Legoyramids

After so many years of competitive programming, Asilbek finally decided to never sit in front of a computer again. In search of a new hobby to keep himself entertained, he discovered a new passion: **building pyramids out of LEGO bricks**.

Each LEGO piece is **1×1** in size and has a **height of 1**. Asilbek declared:

> *"No more coding today! I'm building pyramids. This is the new life — just me and my LEGOs!"*

Of course, he's not doing this just for fun — Asilbek is determined to become the **ultimate LEGO pyramid builder** in the world. But there's a small problem...

The Pyramid Rules

Asilbek builds the pyramid layer by layer from the bottom up:

- The bottom layer has $n \times m$ LEGO blocks.
- Each next layer is **a units shorter in width** and **b units shorter in height** than the layer below it.
- But **no dimension can fall below 1** — if it does, it is considered to be 1.

So, the size of the i-th layer is:

```
layer(i) = max(n - a × (i - 1), 1) × max(m - b × (i - 1), 1)
```

Asilbek has only **C LEGO blocks**, and he wants to build **as many layers as possible** without exceeding this number. Determine the **maximum number of pyramid layers** Asilbek can build using no more than C LEGO blocks in total.

Input

The first line contains a single integer T — the number of test cases.

For each test case:
A single line contains five integers:

```
n m a b C
```

- n, m — dimensions of the base layer
- a, b — reduction per layer (in width and height)
- C — number of LEGO blocks Asilbek has

Output

For each test case, print a single integer — the **maximum number of layers** Asilbek can build without using more than C blocks.

Constraints

- $1 \le T \le 10^5$
- $0 \le n, m \le 10^9$
- $0 \le a, b \le 1$
- $0 \le C \le 10^{18}$

Subtasks

1. (6 points) $a = b = 0$
2. (8 points) $\sum C \le 10^6$
3. (12 points) $\sum (n + m) \le 4 \times 10^5$, $a = 0$
4. (14 points) $\sum (n + m) \le 4 \times 10^5$
5. (60 points) No additional constraints

Example

Input

```
1
4 3 1 1 25
```

Output

```
8
```

Balloonburst

You are given n balloons inside a hall. The ceiling of the hall is at a height of h meters. Each balloon initially floats at height a_i meters and rises upward at a constant rate of v_i meters per second. A balloon **bursts** the moment it reaches or exceeds the ceiling height h. You are also given a duration t in **minutes**. Your task is to determine:

1. How many balloons **do not burst** after t minutes?
2. Among the balloons that do not burst, print the **index** of the one that is at the **maximum height**.
    - If multiple balloons have the same maximum height, choose the one with the **largest index**.
    - If no balloons remain unburst, print `0 -1`.

Input

The first line contains three integers n, h, and t — the number of balloons, the height of the ceiling, and the time in minutes. Each of the next n lines contains two integers a_i and v_i — the initial height and the upward velocity of the i-th balloon.

Output

Print a single line containing two space-separated integers:

- The number of balloons that do not burst after t minutes.
- The index of the highest unburst balloon (or `-1` if there are none).

Constraints

- $1 \leq n, t \leq 10^5$
- $0 \leq a_i < h \leq 10^9$
- $1 \leq v_i \leq 10^4$

Subtasks

1. (15 points) $n = 1$ and $a_1 = 0$
2. (20 points) $n = 1$
3. (25 points) $a_i = 0$ for all i

4. (40 points) No additional constraints

Examples

Example 1

Input

```
4 100 2
10 1
20 5
30 0
50 3
```

Output

```
2 4
```

Example 2

Input

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

Output

```
0 -1
```

Example 3

Input

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

Output

```
2 2
```

Cosmotrips

In the year **3025**, *Intergalactic Miller Agency* maintains two gigantic networks of corridors:

- **Network A** – the ancient "**Gates of Silk**" discovered by humanity's first pioneers.
- **Network B** – the brand-new "**Nebula Express**" built by the Cosmic Union.

Each network consists of exactly $N-1$ corridors and links all N planets, which are numbered from 1 to N. From every planet there is **exactly one** route to every other planet inside one network. Meaning that networks have structure of tree.

Both structures connect exactly the same N planets, but their corridors are *weighted*: the weight of a corridor is its **safety rating** – the lower the safer.

For two planets u, v let

- $g(u,v)$ be the risk of fly from u to v. The risk of fly from u to v is defined as the **maximum weight** of all corridors on the unique route from u to v inside **Network A**
- $f(u,v)$ be the same quantity inside **Network B**

Thus g (respectively f) is the worst corridor you must traverse inside Network A (respectively Network B). Remember that $g(x,x) = f(x,x) = 0$ for any x.

During a cargo mission commander **Nazarbek** needs to reach planet y from planet x. To finish his mission, he does the following:

1. He picks **any** planet z;
2. Flies from x to z inside **Network A** (having $g(x,z)$ risk);
3. Makes "*Hyper-jump*" to the **Network B** (instant, risk-free);
4. Continues trip from z to y inside **Network B** (having $f(z,y)$ risk).

Total risk of trip using stopover z is $\mathrm{risk}(x,y,z) = \max\big(g(x,z),\ f(z,y)\big)$

Nazarbek, being cautious, chooses the best stopover z:

$\mathrm{minrisk}(x,y) = \min_{1 \le z \le N}\ \mathrm{risk}(x,y,z)$

Given Q such missions, output every $\mathrm{minrisk}(x_i, y_i)$

Input

The first line contains an integer T — the number of test cases.

Each test case starts with a line containing two integers N and Q — the number of planets and the number of queries.

The next $N-1$ lines describe **Network A**. The i-th of these lines contains three integers u_i, v_i, w_i — the indices of two planets connected by a corridor and its safety rating.

The following $N-1$ lines describe **Network B** in the same format: a_i, b_i, c_i.

Finally, Q lines follow. The j-th line contains two integers x_j, y_j — the endpoint planets of the j-th mission query.

Constraints

- $2 \le N, Q \le 2 \cdot 10^5$
- $0 \le w_i, c_i \le N$
- $1 \le x_i, y_i \le N$
- It is guaranteed that the two sets of $N-1$ edges each form trees on N vertices

Subtasks

1. **(6 points)** $N, Q \le 500$ and each network forms a **path** (every planet has degree ≤ 2)
2. **(9 points)** $N, Q \le 5,000$ and each network forms a **path**
3. **(21 points)** Each network forms a **path**
4. **(8 points)** $N, Q \le 500$
5. **(11 points)** $N, Q \le 5,000$
6. **(45 points)** No additional constraints

Output

For each query print a single integer from a new line – $\mathrm{minrisk}(x_i, y_i)$.

Examples

Example 1

Input

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

Output

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

Cursedtablet

A long time ago in Uzbekistan, a boy named Alisher found a strange stone tablet under an apricot tree. It had **two rows of numbers** on it:

- The **first row** shows how things are right now - maybe bad, maybe okay.
- The **second row** shows how things *should* be - happy, peaceful, and perfect.

Your job is to help Alisher **turn the first row into the second row**, using special magical moves.

Magic Moves You Can Use

Alisher can use these two types of moves:

- $1\ l\ r\ x$ – Choose **exactly** K **numbers in a row**, starting from position l until position r, and **add the number** x (positive or negative) to each of them. Note that r should always be equal to $l + K - 1$ (e.g: If $l = 2$ and $K = 3$, then a magic move can be $124x$: you add x to positions 2, 3, and 4.)
- 2 – Flip the whole row backward! (The first becomes last, the second becomes second-to-last, etc.)

Important Rules

- Alisher must **not** let any number become bigger than 10^{16} or smaller than -10^{16} after any move.
- Some puzzles are **impossible to solve** — if you can't fix the first row, print -1.
- If it's possible, find the **smallest number of moves** and show which moves to do.

Input

The first line has a number T — the number of puzzles.

For each puzzle:

- First line: two numbers N and K — the number of symbols and the size of the group Alisher can change in one move.
- Second line: N numbers — the current state (first row).
- Third line: N numbers — the goal state (second row).

Output

For each puzzle:

- If it's **not possible** to fix the first row, print -1.
- Otherwise:
    - First, print a number — how many magic moves Alisher needs.
    - Then print each move on its own line:
        - Either $1\ l\ r\ x$
        - Or 2

Subtasks

You will get **100% of the score for subtask** if:

- You print the **minimum number of moves** - mn, and
- Your operations **correctly** turn the first row into the second row in every test case.

You will get **50% of the score for a subtask** if:

- You print mn and the operations are in the **correct format**, but
- The operations **don’t actually** make the first row equal to the second.

So even if you can't solve a test case, printing any valid mn and well-formed operations can still earn you partial credit.

1. (20 points) $a_1 = a_2 = \ldots = a_N$, $b_1 = b_2 = \ldots = b_N$
2. (18 points) $K = 1$
3. (12 points) $K = N$
4. (50 points) No additional constraints

Examples

Example 1

Input

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

Output

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

Handshakes

This is interactive problem!

You have entered the terrifying field of the Squid Game, where there are N participants aiming to win. Each participant is numbered from 1 to N. Among them is your closest friend, but his number is hidden. To save your friend, you must identify his number in a game called "handshakes", watched over by masked Guard.

Game Rules

The participants stand on an infinite line, where each position has a defined location. Participant number i starts at position i. The Guard announces a number D, which will be the step size during the game. Each participant is given a card marked either "L" or "R". Then, the following process is repeated 10^{100} times:

1. **Movement:** All participants move simultaneously based on their card:
    - A participant with an "L" card at position p moves D steps to the left to position $p - D$.
    - A participant with an "R" card moves D steps to the right to position $p + D$.
    - The line is infinite, so participants can move to any (positive or negative) position without restriction.
2. **Handshake:** After the movement, participants who land on the same position will shake hands and **swap** their cards with each other.

Your Task

Your friend is one of the participants, and your job is to identify his number and ensure their survival. The impatient and cold Guard only allows you to make query - assign the cards for each participant and start a game. After 10^{100} rounds, the Guard tells you how many times your friend did handshake during the process. By carefully choosing the card sequence and analyzing the Guard's answers, you must determine your friend's number in as few queries as possible. The Guard only allows **up to Z queries**, otherwise you will be executed. (Note: submitting the final answer does **not** count as a query.)

Communication

The first line of input contains one integer T — the number of test cases.

Each test case proceeds as follows:

- First, read two integers N and D — the number of participants and the step size.
- Then, you can send queries in the format "? S" (without quotes), where S is a string of length N representing the cards given to each participant. The i-th character in S is the card of participant i.
- The jury responds with a single integer K — the number of times your friend shook hands during the process based on your card assignment.
- Once you are ready to submit your answer, print "! X", where X is your guess for your friend's number.

If you receive -1 as a response at any point, you must immediately terminate your program to avoid a wrong verdict.

To ensure correct communication with the interactive judge, **flush the output buffer after making every printing**:

- In C/C++: use `fflush(stdout)` or `cout.flush()`
- In Python: use `sys.stdout.flush()`

Constraints

- $1 \leq T \leq 100$
- $6 \leq N \leq 1500$
- $1 \leq D$
- $6D \leq N$
- $10 \leq Z \leq 20$

Subtasks

1. (6 points) The answer is either 1 or N; $Z = 20$
2. (13 points) $N \leq 15$; $Z = 20$
3. (42 points) $Z = 20$
4. (12 points) $Z = 15$
5. (13 points) $Z = 11$
6. (14 points) $Z = 10$

Example

Example 1

User	Jury	Explanation
	1	$T = 1$
	6 1	$N = 6$ and $D = 1$
? RRRLLL		First query
	5	Handshakes of your friend in first game
? LRRLRL		Second query
	3	Handshakes of your friend in second game
! 3		Final guess

- Initially, $T = 1$, $N = 6$, $D = 1$.
- The hidden participant's id is 3.
- The user sends the query `"RRRLLL"`. For this initial configuration, the friend shakes hands 5 times in total.
- The next query is `"LRRLRL"` — this time, 3 handshakes occur.
- The user submits `"! 3"` — the correct answer.

Attention: this example is not first test of task. You can find first test inside attachment.

Watchtowers

In the ancient city of Khiva there are N watchtowers, built during the era of past khanates and preserved to this day. For some mysterious reason all the towers stand on a single straight line. For convenience, imagine them numbered from 1 to N from **West** to **East**.

Each tower can be of one of the following three kinds:

- **Window facing West.** From such a tower every tower lying to its **west** is visible. In other words, if the window of tower i faces West, then it can see every tower j with $j < i$.
- **Window facing East.** From such a tower every tower lying to its **east** is visible. In other words, if the window of tower i faces East, then it can see every tower j with $j > i$.
- **No window.** No other tower is visible from this tower. Historians have yet to discover the purpose of such structures.

Nazarbek manages the towers. When a group of tourists arrives, they play a hide-and-seek game under two strict rules:

- **Distinct towers:** every tourist must choose a different tower—no sharing allowed.
- **Mutual invisibility:** after everyone has climbed their chosen tower, no selected tower may be visible from any other selected tower.

Nazarbek's goal on each day is to host the largest possible group that still satisfies these rules.

Because some towers are under renovation, on day k tourists may climb **only** the towers with indices in the interval $[L_k, R_k]$.

For each of the next Q days output the maximum possible size of a tourist group that Nazarbek can serve while respecting the game's rule.

Input

The first line contains a single integer T — the number of test cases.

For each test case:

The first line contains two integers N and Q — the number of towers and the number of days.
The second line contains a string S of length N. Character S_i is

- **L** if tower i has a window facing West;
- **R** if tower i has a window facing East;

- **A** if tower i has no window.

Each of the next Q lines contains two integers L_k and R_k — the inclusive interval of towers that may be used on day k.

Output

For each day print, on a new line, a single integer — the largest possible number of tourists that can be placed that day without breaking the rules.

Constraints

Let $\sum N$ be the sum of N over all test cases, and $\sum Q$ the sum of Q over all test cases.

- $1 \le T \le 2 \times 10^5$
- $1 \le \sum N, \sum Q \le 2 \times 10^5$
- $S_i \in \{L, R, A\}$
- $1 \le L_k \le R_k \le N$

Subtasks

1. (9 points) $S_i \in \{A\}$
2. (13 points) $S_i \in \{L, R\}$
3. (21 points) $Q = 1,\ L_1 = 1,\ R_1 = N$
4. (18 points) $N \le 1000$
5. (39 points) No additional constraints.

Examples

Example 1

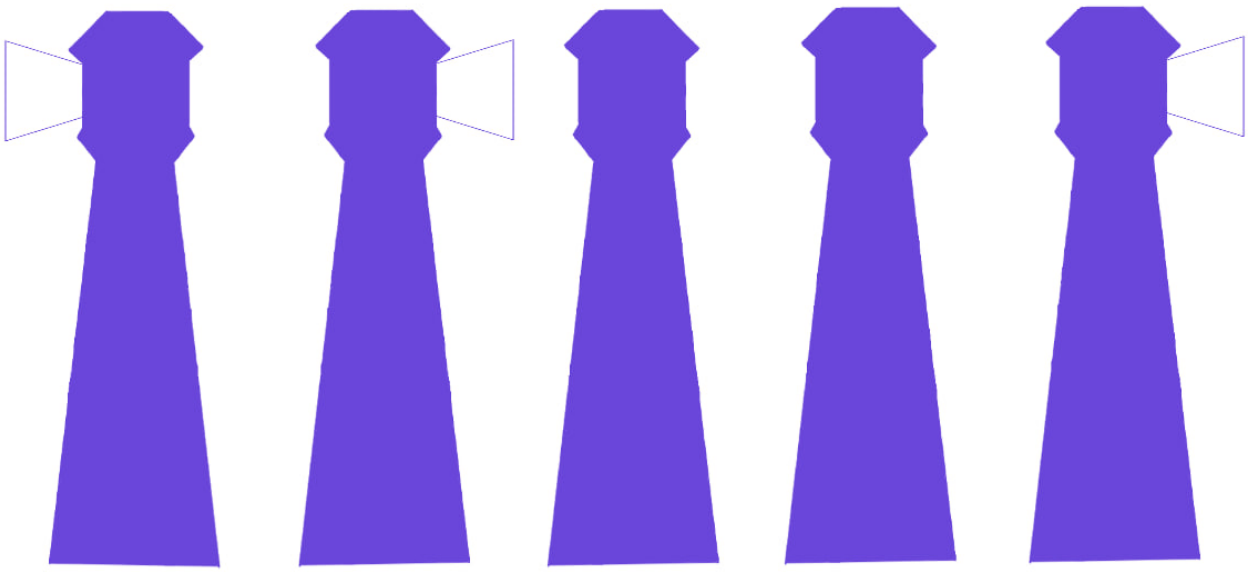
Input

```
3
5 3
LRAAR
1 4
3 4
2 5
4 1
AAAL
4 4
7 3
RRRRARRR
2 5
1 6
5 7
```

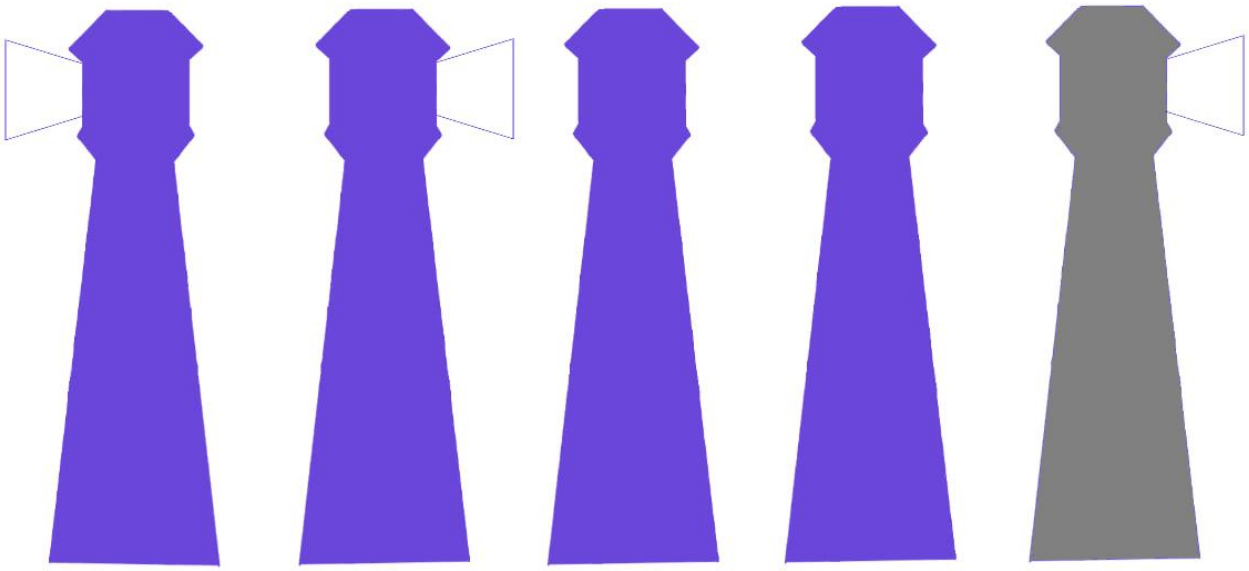
Output

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

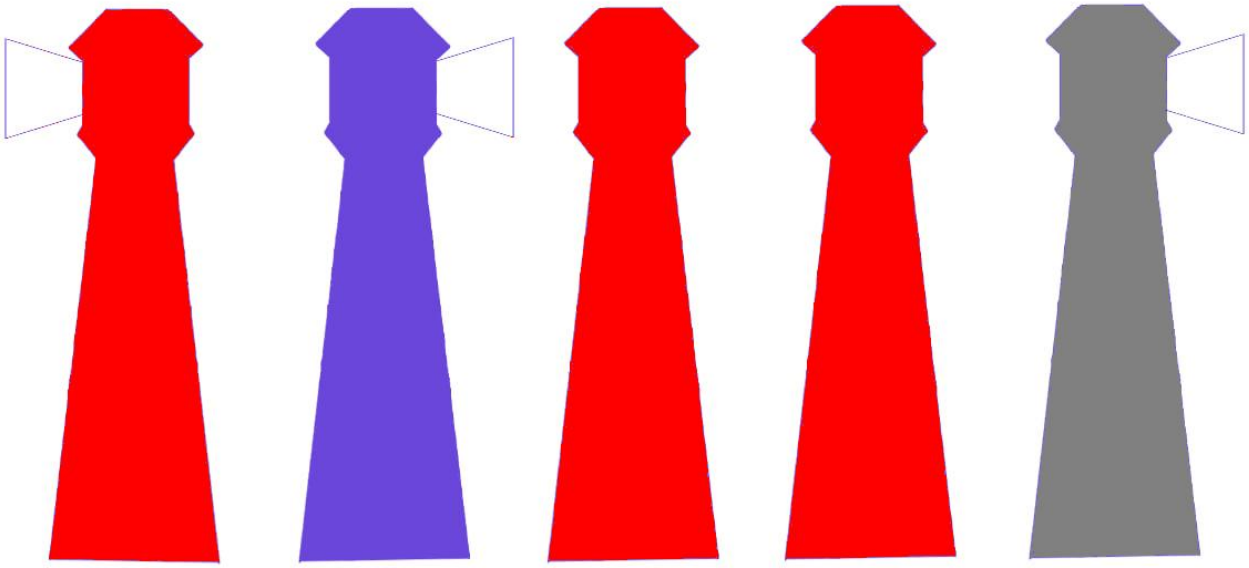
Explanation for the first test case: Here are the towers:



On day 1, only towers in the range $[1;4]$ can be used (purple):



Nazarbek can host a group with maximum size of 3. Without violating the game rules, tourists can be sent to the following towers (red):

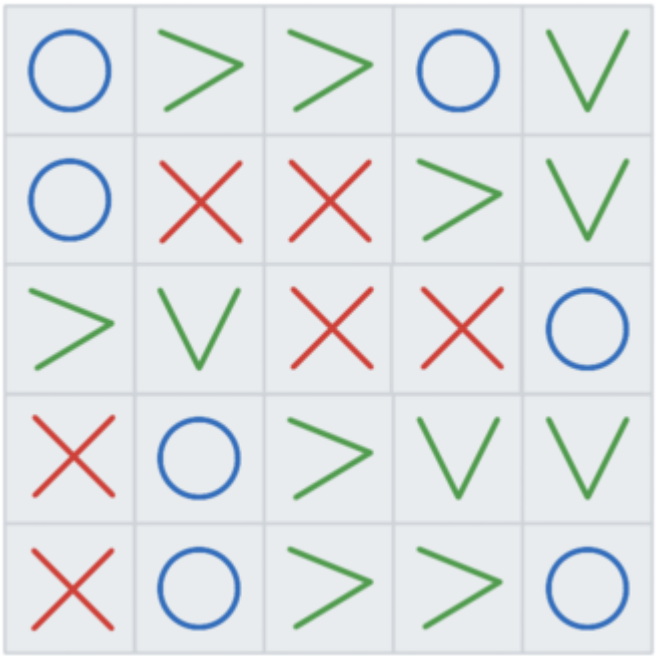


It can be proven that it is impossible to host a group of 4 or more tourists.

Hockeygrid

Firdavs found an ancient game of hockeygrid consisting of N rows and N columns. He found that each cell of the grid is of one of four types:

- **Right arrow:** If a player is inside that cell, he directly moves to the neighboring cell to its right
- **Down Arrow:** If the player is inside this cell, he moves to the neighboring cell to the bottom
- **Circle:** If the player is inside this cell, he can choose to move to the right neighboring cell or the bottom neighboring cell
- **Cross:** If a player is inside this cell, he loses



Hockeygrid of the sample.

Initially the player is in cell $(1, 1)$. To win the game, he must reach the destination cell (N, N) by following the above rules and staying within the grid.

Firdavs then decided to analyze this grid thoroughly. He found out some important observations about this grid:

- From the starting cell $(1, 1)$, one can reach every circle cell
- From each circle cell, you can reach the destination cell (N, N)
- The starting cell and the destination cell are guaranteed to be the circle cells

After that, Firdavs got bored. So he decided to make Q plans for himself. In each plan, Firdavs chooses some set of **circular** cells that he wants to visit in one pass. Note that each plan is considered independently, and he can visit these cells in any order.

Answer Firdavs for each of his plans, is there a path going through all the cells he has chosen.

Input

The first line contains a single integer T — the number of test cases.

For each test case:

The first line contains two integers N and Q — the size of the grid and the number of plans.
Then N lines follow, each consisting of N characters. In the i-th row and j columns will be the type of the cell (i, j):

- `>` Right Arrow
- `v` Down Arrow
- `o` Circle
- `x` Cross

Each of the next Q lines contains one integer k and $2 \cdot k$ integers $x_1\ y_1\ x_2\ y_2\ \ldots\ x_k\ y_k$. The cells (x_i, y_i), which should be visited during this plan.

Output

For each plan print, on a new line, `Yes` if it is possible to visit all important cells, otherwise `No`.

Constraints

Let $\sum N^2$ be the sum of N^2 over all test cases, and $\sum k$ the sum of k over all test cases and plans.

- $1 \le T \le 5 \times 10^4$
- $1 \le \sum N^2 \le 5 \times 10^5$
- $1 \le \sum k \le 5 \times 10^5$

Subtasks

1. (11 points) Grid fully consists of circle cells
2. (26 points) Number of the circle cells is at most 60
3. (63 points) No additional constraints

Examples

Example 1

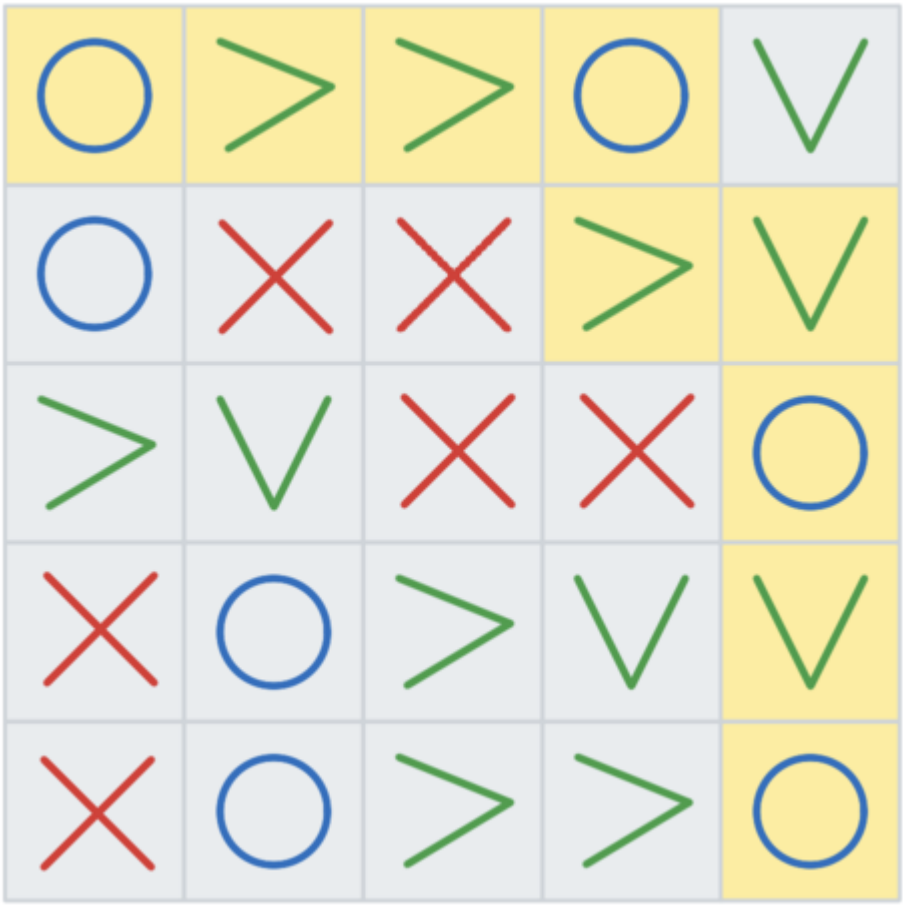
Input

```
1
5 2
o>>ov
oxx>v
>vxxo
xo>vv
xo>>o
2 1 4 3 5
4 2 1 4 2 5 2 1 4
```

Output

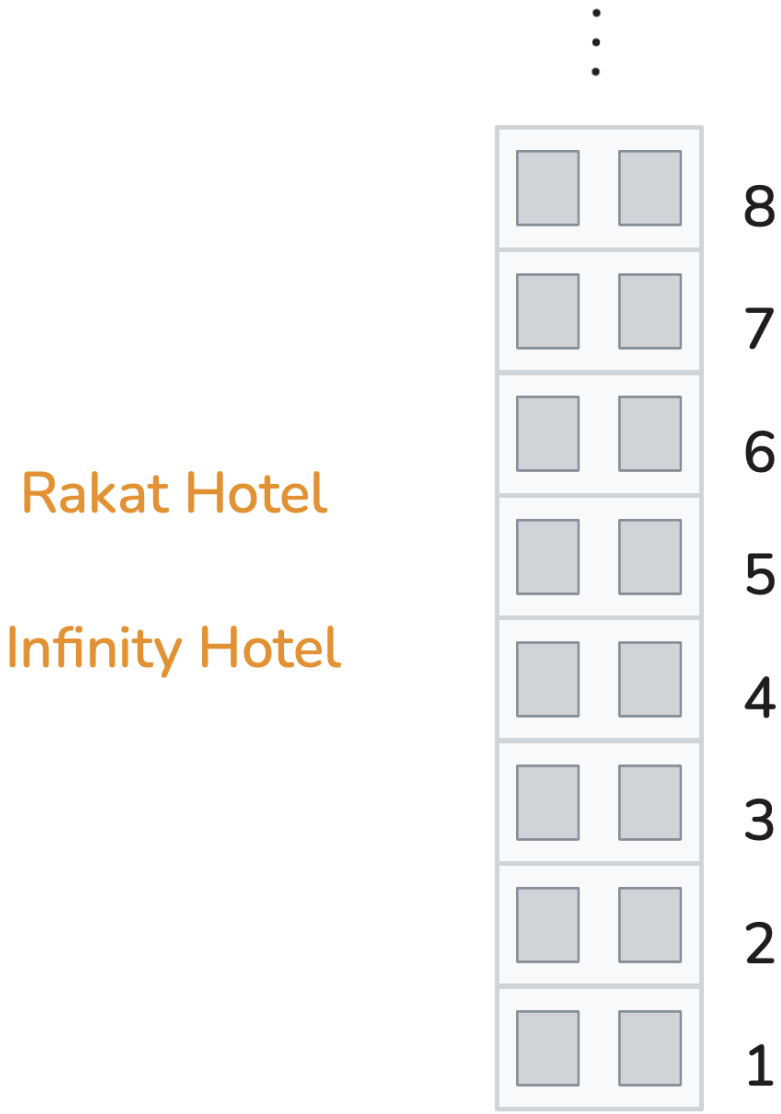
```
Yes
No
```

Explanation for the first test case: We can use the following route to pass through the cells $(1, 4)$ and $(3, 5)$



Rakathotel

Temur lives in an ancient city, and his dream was always to build an infinite hotel. But he was only able to build a hotel of 10^9 floors and named it “Rakat Hotel”.



A city is going to be visited by N tourists. Each tourist is assigned one integer a_i.

Tourists found out about “Rakat Hotel” and, having heard that all floors were empty, immediately decided to check in. Thus, some of the tourists are going to check in in a certain order. Let $b_1, b_2, \ldots, b_k$ be the integer numbers of tourists who are going to check into the hotel in this order.

That is, b_1 is the integer number of the person who will check in first, then b_2, etc. Note that check-in is not mandatory for everyone.

Then the i-person tries to get to the b_i-floor by moving up the floors one by one. If at some point the next floor is occupied, the tourist will not be able to move higher. For example, if array b has size $[4, 10, 1]$, then:

- The first person occupies floor 4
- The second person occupies floor 3, since he cannot move to the upper floor (floor 4 is already occupied, despite the fact that floor 10 is empty)
- The third person occupies floor 1

But unfortunately, this hotel had one strict rule:

- No empty floor between two occupied floors

The above example $b = [4, 10, 1]$ violates this rule, while $b = [4, 10, 2]$ does not.

Now help the tourists figure out what is the maximum number of tourists that can check into the hotel without breaking any rules, in a certain order.

In addition, you are given constant $T \in \{0, 1\}$:

- For $T = 0$ you only need to find the maximum number of tourists
- For $T = 1$ you need to find the order that gives the maximum number of tourists. By searching all possible orders, you need to find the lexicographical maximum array b

Input

The first line contains two integers N and T — the number of tourists and the constant T.
Second line contains N integers, the array a.

Output

Depending on the value of T:

- $T = 0$: Print one line, containing one integer k — the maximum number of tourists
- $T = 1$: Print one line, containing one integer k — the maximum number of tourists. In the second line print array b

Constraints

- $1 \leq N \leq 10^6$
- $1 \leq a_i \leq 10^9$

Subtasks

1. (7 points) $a_1 = a_2 = \ldots = a_N$
2. (10 points) $N \leq 8$
3. (22 points) $N \leq 10^4$
4. (21 points) $T = 0$
5. (15 points) $N \leq 10^5$
6. (25 points) No additional constraints

Examples

Example 1

Input

```
4 1
1 7 5 2
```

Output

```
2
5 7
```

Example 2

Input

```
3 0
3 5 2
```

Output

```
3
```

Explanation of the first test case:
A person with integer 5 checks in on floor 5. A second person with integer 7 checks in on floor 4, since he cannot get to the upper floor.

There is also another array $b = [2, 1]$ that gives the maximum number of tourists, but it is not a lexicographic maximum array.

Rakat Hotel

Infinity Hotel