Reloading

When the cylinder is empty, 6 new bullets are immediately loaded (each independently live or blank with equal probability). Both players are informed of the new counts of live and blank bullets. Additionally, each player is resupplied up to 2 items (a player with k items receives $(2 - k)$ new items, each independently chosen uniformly at random from the above four types).

Goal

You are the player who takes the first turn, and your goal is to win the game by eliminating your opponent.

Input

The first line contains two integers n and w, indicating that you need to win at least w out of n games to get accepted. There are only three tests:

- Sample test: $n = 1$ and $w = 0$. (0%)
- Secret test 1: $n = 2\,000$ and $w = 1\,000$. (50%)
- Secret test 2: $n = 20\,000$ and $w = 11\,000$. (55%)

It can be shown that, when both players play optimally, the player who takes the first turn wins approximately 56.89% of the games.

Interaction Protocol

Game Start

Your solution acts as the player who takes the first turn against the interactor. At the start of each game, the interactor prints an integer g:

- If $g = 0$, all games in the current run are complete. Your solution must terminate immediately.
- Otherwise, g is the current game index ($1 \le g \le n$).

Due to technical issues, your solution is executed **twice** on each test:

- **The interactor can decide whether to restart your solution** by giving $g = 0$ at any time.

You will receive the same n and w for each run in a test. Therefore, n is **not** always equal to the number of games in the current run. It is guaranteed that each game $1, \ldots, n$ is played exactly once across the two runs. Apart from this, your solution can safely ignore the run twice condition.

Game State

Then, the interactor prints the game state line containing 13 non-negative integers:

$$t \quad l \quad b \quad h_1 \quad d_1 \quad m_1 \quad c_1 \quad k_1 \quad h_2 \quad d_2 \quad m_2 \quad c_2 \quad k_2$$

- $t \in \{0, 1, 2\}$: turn status.
- l, b ($1 \le l + b \le 6$): the current counts of live and blank bullets remaining in the cylinder.
- h_1 ($0 \le h_1 \le 5$): your current HP.
- d_1, m_1, c_1, k_1 ($0 \le d_1 + m_1 + c_1 + k_1 \le 2$): the number of `Dummy`, `Magnifier`, `Converter`, and `Cage` items you currently hold.
- h_2 ($0 \le h_2 \le 5$): your opponent's current HP.
- d_2, m_2, c_2, k_2 ($0 \le d_2 + m_2 + c_2 + k_2 \le 2$): the number of `Dummy`, `Magnifier`, `Converter`, and `Cage` items your opponent holds.

Case $t = 0$ (Game Over)

The current game has ended. The interactor proceeds to the next game (printing the new g).

Case $t = 1$ (Your Turn)

You must print an integer to choose an action:

1. `Fire at Opponent`.
2. `Fire at Self`.
3. Use a `Dummy` item.
4. Use a `Magnifier` item.
5. Use a `Converter` item.
6. Use a `Cage` item.

If the action is invalid (e.g., using an item you don't have, or using a `Cage` item more than once in the same turn), the game state remains unchanged, the interactor prints 0 and the same state line (13 integers) again. Although it remains your turn, you must make another valid action to proceed with the game.

Otherwise, the action is effective and the interactor prints a result integer:

- For actions 1–5: Returns 1 if the next bullet before the action is live, 2 if it is blank.
- For action 6: Returns 1 (no bullet info revealed).

The interactor then prints the updated game state line.

Case $t = 2$ (Opponent's Turn)

The interactor simulates your opponent's action and prints two lines:

- The action ID $a \in \{1, \ldots, 6\}$.
- The result integer (1 or 2 as described above).

Your opponent never makes an illegal action. Immediately afterward, the interactor prints the next game state line (13 integers).

For each round of the game, the game simulator uses a pseudorandom number generator with a fixed random seed. Also, the opponent in the interactor employs a deterministic strategy. The decision of restarting your solution is independent of the interaction procedure. Therefore, the same sequence of actions always leads to the same game state.

A testing tool is provided to help you develop and test your solution.

Read **Sample 1, Pass 1** **Write**

```
1 0
1
1 3 3 5 0 0 0 2 5 1 0 1 0
```

```
6
```

```
1
1 3 3 5 0 0 0 1 5 1 0 1 0
```

```
1
```

```
1
1 2 3 5 0 0 0 1 4 1 0 1 0
```

```
6
```

```
0
1 2 3 5 0 0 0 1 4 1 0 1 0
```

```
2
```

```
2
1 2 2 5 0 0 0 1 4 1 0 1 0
```

```
1
```

```
1
2 1 2 5 0 0 0 1 3 1 0 1 0
5
2
2 2 1 5 0 0 0 1 3 1 0 0 0
1
1
1 1 1 4 0 0 0 1 3 1 0 0 0
```

```
1
```

```
1
2 0 1 4 0 0 0 1 2 1 0 0 0
3
2
2 2 4 4 0 1 0 1 2 0 1 0 1
2
1
1 1 4 4 0 1 0 1 1 0 1 0 1
```

```
4
```

```
1
1 1 4 4 0 0 0 1 1 0 1 0 1
```

```
1
```

```
1
0 0 4 4 0 0 0 1 0 0 1 0 1
0
```

Read **Sample 1, Pass 2** **Write**

```
1 0
0
```

Problem H
Hextech High-roll
Time limit: 2 seconds

Jayce is playing a game of *ARAM: Mayhem*. He has reached the Augment Selection phase and is looking for the perfect augmentation to carry his team. Specifically, he is desperately hunting for the Golden augment known as *Slow and Steady*.

The Game UI for card selection. Oops, it looks like Jayce cannot find *Slow and Steady* in the Prismatic pool...

Currently, the game interface presents k augment cards to Jayce. While each card has a different effect, to Jayce, the i-th visible card is worth a_i value points. There is a hidden pool of n other augment cards waiting in the deck. The value points of these cards are known to be the multiset $\{b_1, b_2, \ldots, b_n\}$, but their order is shuffled uniformly at random.

Jayce can refresh each of the k card slots **at most once**. The process works as follows:

- There are individual reroll buttons under each of the k cards (as shown in the game UI).
- Jayce can choose a slot that hasn't been rerolled yet, discard the current card, and draw the top card from the deck to replace it.
- Jayce sees the result of a reroll immediately before deciding his next move: whether to reroll another available slot, or stop.

At the end of the process, Jayce will select one card among the k cards currently visible on the screen.

Jayce plays optimally to maximize the expected value of the card he eventually picks. Calculate this maximum expected power level.

Input

The first line contains two integers k and n ($1 \le k \le n \le 10^5$), indicating the number of visible slots and the number of cards in the deck.

The second line contains k integers $a_1, a_2, \ldots, a_k$ ($-10^5 \le a_i \le 10^5$), indicating the power levels of the initially visible cards.

The third line contains n integers $b_1, b_2, \ldots, b_n$ ($-10^5 \le b_i \le 10^5$), indicating the power levels of the cards in the deck.

Output

Output the maximum expected value of the card as a single decimal real number.

Your answer will be considered correct if its absolute or relative error does not exceed 10^{-4}. Formally speaking, suppose that your output is a and the jury's answer is b, your output is accepted if and only if $\frac{|a-b|}{\max(1,|b|)} \le 10^{-4}$.

Sample Input 1

```
1 2
10
1 100
```

Sample Output 1

```
50.5
```

Sample Input 2

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

Sample Output 2

```
7.4
```

Sample Input 3

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

Sample Output 3

```
-1
```

Sample Input 4

```
5 14
1 3 5 7 9
-1 -1 0 0 2 2 4 4 6 6 8 8 10 10
```

Sample Output 4

```
9.505494505
```

Problem I
Redundancy Refrain
Time limit: 6 seconds

Professor Chen, a distinguished expert in Computational Paleography, has spent years studying the scripts of the *Non-Redundant* dynasty. The scribes of this era followed a peculiar aesthetic doctrine: they believed that a text loses its spiritual potency if it contains "echoes".

In this context, an *anagrammatic echo* is defined as any two distinct segments of a text that, while not necessarily identical in their ordering, contain the exact same symbols in the exact same quantities. For a text represented as a sequence of symbols, an echo exists if there are two different starting positions x and y such that the subarray of length l beginning at x is a permutation of the subarray of length l beginning at y. For example, consider the sequence $[1, 2, 1, 1, 3, 1, 2, 1]$. The segment of length 4 starting at the second position, $[2, 1, 1, 3]$, and the segment of length 4 starting at the fifth position, $[3, 1, 2, 1]$, form an anagrammatic echo.

Professor Chen has unearthed several such scrolls, but many symbols have faded over time, leaving gaps (represented by 0). Your task is to fill these gaps using symbols from the allowed set $\{1, 2, \ldots, k\}$ such that the resulting *Dissonance Score* of the scroll is minimized. The *Dissonance Score* is the maximum length l of any anagrammatic echo present in the completed sequence.

Input

There are multiple test cases. The first line of the input contains an integer T ($1 \le T \le 10^4$) indicating the number of test cases. For each test case:

The first line contains two integers n and k ($2 \le n \le 2 \times 10^5$, $1 \le k \le n$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le k$).

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

Output

For each test case, output n integers $b_1, b_2, \ldots, b_n$ ($1 \le b_i \le k$) representing the restored sequence. The sequence b must match a at all positions where $a_i \neq 0$, and it must achieve the minimum possible Dissonance Score.

If there are multiple ways to restore the scroll optimally, Professor Chen will accept any of them.

Sample Input 1	Sample Output 1
4 8 3 0 1 0 2 1 3 0 2 3 3 1 0 2 9 7 0 1 1 0 5 0 4 0 7 6 6 0 0 0 0 0 0	1 1 3 2 1 3 2 2 1 3 2 2 1 1 3 5 6 4 4 7 1 2 3 4 5 6

This page is intentionally left blank.

Problem J
Tree-mendous Transmission
Time limit: 3 seconds

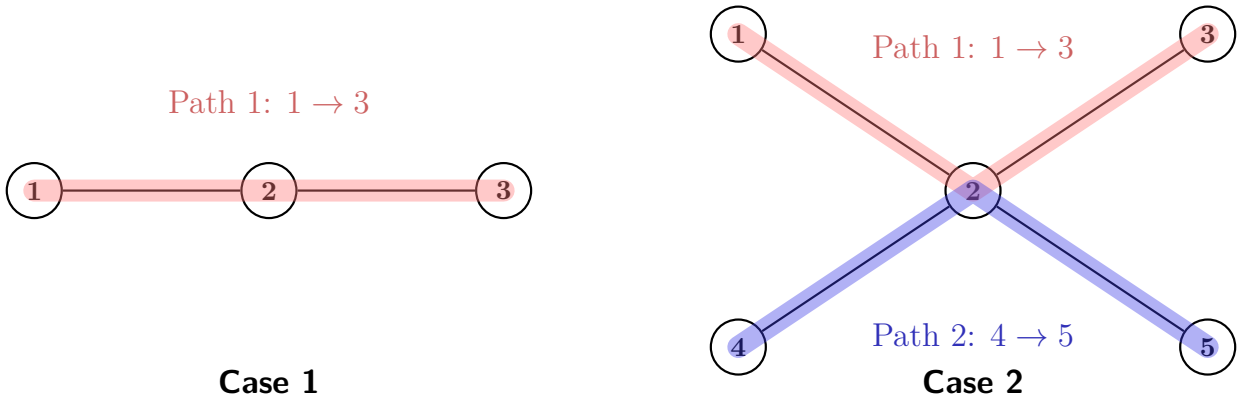
Piggy (who, in his spare time, manages the Interstellar Connectivity Protocol Commission) is facing a delicate hardware conundrum. The latest communication grid, consisting of n relay stations connected by $(n-1)$ fiber-optic links to form a tree structure, is finally ready for deployment. However, the system must be powered on with extreme caution.

The activation process involves firing laser pulses along specific paths. A path is defined as the unique simple route between two stations. Due to the catastrophic risk of "Quantum Backflow Interference" — a phenomenon that keeps the commission's lawyers awake at night — the following protocol must be strictly observed:

- Transmission paths must be established one by one.
- To prevent power surges, each newly established path may share **at most one station** with the set of stations already activated by any previous path.

Formally, let S be the set of currently activated stations. For any new path P consisting of a set of stations $V(P)$, the condition $|V(P) \cap S| \leq 1$ must hold. Once a path is successfully fired, all stations along that path are considered part of the activated set S.

Your mission, should you choose to accept it, is to determine the minimum number of laser pulses required to activate every station in the entire network and provide one such sequence of paths to satisfy the legal department.



The illustration of the first two sample test cases. Note that in Case 2, the second pulse shares only station 2 with the first, satisfying the non-interference rule.

Input

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

The first line contains an integer n ($2 \leq n \leq 3 \times 10^5$), the number of stations in the grid.

For the following $(n-1)$ lines, the i-th line contains two integers u_i and v_i ($1 \leq u_i, v_i \leq n, u_i \neq v_i$), describing a bidirectional fiber-optic link between station u_i and station v_i.

It is guaranteed that the links form a tree and that the sum of n over all test cases does not exceed 3×10^5.

Output

For each test case, output the minimum number of paths k on the first line. Following this, output k lines, each containing two integers a_i and b_i, representing the endpoints of the i-th path in the sequence. Note that a_i may equal b_i if the path consists of a single station.

If multiple optimal sequences exist, any one that strictly follows the non-interference protocol will be accepted.

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

Problem K
Magically Marked Matching Master
Time limit: 2 seconds

This is a multi-pass, interactive problem. Remember to flush the output buffer after every print. To flush your output, you can use:

- `fflush(stdout)` or `cout.flush()` in C/C++;
- `System.out.flush()` in Java and Kotlin;
- `sys.stdout.flush()` in Python.

The concept of the game, illustrated by ChatGPT.

In the classical *online matching* game, edges of a graph are revealed one by one. Each time an edge is shown, you must immediately and irrevocably decide whether to include it in your matching. Your goal is to ensure that no two chosen edges share a vertex while maximizing the total number of edges selected.

You are an ambitious computer scientist, hungry for fame. As you well know, finding a maximum matching in an online setting is provably impossible with high probability—even for the simplest classes of graphs like bipartite graphs. Yet where others see a fundamental barrier, you see an opportunity for glory. Today, you claim to have cracked the long-standing open problem for trees and are ready to unveil your groundbreaking algorithm to the world.

There is just one problem: yesterday, you discovered a bug in your proof, and you still have no idea how to fix it. Unwilling to abandon the spotlight, you decide to turn your demonstration into a magic show—your assistant, hidden backstage, will transmit *a little* hint about the entire graph before the game begins.

Your original plan was simple: have your assistant send the answer to each edge query directly. To this end, you have installed a secret earpiece capable of receiving a single binary string of length $(n-1)$. But the audience is skeptical of your "breakthrough." To prevent any pre-arranged trickery, they demand that the edges be presented in a uniformly random order—one that neither you nor your assistant can predict or control.

The stage is set. The lights are on you. Now prove your claim—without changing the channel.

Interaction Protocol

Your solution is executed twice on each test. In the `prepare` round, your solution will act as the assistant. In the `play` round, your solution will act as *you*, the magician. In either of the rounds, your solution will be evaluated as an interactive procedure.

Prepare Round

The first line of the input contains the word `prepare`. The second line contains an integer n ($2 \le n \le 500$) — the number of vertices. The following $(n-1)$ lines each contain two integers u and v ($1 \le u, v \le n$, $u < v$), denoting an undirected edge between u and v in the tree.

You must output one line containing a binary string s of exactly length $(n-1)$ consisting of characters `0` and `1` only. This is the string transmitted to the second `play` round. If your output does not align with the desired format, you will receive the `Wrong Answer` verdict.

Play Round

Your solution will be restarted for the `play` round.

The first line contains the word `play`. The second line contains the integer n. The third line contains the binary string s exactly as you printed it in the `prepare` round.

After that, the game starts with $(n-1)$ turns. In each turn, a line containing two integers u and v ($1 \le u, v \le n$, $u < v$) will be provided. As a response, you should output one line containing either `take` or `ignore`. All edges in the first round will be provided exactly once, and the order of the edges is chosen uniformly at random from $(n-1)!$ possible permutations.

When there is an output in invalid format, the game will end immediately with no further information from the interactor.

Evaluation of Correctness

Let M be the set of edges your solution takes. It is considered acceptable if no vertex is incident to more than one taken edge and $|M| = |M^*|$, where M^* is the maximum matching of the graph.

There are exactly 50 tests, excluding the sample tests. In order to pass the problem, your solution should be correct on all of the tests, including the sample tests.

All shuffles are guaranteed to be fixed before receiving the solution's output. The randomness is fixed for each test, implemented by fixing the seed of a pseudorandom number generator.

A testing tool is provided to help you develop and test your solution.

Read **Sample 1, Pass 1** **Write**

```
prepare
3
1 2
1 3
```

```
00
```

Read **Sample 1, Pass 2** **Write**

```
play
3
00
1 3
```

```
take
```

```
1 2
```

```
ignore
```

Explanation of Sample 1: Since $|M^*| = 1$, taking any edge is acceptable, and the hint can be arbitrary.

Problem L
Logical Resonance
Time limit: 2 seconds

This is a multi-pass problem.

In the digital archives of Kivotos, Plana has discovered a collection of mysterious records known as Bitemporal Logs. Each log consists of n entries labeled 1 to n forming a rooted tree. However, the structural constraints differ depending on whether the temporal flow is retrospective (**Logic A**) or prospective (**Logic B**):

- **Logic A:** The tree is rooted at entry 1; every other entry i has a parent p_i such that $p_i < i$.
- **Logic B:** The tree is rooted at entry n; every other entry i has a parent q_i such that $q_i > i$.

To analyze the structural properties, Plana defines two types of entries:

- **Terminal:** An entry that serves as a parent to *no* other entries.
- **Hub:** An entry that serves as a parent to *at least one* other entry.

Plana observed a perfect symmetry called *Logical Resonance*. This property holds between a Logic A log and a Logic B log if and only if:

$$\textit{For every } i\textit{, entry } i \textit{ is a Terminal in Logic A} \iff \textit{entry } i \textit{ is a Hub in Logic B.}$$

Plana has mathematically proven that the number of valid Logic A logs and Logic B logs under this constraint is identical. Now, she tasks you with designing a Universal Translation Protocol — a bijection — to transform one log format into the other.

Evaluation of correctness

Your solution is executed twice on each test. In the first run, your solution needs to convert each Logic A log into a Logic B log that satisfies the Logical Resonance condition. In the second run, given a Logic B log produced by your first run, your solution needs to exactly recover the original Logic A log.

The input of the second run consists of the same Logic B logs as your output from the first run, possibly in a different order. For each input Logic B log in the second run, you need to output its corresponding Logic A log. Your solution is considered correct if, for every such Logic B log, your output is exactly the same Logic A log that generated it in the first run.

A testing tool is provided to help you develop and test your solution.

Input

The first line of the input contains two integers r ($r \in \{1, 2\}$) and T ($1 \le T \le 10^5$), representing the run number and the number of test cases.

For each test case, the first line contains an integer n ($2 \le n \le 10^3$).

If $r = 1$, the second line contains n integers $p_1, p_2, \ldots, p_n$ representing the Logic A log. It is guaranteed that $p_1 = 0$, and for $2 \le i \le n$, $1 \le p_i < i$ is the entry that entry i is attached to. Here we use 0 to denote that an entry has no parent (i.e., it is the root).

Otherwise, if $r = 2$, the second line contains n integers $q_1, q_2, \ldots, q_n$ representing the Logic B log. It is guaranteed that $q_n = 0$, and for $1 \le i \le n - 1$, $i < q_i \le n$ is the entry that entry i is attached to.

It is guaranteed that the sum of n^2 over all test cases does not exceed 10^7.

Output

If $r = 1$, for each test case, output n integers separated by a space, $q_1, q_2, \ldots, q_n$, representing the converted Logic B log. It must hold that $q_n = 0$, and for $1 \le i \le n-1$, $i < q_i \le n$. The *Logical Resonance* property must hold: entry i is a Terminal in Logic A if and only if entry i is a Hub in Logic B.

Otherwise, if $r = 2$, for each test case, output n integers separated by a space, $p_1, p_2, \ldots, p_n$, representing the restored Logic A log.

Sample Input 1 **Pass 1** **Sample Output 1**

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

Sample Input 1 **Pass 2** **Sample Output 1**

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

Explanation of Sample 1: A possible valid bijection is shown below.