Problem A. Pacman vs. Vampire

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Bakhtiar sir's Artificial Intelligence Lab is famous for its exciting 'Pacman' challenges. But some students, unwilling to risk their marks solving the tasks on the spot, have been secretly studying previous years' solutions passed down by seniors. However, this time Bakhtiar sir outsmarted them by introducing a brand new lab task — a twist on the classic Pacman game where the ghosts are replaced by Vampires.

The game proceeds in turns. In each turn, Pacman moves first, and then the vampires move one by one. Pacman will eat the food pellet immediately after reaching it. The game ends as soon as Pacman eats the food pellet and exits the game, or after 10^7 turns.

The input of the task is a two-dimensional grid world containing:

- Walkable empty spaces (`.`)
- Impassable walls (`#`)
- The starting position of Pacman (`P`)
- One food pellet (`F`)
- Vampires (`V`)

The moves available to Pacman are:

- `U` (Up): Pacman moves one cell up if the cell above is within the grid and not a wall. Otherwise, Pacman stays in place.
- `D` (Down): Pacman moves one cell down if the cell below is within the grid and not a wall. Otherwise, Pacman stays in place.
- `L` (Left): Pacman moves one cell to the left if the cell to the left is within the grid and not a wall. Otherwise, Pacman stays in place.
- `R` (Right): Pacman moves one cell to the right if the cell to the right is within the grid and not a wall. Otherwise, Pacman stays in place.
- `S` (Stay): Pacman remains in the current cell.
- `E` (Exit): Ends the game immediately if Pacman has already eaten the food pellet. Otherwise, Pacman stays in place.

The moves available to each Vampire are:

- `U` (Up): Vampire moves one cell up if the cell above is within the grid and not a wall. Otherwise, Vampire stays in place.
- `D` (Down): Vampire moves one cell down if the cell below is within the grid and not a wall. Otherwise, Vampire stays in place.
- `L` (Left): Vampire moves one cell to the left if the cell to the left is within the grid and not a wall. Otherwise, Vampire stays in place.
- `R` (Right): Vampire moves one cell to the right if the cell to the right is within the grid and not a wall. Otherwise, Vampire stays in place.

- `S` (Stay): Vampire remains in the current cell.

After the end of any turn, if a Vampire that hasn't bitten Pacman yet is in the same cell as Pacman, it will bite it. Each Vampire can bite Pacman at most once, but multiple Vampires can bite Pacman after the same turn.

The output of the lab task is a sequence of moves for Pacman. After a student submits his move sequence, Bakhtiar sir's AI will read it and control the Vampires optimally to maximize the number of Vampires that can bite Pacman before the game ends.

Pacman gets +**500 points** for eating the food pellet. However, each time a Vampire bites Pacman, **-10 points** is added to the score (Pacman loses 10 points). The final score can be negative.

To obtain full marks in the lab, a student must submit a move sequence that maximizes the final score, despite the Vampires moving optimally.

Tanbir, surprised by this new task, wants to solve it carefully. Before actually finding a move sequence, he wants to know the **maximum score** that is theoretically achievable for a given grid and initial position of Pacman and vampires. Your task is to calculate this.

Input

The first line of the input contains an integer t ($1 \leq t \leq 1000$) — the number of test cases.

The first line of each test case contains two space-separated integers n and m ($1 \leq n, m \leq 10^6$) — the number of rows and columns in the grid world.

The next n lines contain a string of length m — describing the grid world layout.

Each character in the grid is one of the following

- `'.'`: represents an empty space
- `'#'`: represents a wall
- `'P'`: represents Pacman's starting position
- `'F'`: represents the food pellet
- `'V'`: represents a Vampire

It is guaranteed that the grid for each test case contains exactly one 'P' and exactly one 'F', and the sum of $n \times m$ over all test cases does not exceed 10^6.

Output

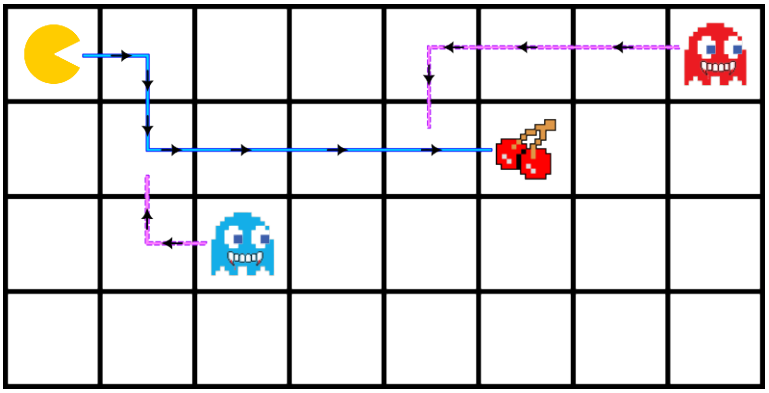
For each test case, print a single integer in a line — the maximum score achievable.

Example

standard input	standard output
8 4 8 P......V .....F.. ..V..... .... 4 8 PV...... ...V.... F....... ..V..... 4 8 PV...... #..V.... F....... ..V..... 5 6 P...#. ..#..# .V.#.F .#..#. V..... 3 2 #V PF #V 1 12 P..V.V...V.F 3 12 P#...#...#.F .#.#.#.#.#.# ...#...#...# 8 8 PVVVVVVV VVVVVVVV VVVVVVVV VVVVVVVV VVVVVVVV VVVVVVVV VVVVVVVV VVVVVVVF	480 500 470 490 480 470 500 -120

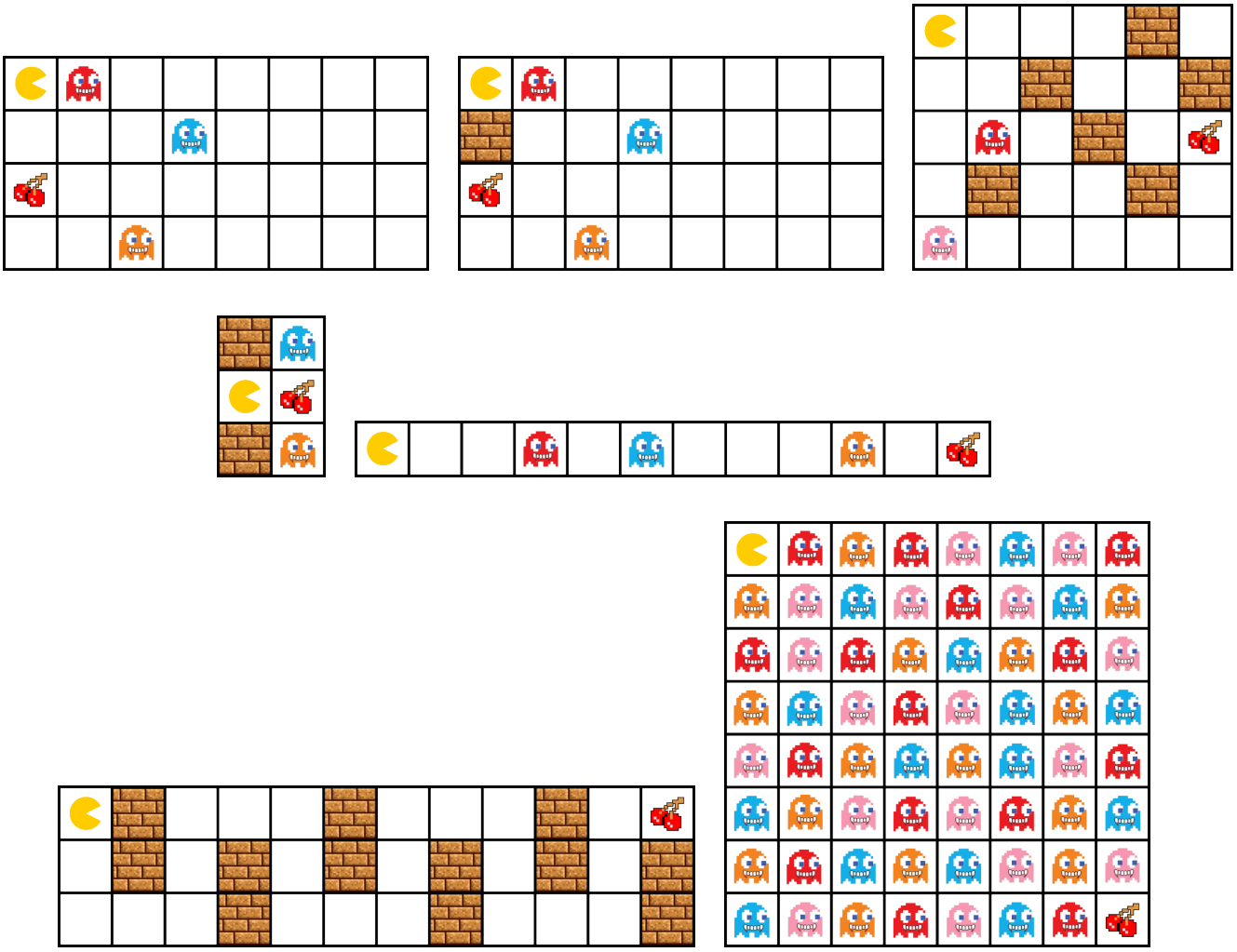
Note

The first test case is illustrated in the following diagram:



Here, one of the optimal move sequences for Pacman is `"RDRRRRE"`. The vampire in the first row (from the top) can meet Pacman with the move sequence `"LLLDXX"`. The vampire in the third row can meet Pacman with the move sequence `"LUXXXX"`. Here, `'X'` means any move.

The layouts of all other sample test cases are illustrated below:



Problem B. Roman Empire

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

After Roman Reigns' historic 1316-day title reign came to an end at WrestleMania 40, his cousin Dwayne "The Rock" Johnson decided to write a 1316-page book chronicling the reign.

To make it iconic, he wants to number each page using Roman numerals. Unfortunately, The Final Boss is not very good at counting.

You need to help him convert the page numbers from standard representation to Roman numerals.

Roman numerals are a number system that uses letters from the Latin alphabet to represent values. The basic symbols and their corresponding values are:

$$\texttt{I} = 1, \quad \texttt{V} = 5, \quad \texttt{X} = 10, \quad \texttt{L} = 50, \quad \texttt{C} = 100, \quad \texttt{D} = 500, \quad \texttt{M} = 1000$$

Roman numerals are usually written from largest to smallest. When this rule is followed strictly, the numeral is evaluated as a sum of its parts. This is called additive notation. For example: $1316 = 1000 + 100 + 100 + 100 + 10 + 5 + 1 = \texttt{MCCCXVI}$.

However, to avoid repeating the same symbol four or more times in a row, Roman numerals also allow a smaller value to appear before a larger one. In such cases, the smaller value is subtracted from the larger. This is called subtractive notation. Examples:
$\texttt{IV} = 5 - 1 = 4$, $\texttt{IX} = 10 - 1 = 9$, $\texttt{XL} = 50 - 10 = 40$, $\texttt{XC} = 100 - 10 = 90$, $\texttt{CD} = 500 - 100 = 400$, $\texttt{CM} = 1000 - 100 = 900$.

Using both additive and subtractive notation, the number 1248 can be expressed as `MCCXLVIII`.

Input

The first line contains a single integer t ($1 \le t \le 1316$) — the number of test cases.

Each test case consists of a single line containing a single integer n ($1 \le n \le 1316$) — the standard representation of the page number that needs to be converted.

Output

For each test case, output one string in a line — the Roman numeral representation of n.

Example

standard input	standard output
8 3 12 17 19 49 999 1248 1316	III XII XVII XIX XLIX CMXCIX MCCXLVIII MCCCXVI

Problem C. Cursed Queries

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2.5 seconds
Memory limit:	512 megabytes

You are given a cursed number m and an array a of length n.

You will then be given q queries. Each query is of one of the following two types:

1. `1 i x` — Assign $a_i \leftarrow x$.
2. `2 l r k` — Determine the k-goodness of the subarray $a[l \dots r]$.

Your task is to process all the queries and print the result for each type 2 query.

A number is called **good** if it is not divisible by the cursed number m.

A number x is called k**-good** if x remains good after adding k to it any number of times.

The k**-goodness** of an array a is the number of elements in the array that are k-good.

A *subarray* of an array is a contiguous portion of it. Formally, if a is an array of size n and $1 \le l \le r \le n$, an array $a[l \dots r] = [a_l, a_{l+1}, \dots, a_r]$ is called a subarray of a.

The k-goodness of the subarray $a[l \dots r]$ is the number of integers i $(l \le i \le r)$ such that a_i is k-good.

Input

The first line of the input contains a single integer t $(1 \le t \le 10^4)$ — the number of test cases.

The first line of each test case contains two space-separated integers n and m $(1 \le n \le 10^5, 1 \le m \le 5000)$ — the size of the array and the cursed number.

The second line of each test case contains n space-separated integers $a_1, a_2, \dots, a_n$ $(1 \le a_i \le 10^9)$ — the elements of the array.

The third line of each test case contains a single integer q $(1 \le q \le 10^5)$ — the number of queries. Each of the next q lines describes a query.

Each query is of one of the following two types:

1. `1 i x` $(1 \le i \le n, 1 \le x \le 10^9)$ — Assign $a_i \leftarrow x$.
2. `2 l r k` $(1 \le l \le r \le n, 1 \le k \le 10^9)$ — Determine the k-goodness of the subarray $a[l \dots r]$.

It is guaranteed that the sum of n over all test cases and the sum of q over all test cases do not exceed 10^5. It is also guaranteed that each test case contains at least one query of type 2.

Output

For each query of type 2, print a single line containing one integer — the k-goodness of the subarray $a[l \dots r]$.

Example

standard input	standard output
2 9 8 1 5 8 1 6 6 8 6 5 8 1 2 9 2 2 9 12 1 4 6 2 4 8 4 2 5 5 7 1 6 10 2 3 9 6 2 9 9 3 5 10 1 2 3 4 5 5 2 1 5 1 2 1 5 2 2 1 5 3 2 1 5 4 2 1 5 5	6 4 0 1 0 0 3 0 3 4

Note

In the first test case, the cursed number is 8.

After the first query, the array will become $[1, 9, 8, 1, 6, 6, 8, 6, 5]$.

For the second query, the 12-goodness of the subarray $[9, 8, 1, 6, 6, 8, 6, 5]$ is 6.

For the fourth query, the 4-goodness of the subarray $[6, 6, 6, 8, 6]$ is 4.

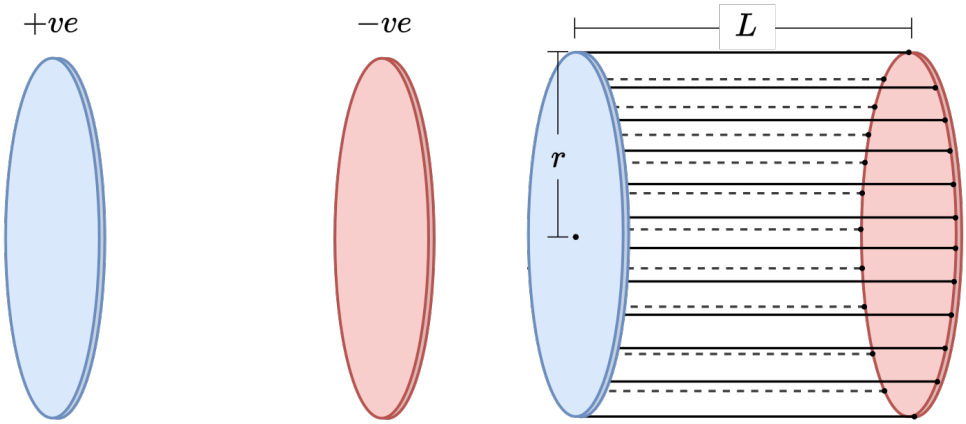
Problem D. Disruptor's Incapacitated Capacitor

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

You were already dead from the neck up, now from the neck down.

— Thrall, the *Disruptor*

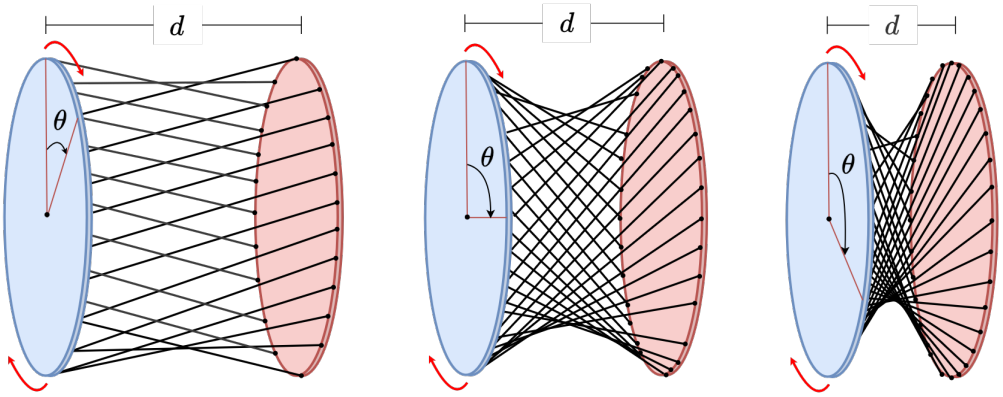
After winning a hard-fought battle against Ogre Magi, Thrall returned to his manufactory in upland Druud to repair his stormcrafting device. He was tinkering around with the plates of a faulty circular parallel plate capacitor that he retrieved from the heart of the device. Both of the circular plates have a uniform radius of r units. Thrall wanted to control the distance d between the two plates, and so he decided to join them by soldering many equally spaced strings of length L. One end of each string was soldered to the side of the $+ve$ plate, while the other end was soldered to the side of the $-ve$ plate.



(a) The circular plates of the capacitor. (b) Thrall's contraption.

Figure 1: Salvaging the spoils of war.

This contraption enabled him to rotate one of the circular plates (say, the $+ve$ plate) and consequently calibrate the value of d.



(a) $\theta = 60°$ (b) $\theta = 90°$ (c) $\theta = 120°$

Figure 2: Rotating the $+ve$ plate by different angles (θ).

If Thrall rotates the $+ve$ circular plate by an angle of θ degrees, what would be the corresponding value of d, the distance between the circular plates?

Input

The first line contains an integer t ($1 \leq t \leq 100$) — the number of test cases.

Each test case consists of a single line containing three space-separated integers r ($0 < r \leq 10^9$), L ($0 < L \leq 10^9$), and θ ($0° < \theta° < 180°$) — the radius of the plates, the length of the strings, and the angle (in degrees) by which Thrall rotates the $+ve$ plate respectively.

It is guaranteed that for all test cases, the string length L is sufficient to allow the rotation of the $+ve$ plate by θ degrees, resulting in a real and non-negative value of d.

Output

For each test case, output a single number d in a line — the distance between the circular plates.

The answer will be considered correct if the absolute error does not exceed 10^{-6}.

Examples

standard input	standard output
4 1 2 60 1 2 90 1 2 120 420 911 69	1.732051 1.414214 1.000000 776.886872
2 17 42 1 500000000 1000000000 179	41.998952 8726535.498374

Problem E. Queen of Diamonds

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

After a grueling Division 2 contest, Nayeem fell asleep. The next morning, he woke up in a strange place called Borderland, where he was forced to participate in a peculiar challenge: the "Queen of Diamonds" game!

In the game arena, there is a line of n bottles, each having a color label. Nayeem is armed with a special pistol. His task is to shoot and remove **some** (possibly none) bottles such that, in the remaining bottles, every color appears an even number of times.

Nayeem has to play multiple rounds. In each round, he will survive the game if and only if he can achieve this condition.

Your task is to determine the total number of different ways Nayeem can shoot the bottles. Two results are considered different if there is at least one bottle whose state (removed or kept) differs between them. Since the answer can be very large, output it modulo 1000000007.

Input

The first line of the input contains an integer r ($1 \le r \le 2 \times 10^4$) — the number of rounds Nayeem has to play. The descriptions of r rounds follow.

The first line of each round contains a single integer n ($1 \le n \le 10^6$) — the number of bottles.

The second line of each round contains a string s of length n consisting of lowercase English letters — describing the color of the bottles. The i-th character ($1 \le i \le n$) of the string represents the color of the i-th bottle.

It is guaranteed that the sum of n over all test cases does not exceed 2×10^6.

Output

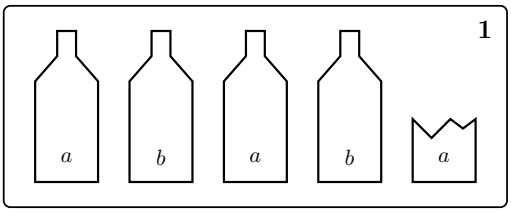
For each round, output a single integer in a line — the total number of ways Nayeem can beat the game modulo 1000000007.

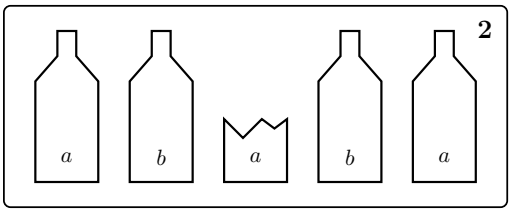
Example

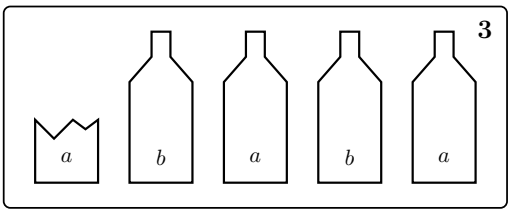
standard input	standard output
6 5 ababa 5 zzzzz 6 abcabc 7 missile 11 mississippi 31 thefiveboxingwizardsjumpquickly	8 16 8 4 128 32

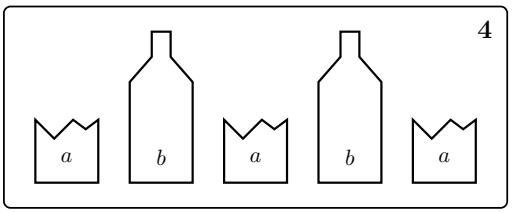
Note

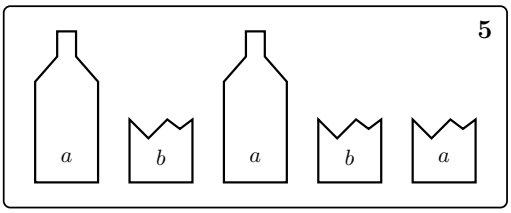
The 8 ways to beat the game for the first round are illustrated in the following diagram:

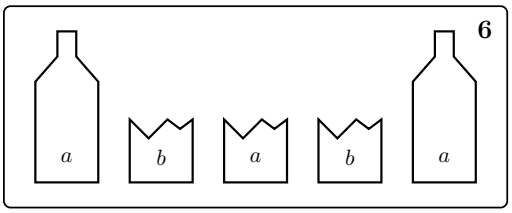

1
a
b
a
b
a

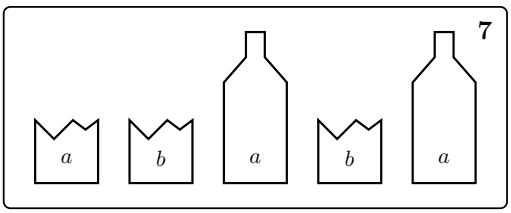

2
a
b
a
b
a

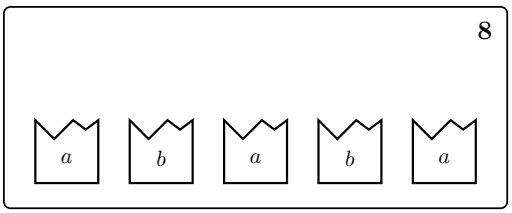

3
a
b
a
b
a


4
a
b
a
b
a


5
a
b
a
b
a


6
a
b
a
b
a


7
a
b
a
b
a


8
a
b
a
b
a

Problem F. Fertilize to Maximize

Input file:	`standard input`
Output file:	`standard output`
Time limit:	5 seconds
Memory limit:	256 megabytes

The garden at the Institute of Unbearable Torture (IUT) is known for its breathtaking beauty. Rows of vibrant flower columns line the pathways, each a testament to nature's charm. But lately, the greenery has started to fade.

There are n flower columns in the garden. Each column has a total of t_i flowers, out of which s_i are healthy and green. To restore the garden's full glory, the campus gardening team will plant m magical seeds. Each seed, when planted in a column, instantly grows into a new healthy flower.

The beauty score of a column is defined as: $\text{Score}_i = \dfrac{1}{t_i - s_i + 1}$

Your task is to determine the **maximum average beauty score** across all columns after planting the M seeds optimally.

Input

The first line contains an integer t ($1 \le t \le 10^5$) — the number of test cases.

Each test case consists of three lines:

The first line contains two integers n and m ($1 \le n \le 10^6,\ 0 \le m \le 10^9$) — the number of flower columns and the number of magical seeds to be planted.

The second line contains n integers $s_1, s_2, \ldots, s_n$ ($1 \le s_i \le t_i$) — the number of healthy flowers in each column.

The third line contains n integers $t_1, t_2, \ldots, t_n$ ($s_i \le t_i \le 10^9$) — the total number of flowers in each column.

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

Output

Print a single real number — the maximum average beauty score achievable after planting the M seeds optimally.

The answer will be considered correct if the absolute error does not exceed 10^{-6}.

Example

standard input	standard output
2 3 2 1 2 3 2 3 7 1 0 1 1	0.4 1

Note

In the first test case, one of the optimal ways to plant the two seeds is to plant one seed in column 1 and another in column 3. After that, the beauty scores of the columns will be $\dfrac{1}{3-2+1} = 0.5$, $\dfrac{1}{3-2+1} = 0.5$ and $\dfrac{1}{8-4+1} = 0.2$ respectively. The average beauty score will be 0.4.

Problem G. GCD vs. LCM

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

Faris loves to play with numbers. Today, he's looking for an interesting array. He defines an array of n positive integers to be **interesting** if it satisfies the following property:

$$\gcd(a_1, a_2, \ldots, a_n) < \gcd(a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n) \text{ for all integer } i \text{ where } 1 \leq i \leq n$$

In other words, the GCD of the entire array is **strictly less** than the GCD of any subset formed by removing exactly one element.

Being amazed by this property, Faris has challenged you, a famous mathematician, to construct any such interesting array of size n — but with an extra challenge: the LCM of the elements of the array cannot exceed l.

Given two integers n and l, your task is to construct such an array, or report that it is impossible.

The greatest common divisor or **GCD** of a set of positive integers is the largest positive integer that divides all of them. Formally,

$$\gcd(a_1, a_2, \ldots, a_n) = \max\Big\{d \in \mathbb{Z}^+ \;\Big|\; d \mid a_i \;\forall i \in \{1, 2, \ldots, n\}\Big\}$$

The least common multiple or **LCM** of a set of positive integers is the smallest positive integer that is divisible by all of them. Formally,

$$\text{lcm}(a_1, a_2, \ldots, a_n) = \min\Big\{m \in \mathbb{Z}^+ \;\Big|\; a_i \mid m \;\forall i \in \{1, 2, \ldots, n\}\Big\}$$

Input

The first line of the input contains a single integer t ($1 \leq t \leq 10^5$) — the number of test cases. Then t test cases follow.

Each test case consists of a single line containing two space-separated integers n ($2 \leq n \leq 10^5$) and l ($1 \leq l \leq 10^{18}$) — the size of the array to be constructed and the maximum allowed LCM.

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 space-separated positive integers — the elements of the interesting array.

If no such array exists, output -1.

Example

standard input	standard output
2 3 500 4 200	21 60 70 -1

Note

For the first test case, $\gcd(21, 60, 70) = 1$ and $\text{lcm}(21, 60, 70) = 420 < 500$. Here, $1 < \gcd(21, 60) = 3$, $1 < \gcd(21, 70) = 7$ and $1 < \gcd(60, 70) = 10$.

For the second test case, it can be proven that no interesting array exists under the given constraints.

Problem H. Pythagoras' Playhouse

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Little Pythagoras has a semicircular playhouse of radius r, where he loves to play with triangular toys. But he's very picky about his toys — he only plays with special ones he calls Pythagorean triangles.

A **Pythagorean triangle** is a non-degenerate triangle that has **one right angle** and **all three of its side lengths are integers**.

Today, Pythagoras is visiting the fair, which has toys of all possible shapes and sizes. Pythagoras wants to buy every Pythagorean triangle from the fair that can somehow fit entirely inside his playhouse.

Your task is to count the number of toys that Pythagoras will buy, *i.e.*, the number of different Pythagorean triangles that can somehow fit inside his playhouse (one triangle at a time).

Two triangles are considered the same if their three side lengths are the same when sorted in non-decreasing order. For example, triangles with sides $(3, 4, 5)$ and $(5, 3, 4)$ are considered the same because their sorted side lengths are the same, *i.e.*, $(3, 4, 5)$.

Input

The first line of the input contains a single integer t ($1 \le t \le 100$) — the number of test cases.

Each test case consists of a single line containing a single integer r ($1 \le r \le 10^7$) — the radius of the playhouse.

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

Output

For each test case, output a single integer in a line — the number of toys that Pythagoras will buy from the fair today.

Example

standard input	standard output
5 4 8 10 50 260	1 4 6 52 406

Note

For every right-angled triangle with side lengths (a, b, c) where c is the length of the hypotenuse, $a^2 + b^2 = c^2$. The converse is also true, *i.e.*, for a triangle with side lengths (a, b, c), if $a^2 + b^2 = c^2$, then the triangle is a right-angled triangle where c denotes the length of the hypotenuse.

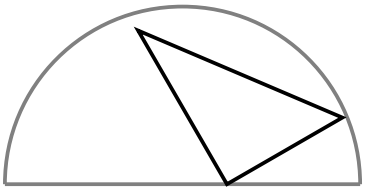
For example:

- A triangle with side lengths $(3, 4, 5)$ is a Pythagorean triangle.
- A triangle with side lengths $(3, 3, 3)$ is not — it has no right angle.
- A triangle with side lengths $(1, 1, \sqrt{2})$ is not — even though it has a right angle, one of the sides is not an integer.

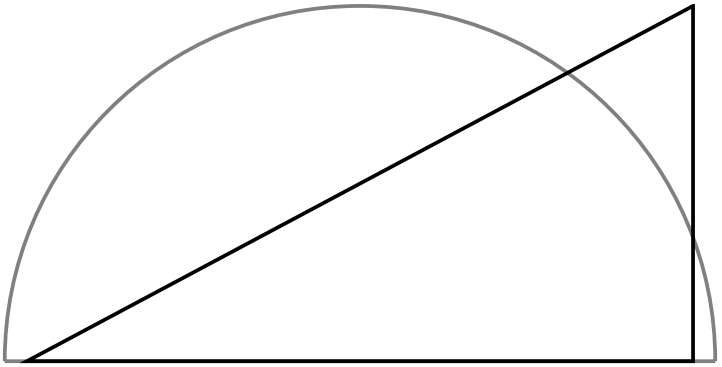
Although $(5, 12, 13)$ and $(6, 8, 10)$ are valid Pythagorean triangles, it can be proven that neither of them can fit inside a semicircle of radius 4 in any orientation.

For the first test case, the only Pythagorean triangle that fits inside the playhouse has sorted side lengths $(3, 4, 5)$.

Two examples from the first two test cases are illustrated in the following diagram:



(1) A triangle with side lengths $(3, 4, 5)$ can fit inside a playhouse of radius 4.



(2) A triangle with side lengths $(8, 15, 17)$ can't fit inside a playhouse of radius 8.

Problem I. Crisis In Flatland

Input file:	`standard input`
Output file:	`standard output`
Time limit:	5 seconds
Memory limit:	1024 megabytes

Flatland is on the verge of collapse.

Tensions between the East and the West have reached an all-time high, and civil unrest threatens to erupt at any moment. As a last-ditch effort to save the nation, both sides have agreed to a peace negotiation — and Pranto has been elected as the neutral mediator. The negotiations are scheduled to begin soon, but there's a problem: Pranto overslept. Woken up by a call from Shanto, Pranto realizes he has very little time left to reach the Negotiation Hall on the other side of Flatland. To make matters worse, private cars have been banned in Flatland to reduce traffic congestion. Now, Pranto must rely entirely on public transportation.

Flatland is organized as a grid of size $n \times m$, where each cell is a city. Pranto starts at city u, and the Negotiation Hall is at city v. A city is identified as a pair (r, c), representing the city in the r-th row and c-th column. Additionally, each city has a **boarding cost** $B_{r,c}$ and a **traffic jam** $J_{r,c}$.

There are two types of public transport in Flatland:

- **Trains** travel horizontally between any two cities in the same row. A train ride takes exactly the horizontal distance between the start and end cities. Formally, you can go from a city (r, c_1) to another city in the same row (r, c_2) in time $|c_2 - c_1|$ using a train.
- **Buses** travel vertically between any two cities in the same column. A bus ride takes the vertical distance between the start and end cities plus the total traffic jam from all intermediate cities on that column, **not including the endpoints**. Formally, you can go from a city (r_1, c) to another city in the same column (r_2, c) in time $|r_2 - r_1| + \sum_{r=\min(r_1,r_2)+1}^{\max(r_1,r_2)-1} J_{r,c}$.

To board a train or a bus at city (r, c), Pranto must pay a boarding cost of $B_{r,c}$. This cost is the same regardless of the destination or travel distance.

Your task is to help Pranto find the minimum total cost to reach the Negotiation Hall within at most T units of time, or report that it is **impossible**.

Input

The first line of the input contains a single integer t ($1 \le t \le 100$) — the number of test cases.

The first line of each test case contains seven space-separated integers n, m, r_u, c_u, r_v, c_v, and T ($2 \le n, m \le 500,\ 1 \le r_u, r_v \le n,\ 1 \le c_u, c_v \le m,\ 0 \le T \le 10^9$) — the number of rows and columns in Flatland, the row and column of the starting city u, the row and column of the destination city v, and the maximum allowed time.

The next n lines each contain m space-separated integers ($1 \le B_{r,c} \le 10^3$) — the boarding cost matrix. The c-th integer in the r-th line denotes $B_{r,c}$.

The following n lines each contain m space-separated integers ($0 \le J_{r,c} \le 10^9$) — the traffic jam matrix. The c-th integer in the r-th line denotes $J_{r,c}$.

It is guaranteed that the sum of $n \cdot m$ over all test cases does not exceed 10^3, and that the sum of all boarding costs in a single test case does not exceed 10^4.

Output

For each test case, output a single integer:

The minimum total boarding cost required to reach the destination city within at most T units of time. If it is not possible, output -1.

Example

standard input	standard output
6 3 3 1 1 3 3 4 1 100 100 1 1 100 100 1 1 0 0 0 0 0 0 0 0 0 3 3 1 1 3 3 4 1 1 1 100 1 100 1 1 1 0 100 0 0 0 0 0 100 0 3 3 1 1 3 3 4 1 1 1 1 1 1 1 1 1 0 0 0 100 0 100 0 0 0 3 3 1 1 3 3 4 1 1 1 100 1 100 1 1 1 0 0 0 100 100 100 0 0 0 3 3 1 1 3 3 4 1 1 1 100 100 100 1 1 1 0 0 0 100 100 100 0 0 0 3 3 1 1 3 3 3 1 1 1 100 100 100 1 1 1 0 0 0 100 100 100 0 0 0	4 2 3 4 102 -1

Problem J. Bit Lobon

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Given two integers L and R, your task is to compute the following three values:

- A: the **bitwise AND** of all integers from L to R inclusive.
- O: the **bitwise OR** of all integers from L to R inclusive.
- X: the **bitwise XOR** of all integers from L to R inclusive.

Formally, you need to compute:

$$A = L \ \& \ (L+1) \ \& \ (L+2) \ \& \ \ldots \ \& \ R$$

$$O = L \mid (L+1) \mid (L+2) \mid \ \ldots \ \mid R$$

$$X = L \ \oplus \ (L+1) \ \oplus \ (L+2) \ \oplus \ \ldots \ \oplus \ R$$

Bitwise operations work on the **binary representation** of integers, operating on bits of the same positions of the numbers involved.

- `&` denotes the bitwise AND operation — the result has a 1 only in bit positions where **both** numbers have a 1. For example: $6 = 0110_2,\ 10 = 1010_2,\ 6\ \&\ 10 = 0010_2 = 2$
- `|` denotes the bitwise OR operation — the result has a 1 in bit positions where **at least one** number has a 1. For example: $6 = 0110_2,\ 10 = 1010_2,\ 6 \mid 10 = 1110_2 = 14$
- $\oplus$ denotes the bitwise XOR operation — the result has a 1 in bit positions where the bits of the two numbers are **different**. For example: $6 = 0110_2,\ 10 = 1010_2,\ 6\ \oplus\ 10 = 1100_2 = 12$

Input

The first line of the input contains a single integer t ($1 \le t \le 2 \times 10^5$) — the number of test cases. Then t test cases follow.

Each test case consists of a single line containing two space-separated integers, L and R ($0 \le L \le R \le 10^{18}$) — the boundaries of the range (inclusive).

Output

For each test case, output three integers A, O, and X in a line — the bitwise AND, OR, and XOR of all integers from L to R inclusive, in order.

Example

standard input	standard output
5 8 12 0 5 1 10 16 32 100000000000 100000000000	8 15 12 0 7 1 0 15 11 0 63 32 100000000000 100000000000 100000000000

Note

The calculations of the first test case:

$$\begin{aligned} 8 &= 1000_2 \\ 9 &= 1001_2 \\ 10 &= 1010_2 \\ 11 &= 1011_2 \\ 12 &= 1100_2 \\ \hline A &= 1000_2 \\ O &= 1111_2 \\ X &= 1100_2 \end{aligned}$$

Problem K. Kaboom!

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

This is an interactive problem.

You are Professor Utonium, the brilliant scientist of Townsville. You were about to start your project of creating the 'perfect little girls'. Unfortunately, you discover that your lab assistant Jojo has removed all the labels from the chemical containers in the lab.

Now, you have N unlabeled chemicals, numbered from 1 to N. Among them are two very dangerous chemicals – Chemical X and Chemical Y. When these two are mixed in the same mixture, they react explosively.

To identify the exact indices of Chemical X and Chemical Y, you'll need to carry out a series of controlled experiments. In each experiment, you may mix a subset of the chemicals in a secure environment. If both Chemical X and Chemical Y are present in the mixture, it results in `"Kaboom"` – an explosion! If not, you'll simply observe a `"Noboom"`, meaning the mixture is stable.

But there's a catch — you can perform at most 20 experiments before the Mayor and Miss Bellum shut the lab down for safety violations!

Input

The first line of the input contains a single integer N ($2 \le N \le 1000$) — the number of chemicals in the lab.

Interaction Protocol

The interaction begins by reading the value of N.

After that, the experiments begin.

For performing an experiment, output $(m + 1)$ integers ($1 \le m \le N$) in a line in the following format (without the quotes):

- `"? m` c_1 c_2 $\ldots$ c_m`"` ($1 \le c_i \le N$, where $1 \le i \le m$)

Here, $\{c_1, c_2, \ldots, c_m\}$ are the indices of the m selected chemicals for the mixture. After each experiment, the judge will respond with:

- `"Kaboom"`, if the experiment causes an explosion, *i.e.*, the indices of Chemical X and Chemical Y are present in $\{c_1, c_2, \ldots, c_m\}$.
- `"Noboom"`, if the experiment doesn't cause an explosion.

After performing at most 20 experiments, you must output your final answer in a line in the following format (without the quotes):

- `"! x y"` ($1 \le x < y \le N$)

Here, x and y are the indices of the explosive chemicals.

After printing a line, do not forget to output the end of the line and flush the output. Otherwise, you will get `Idleness Limit Exceeded`. To flush the output, you can use:

- `fflush(stdout)` in C;

- `fflush(stdout)`, `std::cout.flush()` or `std::cout « std::flush` in C++;
- `System.out.flush()` in Java;
- `sys.stdout.flush()` in Python;
- `std::io::stdout().flush()` in Rust;
- Appropriate `flush` operation in any language.

Example

standard input	standard output
10 Noboom Kaboom Noboom Noboom Kaboom Noboom	 ? 3 1 2 3 ? 5 1 2 3 4 5 ? 2 4 5 ? 2 1 5 ? 3 2 3 4 ? 2 3 4 ! 2 4