A. Turtle Puzzle: Rearrange and Negate

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   256 megabytes
```

You are given an array a of n integers. You must perform the following two operations on the array (the first, then the second):

1. Arbitrarily rearrange the elements of the array or leave the order of its elements unchanged.
2. Choose at most one contiguous segment of elements and replace the signs of all elements in this segment with their opposites. Formally, you can choose a pair of indices l, r such that $1 \le l \le r \le n$ and assign $a_i = -a_i$ for all $l \le i \le r$ (negate elements). Note that you may choose not to select a pair of indices and leave all the signs of the elements unchanged.

What is the **maximum sum of the array elements** after performing these two operations (the first, then the second)?

Input

The first line of the input contains a single integer t ($1 \le t \le 1000$) — the number of test cases. The descriptions of the test cases follow.

The first line of each test case contains a single integer n ($1 \le n \le 50$) — the number of elements in array a.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($-100 \le a_i \le 100$) — elements of the array.

Output

For each test case, output the **maximum sum of the array elements** after sequentially performing the two given operations.

Standard Input	Standard Output

8
3
-2 3 -3
1
0
2
0 1
1
-99
4
10 -2 -3 7
5
-1 -2 -3 -4 -5
6
-41 22 -69 73 -15 -50
12
1 2 3 4 5 6 7 8 9 10 11 12
	8
	0
	1
	99
	22
	15
	270
	78

Note

In the first test case, you can first rearrange the array to get $[3, -2, -3]$ (operation 1), then choose $l = 2, r = 3$ and get the sum $3 + -((-2) + (-3)) = 8$ (operation 2).

In the second test case, you can do nothing in both operations and get the sum 0.

In the third test case, you can do nothing in both operations and get the sum $0 + 1 = 1$.

In the fourth test case, you can first leave the order unchanged (operation 1), then choose $l = 1, r = 1$ and get the sum $-(-99) = 99$ (operation 2).

In the fifth test case, you can first leave the order unchanged (operation 1), then choose $l = 2, r = 3$ and get the sum $10 + -((-2) + (-3)) + 7 = 22$ (operation 2).

In the sixth test case, you can first leave the order unchanged (operation 1), then choose $l = 1, r = 5$ and get the sum $-((-1) + (-2) + (-3) + (-4) + (-5)) = 15$ (operation 2).

B. Turtle Math: Fast Three Task

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   256 megabytes
```

You are given an array $a_1, a_2, \ldots, a_n$.

In one move, you can perform either of the following two operations:

- Choose an element from the array and remove it from the array. As a result, the length of the array decreases by 1;
- Choose an element from the array and increase its value by 1.

You can perform any number of moves. If the current array becomes empty, then no more moves can be made.

Your task is to find the **minimum** number of moves required to make the sum of the elements of the array a divisible by 3. It is possible that you may need 0 moves.

Note that the sum of the elements of an empty array (an array of length 0) is equal to 0.

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 a single integer n ($1 \le n \le 10^5$).

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^4$).

The sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output a single integer: the minimum number of moves.

Standard Input	Standard Output
8	1
4	0
2 2 5 4	0
3	1
1 3 2	1
4	2
3 7 6 8	1
1	1
1	
4	
2 2 4 2	
2	
5 5	
7	
2 4 8 1 9 3 4	
2	
4 10	

Note

In the first test case, initially the array $a = [2, 2, 5, 4]$. One of the optimal ways to make moves is:

- remove the current 4th element and get $a = [2, 2, 5]$;

As a result, the sum of the elements of the array a will be divisible by 3 (indeed, $a_1 + a_2 + a_3 = 2 + 2 + 5 = 9$).
In the second test case, initially, the sum of the array is $1 + 3 + 2 = 6$, which is divisible by 3. Therefore, no moves are required. Hence, the answer is 0.

In the fourth test case, initially, the sum of the array is 1, which is not divisible by 3. By removing its only element, you will get an empty array, so its sum is 0. Hence, the answer is 1.

C. Turtle Fingers: Count the Values of k

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

You are given three **positive** integers a, b and l ($a, b, l > 0$).

It can be shown that there always exists a way to choose **non-negative** (i.e. ≥ 0) integers k, x, and y such that $l = k \cdot a^x \cdot b^y$.

Your task is to find the number of distinct possible values of k across all such ways.

Input

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

The following t lines contain three integers, a, b and l ($2 \leq a, b \leq 100$, $1 \leq l \leq 10^6$) — description of a test case.

Output

Output t lines, with the i-th ($1 \leq i \leq t$) line containing an integer, the answer to the i-th test case.

Standard Input	Standard Output
`11`	`6`
`2 5 20`	`1`
`2 5 21`	`5`
`4 6 48`	`12`
`2 3 72`	`6`
`3 5 75`	`11`
`2 2 1024`	`24`
`3 7 83349`	`4`
`100 100 1000000`	`1`
`7 3 2`	`3`
`2 6 6`	`24`
`17 3 632043`	

Note

In the first test case, $a = 2, b = 5, l = 20$. The possible values of k (and corresponding x, y) are as follows:

- Choose $k = 1, x = 2, y = 1$. Then $k \cdot a^x \cdot b^y = 1 \cdot 2^2 \cdot 5^1 = 20 = l$.
- Choose $k = 2, x = 1, y = 1$. Then $k \cdot a^x \cdot b^y = 2 \cdot 2^1 \cdot 5^1 = 20 = l$.
- Choose $k = 4, x = 0, y = 1$. Then $k \cdot a^x \cdot b^y = 4 \cdot 2^0 \cdot 5^1 = 20 = l$.
- Choose $k = 5, x = 2, y = 0$. Then $k \cdot a^x \cdot b^y = 5 \cdot 2^2 \cdot 5^0 = 20 = l$.
- Choose $k = 10, x = 1, y = 0$. Then $k \cdot a^x \cdot b^y = 10 \cdot 2^1 \cdot 5^0 = 20 = l$.
- Choose $k = 20, x = 0, y = 0$. Then $k \cdot a^x \cdot b^y = 20 \cdot 2^0 \cdot 5^0 = 20 = l$.

In the second test case, $a = 2, b = 5, l = 21$. Note that $l = 21$ is not divisible by either $a = 2$ or $b = 5$. Therefore, we can only set $x = 0, y = 0$, which corresponds to $k = 21$.

In the third test case, $a = 4, b = 6, l = 48$. The possible values of k (and corresponding x, y) are as follows:

- Choose $k = 2, x = 1, y = 1$. Then $k \cdot a^x \cdot b^y = 2 \cdot 4^1 \cdot 6^1 = 48 = l$.
- Choose $k = 3, x = 2, y = 0$. Then $k \cdot a^x \cdot b^y = 3 \cdot 4^2 \cdot 6^0 = 48 = l$.
- Choose $k = 8, x = 0, y = 1$. Then $k \cdot a^x \cdot b^y = 8 \cdot 4^0 \cdot 6^1 = 48 = l$.
- Choose $k = 12, x = 1, y = 0$. Then $k \cdot a^x \cdot b^y = 12 \cdot 4^1 \cdot 6^0 = 48 = l$.
- Choose $k = 48, x = 0, y = 0$. Then $k \cdot a^x \cdot b^y = 48 \cdot 4^0 \cdot 6^0 = 48 = l$.

D. Turtle Tenacity: Continual Mods

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   256 megabytes
```

Given an array $a_1, a_2, \ldots, a_n$, determine whether it is possible to **rearrange its elements** into $b_1, b_2, \ldots, b_n$, such that $b_1 \bmod b_2 \bmod \ldots \bmod b_n \neq 0$.

Here $x \bmod y$ denotes the remainder from dividing x by y. Also, the modulo operations are calculated from left to right. That is, $x \bmod y \bmod z = (x \bmod y) \bmod z$. For example, $2024 \bmod 1000 \bmod 8 = (2024 \bmod 1000) \bmod 8 = 24 \bmod 8 = 0$.

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 a single integer n ($2 \le n \le 10^5$).

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$).

The sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output "YES" if it is possible, "NO" otherwise.

You can output the answer in any case (upper or lower). For example, the strings "yEs", "yes", "Yes", and "YES" will be recognized as positive responses.

Standard Input	Standard Output
8	YES
6	NO
1 2 3 4 5 6	YES
5	NO
3 3 3 3 3	YES
3	NO
2 2 3	YES
5	NO
1 1 2 3 7	
3	
1 2 2	
3	
1 1 2	
6	
5 2 10 10 10 2	
4	
3 6 9 3	

Note

In the first test case, rearranging the array into $b = [1, 2, 3, 4, 5, 6]$ (doing nothing) would result in $1 \bmod 2 \bmod 3 \bmod 4 \bmod 5 \bmod 6 = 1$. Hence it is possible to achieve the goal.

In the second test case, the array b must be equal to $[3, 3, 3, 3, 3]$, which would result in $3 \bmod 3 \bmod 3 \bmod 3 \bmod 3 = 0$. Hence it is impossible to achieve the goal.

In the third test case, rearranging the array into $b = [3, 2, 2]$ would result in $3 \bmod 2 \bmod 2 = 1$. Hence it is possible to achieve the goal.

,

E. Turtle vs. Rabbit Race: Optimal Trainings

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

Isaac begins his training. There are n running tracks available, and the i-th track ($1 \le i \le n$) consists of a_i equal-length sections.

Given an integer u ($1 \le u \le 10^9$), finishing each section can increase Isaac's ability by a certain value, described as follows:

- Finishing the 1-st section increases Isaac's performance by u.
- Finishing the 2-nd section increases Isaac's performance by $u - 1$.
- Finishing the 3-rd section increases Isaac's performance by $u - 2$.
- $\ldots$
- Finishing the k-th section ($k \ge 1$) increases Isaac's performance by $u + 1 - k$. (The value $u + 1 - k$ can be negative, which means finishing an extra section decreases Isaac's performance.)

You are also given an integer l. You must choose an integer r such that $l \le r \le n$ and Isaac will finish **each** section of **each** track $l, l + 1, \ldots, r$ (that is, a total of $\sum_{i=l}^{r} a_i = a_l + a_{l+1} + \ldots + a_r$ sections).

Answer the following question: what is the optimal r you can choose that the increase in Isaac's performance is maximum possible?

If there are multiple r that maximize the increase in Isaac's performance, output the **smallest** r.

To increase the difficulty, you need to answer the question for q different values of l and u.

Input

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

The descriptions of the test cases follow.

The first line contains a single integer n ($1 \le n \le 10^5$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^4$).

The third line contains a single integer q ($1 \le q \le 10^5$).

The next q lines each contain two integers l and u ($1 \le l \le n, 1 \le u \le 10^9$) — the descriptions to each query.

The sum of n over all test cases does not exceed $2 \cdot 10^5$. The sum of q over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output q integers: the i-th integer contains the optimal r for the i-th query. If there are multiple solutions, output the **smallest** one.

Standard Input	Standard Output

5
6
3 1 4 1 5 9
3
1 8
2 7
5 9
1
10
1
1 1
9
	3 4 5
	1
	9 2 9 4 9
	5 2 5 5 5 2 4 5 4 2
	10 6 9 7 7

```
5 10 9 6 8 3 10 7 3
5
8 56
1 12
9 3
1 27
5 45
5
7 9 2 5 2
10
1 37
2 9
3 33
4 32
4 15
2 2
4 2
2 19
3 7
2 7
10
9 1 6 7 6 3 10 7 3 10
5
10 43
3 23
9 3
6 8
5 14
```

Note

For the 1-st query in the first test case:

- By choosing $r = 3$, Isaac finishes $a_1 + a_2 + a_3 = 3 + 1 + 4 = 8$ sections in total, hence his increase in performance is $u + (u - 1) + \ldots + (u - 7) = 8 + 7 + 6 + 5 + 4 + 3 + 2 + 1 = 36$.
- By choosing $r = 4$, Isaac finishes $a_1 + a_2 + a_3 + a_4 = 3 + 1 + 4 + 1 = 9$ sections in total, hence his increase in performance is $u + (u - 1) + \ldots + (u - 8) = 8 + 7 + 6 + 5 + 4 + 3 + 2 + 1 + 0 = 36$.

Both choices yield the optimal increase in performance, however we want to choose the **smallest** r. So we choose $r = 3$.

For the 2-nd query in the first test case, by choosing $r = 4$, Isaac finishes $a_2 + a_3 + a_4 = 1 + 4 + 1 = 6$ sections in total, hence his increase in performance is $u + (u - 1) + \ldots + (u - 5) = 7 + 6 + 5 + 4 + 3 + 2 = 27$. This is the optimal increase in performance.

For the 3-rd query in the first test case:

- By choosing $r = 5$, Isaac finishes $a_5 = 5$ sections in total, hence his increase in performance is $u + (u - 1) + \ldots + (u - 4) = 9 + 8 + 7 + 6 + 5 = 35$.
- By choosing $r = 6$, Isaac finishes $a_5 + a_6 = 5 + 9 = 14$ sections in total, hence his increase in performance is $u + (u - 1) + \ldots + (u - 13) = 9 + 8 + 7 + 6 + 5 + 4 + 3 + 2 + 1 + 0 + (-1) + (-2) + (-3) + (-4) = 35$.

Both choices yield the optimal increase in performance, however we want to choose the **smallest** r. So we choose $r = 5$.

Hence the output for the first test case is $[3, 4, 5]$.

F. Turtle Mission: Robot and the Earthquake

```
Input file:     standard input
Output file:    standard output
Time limit:     3 seconds
Memory limit:   256 megabytes
```

The world is a grid with n rows and m columns. The rows are numbered $0, 1, \ldots, n-1$, while the columns are numbered $0, 1, \ldots, m-1$. In this world, the columns are **cyclic** (i.e. the top and the bottom cells in each column are adjacent). The cell on the i-th row and the j-th column ($0 \le i < n, 0 \le j < m$) is denoted as (i, j).

At time 0, the cell (i, j) (where $0 \le i < n, 0 \le j < m$) contains either a **rock** or **nothing**. The state of cell (i, j) can be described using the integer $a_{i,j}$:

- If $a_{i,j} = 1$, there is a rock at (i, j).
- If $a_{i,j} = 0$, there is nothing at (i, j).

As a result of aftershocks from the earthquake, the columns follow tectonic plate movements: each column moves cyclically **upwards** at a velocity of 1 cell per unit of time. Formally, for some $0 \le i < n, 0 \le j < m$, if (i, j) contains a rock at the moment, it will move from (i, j) to $(i-1, j)$ (or to $(n-1, j)$ if $i = 0$).

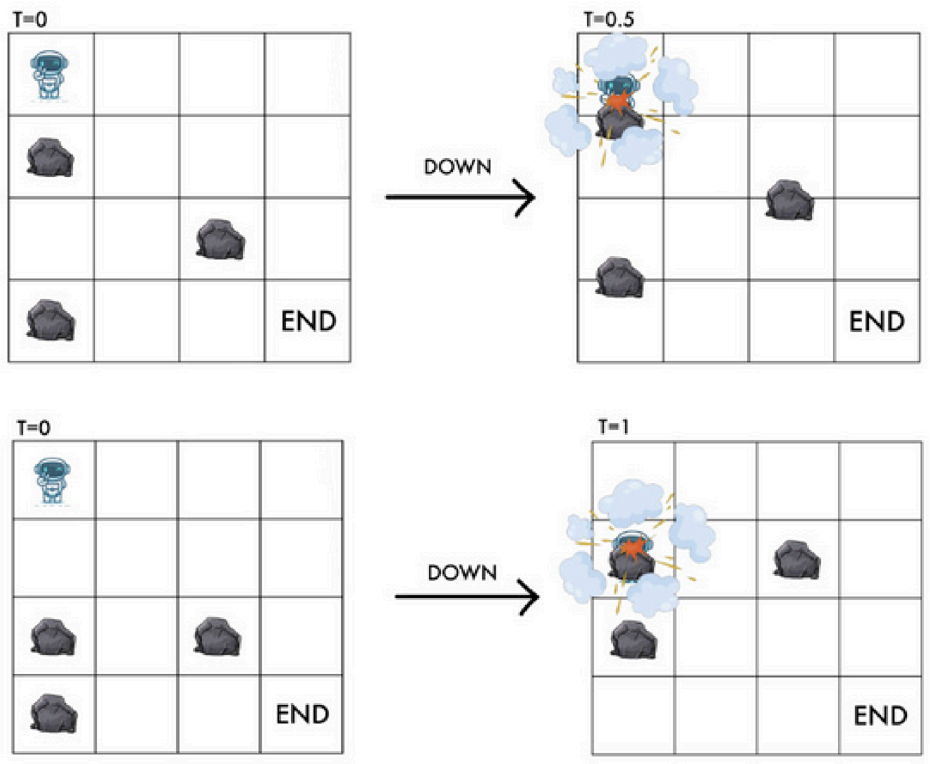
The robot called RT is initially positioned at $(0, 0)$. It has to go to $(n-1, m-1)$ to carry out an earthquake rescue operation (to the bottom rightmost cell). The earthquake doesn't change the position of the robot, they only change the position of rocks in the world.

Let RT's current position be (x, y) ($0 \le x < n, 0 \le y < m$), it can perform the following operations:

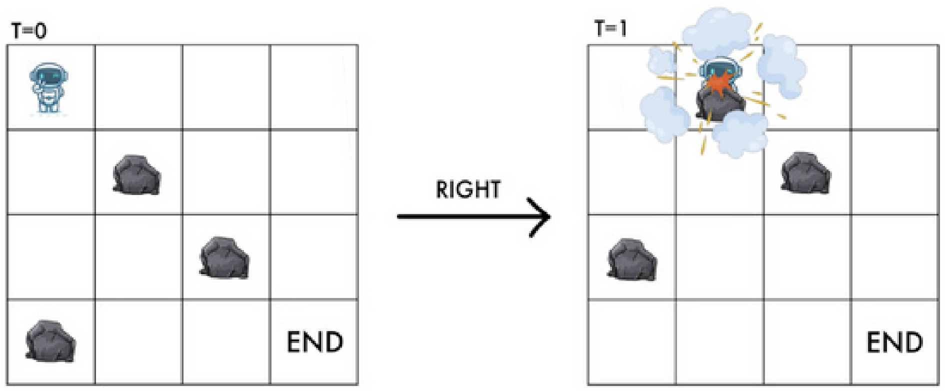
- Go one cell cyclically upwards, i.e. from (x, y) to $((x + n - 1) \bmod n, y)$ using 1 unit of time.
- Go one cell cyclically downwards, i.e. (x, y) to $((x + 1) \bmod n, y)$ using 1 unit of time.
- Go one cell to the right, i.e. (x, y) to $(x, y + 1)$ using 1 unit of time. (RT may perform this operation only if $y < m - 1$.)

Note that RT cannot go left using the operations nor can he stay at a position.

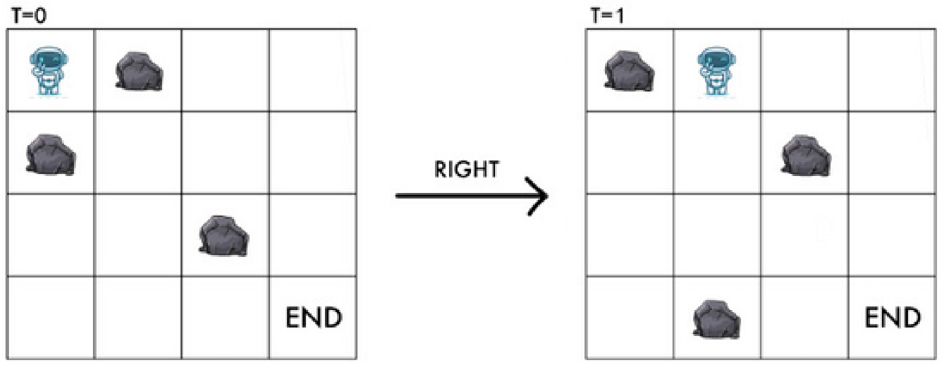
Unfortunately, RT will explode upon colliding with a rock. As such, when RT is at (x, y) and there is a rock at $((x + 1) \bmod n, y)$ or $((x + 2) \bmod n, y)$, RT cannot move down or it will be hit by the rock.



Similarly, if $y + 1 < m$ and there is a rock at $((x + 1) \bmod n, y + 1)$, RT cannot move right or it will be hit by the rock.



However, it is worth noting that if there is a rock at $(x \bmod n, y+1)$ and $((x+1) \bmod n, y)$, RT can still move right safely.



Find the minimum amount of time RT needs to reach $(n-1, m-1)$ without colliding with any rocks. If it is impossible to do so, output -1.

Input

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

In each test case, the first line contains two integers n, m ($3 \le n, m \le 10^3$) — the size of the planet's boundaries.

Each of the next n lines contains m integers. The $(j+1)$-th integer on the $(i+1)$-th line ($0 \le i < n, 0 \le j < m$) is $a_{i,j}$ ($0 \le a_{i,j} \le 1$), which denotes whether or not there is a rock at (i, j) at time 0.

Additionally, it is guaranteed that $a_{0,0} = 0$, and $a_{i,m-1} = 0$ for $0 \le i < n$. In other words, there is no rock at RT's initial position as well as column $m-1$.

The sum of $n \cdot m$ over all test cases does not exceed 10^6.

Output

For each test case:

- If the destination can be reached without colliding with any rocks, output a single integer — the minimum amount of time RT needs to reach $(n-1, m-1)$.
- Otherwise, output -1.

Standard Input	Standard Output

```
6
4 5
0 1 0 0 0
0 0 1 0 0
1 0 1 1 0
0 0 0 0 0
3 3
0 0 0
1 0 0
0 0 0
5 3
0 0 0
```
	```
	7
	3
	3
	8
	-1
	12
	```

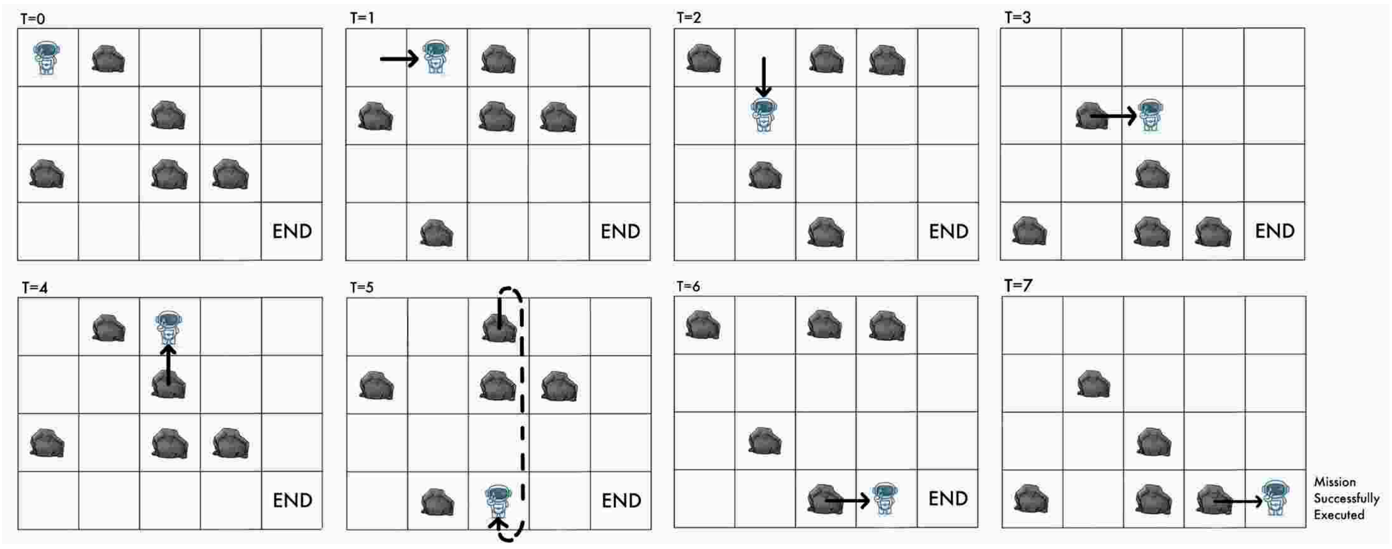
```
0 0 0
1 0 0
0 0 0
1 0 0
3 7
0 0 1 0 0 1 0
1 0 1 0 1 0 0
0 1 0 0 0 0 0
3 4
0 1 0 0
1 0 0 0
0 1 1 0
5 5
0 0 0 0 0
0 1 0 1 0
0 1 0 1 0
0 1 0 1 0
0 0 0 1 0
```

```
6
3 3
0 0 0
0 0 0
0 0 0
4 3
0 1 0
1 0 0
0 1 0
1 0 0
4 3
0 1 0
0 1 0
0 1 0
0 1 0
3 3
0 0 0
1 1 0
0 0 0
3 3
0 1 0
0 0 0
0 1 0
5 5
0 0 0 0 0
0 1 1 0 0
0 1 1 0 0
0 0 0 0 0
0 0 1 0 0
```

```
3
3
-1
-1
3
8
```

Note

Visual explanation of the first test case in the example:


T=0
END
T=1
END
T=2
END
T=3
END
T=4
END
T=5
END
T=6
END
T=7
Mission Successfully Executed

,

G. Turtle Magic: Royal Turtle Shell Pattern

```
Input file:     standard input
Output file:    standard output
Time limit:     3 seconds
Memory limit:   256 megabytes
```

Turtle Alice is currently designing a fortune cookie box, and she would like to incorporate the theory of LuoShu into it.

The box can be seen as an $n \times m$ grid ($n, m \geq 5$), where the rows are numbered $1, 2, \ldots, n$ and columns are numbered $1, 2, \ldots, m$. Each cell can either be **empty** or have a single fortune cookie of one of the following shapes: **circle** or **square**. The cell at the intersection of the a-th row and the b-th column is denoted as (a, b).

Initially, the entire grid is empty. Then, Alice performs q operations on the fortune cookie box. The i-th operation ($1 \leq i \leq q$) is as follows: specify a currently empty cell (r_i, c_i) and a shape (circle or square), then put a fortune cookie of the specified shape on cell (r_i, c_i). Note that after the i-th operation, the cell (r_i, c_i) is no longer empty.

Before all operations **and** after each of the q operations, Alice wonders what the number of ways to place fortune cookies in **all remaining empty cells** is, such that the following condition is satisfied:

No three consecutive cells (in horizontal, vertical, and both diagonal directions) contain cookies of the same shape. Formally:

- There does not exist any (i, j) satisfying $1 \leq i \leq n, 1 \leq j \leq m-2$, such that there are cookies of the same shape in cells $(i, j), (i, j+1), (i, j+2)$.
- There does not exist any (i, j) satisfying $1 \leq i \leq n-2, 1 \leq j \leq m$, such that there are cookies of the same shape in cells $(i, j), (i+1, j), (i+2, j)$.
- There does not exist any (i, j) satisfying $1 \leq i \leq n-2, 1 \leq j \leq m-2$, such that there are cookies of the same shape in cells $(i, j), (i+1, j+1), (i+2, j+2)$.
- There does not exist any (i, j) satisfying $1 \leq i \leq n-2, 1 \leq j \leq m-2$, such that there are cookies of the same shape in cells $(i, j+2), (i+1, j+1), (i+2, j)$.

You should output all answers modulo $998\,244\,353$. Also note that it is possible that after some operations, the condition is already not satisfied with the already placed candies, in this case you should output 0.

Input

The first line of the input contains a single integer t ($1 \leq t \leq 10^3$) — the number of test cases.

The first line of each test case contains three integers n, m, q ($5 \leq n, m \leq 10^9, 0 \leq q \leq \min(n \times m, 10^5)$).

The i-th of the next q lines contains two integers r_i, c_i and a single string shape_i ($1 \leq r_i \leq n, 1 \leq c_i \leq m, \text{shape}_i =$ `"circle"` or `"square"`), representing the operations. It is guaranteed that the cell on the r_i-th row and the c_i-th column is initially empty. That means, each (r_i, c_i) will appear at most once in the updates.

The sum of q over all test cases does not exceed 10^5.

Output

For each test case, output $q+1$ lines. The first line of each test case should contain the answer before any operations. The i-th line ($2 \leq i \leq q+1$) should contain the answer after the first $i-1$ operations. All answers should be taken modulo $998\,244\,353$.

Standard Input	Standard Output

```
2
6 7 4
3 3 circle
3 6 square
5 3 circle
5 4 square
5 5 3
1 1 circle
1 2 circle
1 3 circle
```

```
8
4
3
1
0
8
4
1
0
```

Note

In the second sample, after placing a circle-shaped fortune cookie to cells $(1, 1)$, $(1, 2)$ and $(1, 3)$, the condition is already not satisfied. Therefore, you should output 0.