A. Bus to Pénjamo

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

Ya vamos llegando a Péeeenjamoo
♬♬♬

There are n families travelling to Pénjamo to witness Mexico's largest-ever "walking a chicken on a leash" marathon. The i-th family has a_i family members. All families will travel using a single bus consisting of r rows with 2 seats each.

A person is considered *happy* if:

- Another family member is seated in the same row as them, or
- They are sitting alone in their row (with an empty seat next to them).

Determine the maximum number of happy people in an optimal seating arrangement. Note that **everyone** must be seated in the bus.

It is guaranteed that all family members will fit on the bus. Formally, it is guaranteed that $\sum_{i=1}^{n} a_i \leq 2r$.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \leq t \leq 1000$). The description of the test cases follows.

The first line of each test case contains two integers n and r ($1 \leq n \leq 100$; $1 \leq r \leq 500$) — the number of families and the number of rows in the bus.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 10$) — the number of family members in each family.

Output

For each test case, output the maximum number of happy people in an optimal seating arrangement.

Standard Input	Standard Output
4 3 3 2 3 1 3 3 2 2 2 4 5 1 1 2 2 4 5 3 1 1 3	4 6 6 6

Note

In the first test case, the two members of the first family can sit together in the first row, while the two members of the second family can sit together in the second row. The remaining member of the second family can sit in

the third row along with a member of the third family. This seating arrangement is shown below, where the 4 happy people are colored green.

1	1
2	2
2	3

In the second test case, a possible seating arrangement with 6 happy people is shown below.

3	3
1	1
2	2

In the third test case, a possible seating arrangement with 6 happy people is shown below.

4	4
	2
3	3
1	

,

B. Kar Salesman

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

Karel is a salesman in a car dealership. The dealership has n different models of cars. There are a_i cars of the i-th model. Karel is an excellent salesperson and can convince customers to buy up to x cars (of Karel's choice), as long as the cars are from different models.

Determine the minimum number of customers Karel has to bring in to sell all the cars.

Input

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

The first line of each test case contains two integers n and x ($1 \le n \le 5 \cdot 10^5$; $1 \le x \le 10$) — the number of different models of cars and the maximum number of cars Karel can convince a customer to buy.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$) — the number of cars of each model.

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

Output

For each test case, output the minimum possible number of customers needed to sell all the cars.

Standard Input	Standard Output
4 3 2 3 1 2 3 3 2 1 3 5 3 2 2 1 9 2 7 4 2 5 3 3 5 2 5	3 3 9 7

Note

For the first case, Karel only needs to lure in 3 customers. He will convince the customers to buy the following models of cars:

- Customer 1 buys 2 cars with model 1 and 3.
- Customer 2 buys 2 cars with model 1 and 2.
- Customer 3 buys 2 cars with model 1 and 3.

For the second case, Karel only needs to lure in 3 customers. He will convince the customers to buy the following models of cars:

- Customer 1 buys 2 cars with model 1 and 3.
- Customer 2 buys 3 cars with model 1, 2 and 3.

- Customer 3 buys 1 car with model 3.

,

C. Gerrymandering

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

We all steal a little bit. But I have only one hand, while my adversaries have two.

Álvaro Obregón

Álvaro and José are the only candidates running for the presidency of Tepito, a rectangular grid of 2 rows and n columns, where each cell represents a house. It is guaranteed that n is a multiple of 3.

Under the voting system of Tepito, the grid will be split into districts, which consist of any 3 houses that are connected*. Each house will belong to exactly one district.

Each district will cast a single vote. The district will vote for Álvaro or José respectively if at least 2 houses in that district select them. Therefore, a total of $\frac{2n}{3}$ votes will be cast.

As Álvaro is the current president, he knows exactly which candidate each house will select. If Álvaro divides the houses into districts optimally, determine the maximum number of votes he can get.

*A set of cells is connected if there is a path between any 2 cells that requires moving only up, down, left and right through cells in the set.

Input

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

The first line of each test case contains one integer n ($3 \le n \le 10^5$; n is a multiple of 3) — the number of columns of Tepito.

The following two lines each contain a string of length n. The i-th line contains the string s_i, consisting of the characters `A` and `J`. If $s_{i,j} =$ `A`, the house in the i-th row and j-th column will select Álvaro. Otherwise if $s_{i,j} =$ `J`, the house will select José.

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

Output

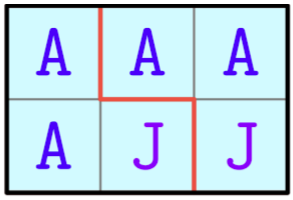
For each test case, output a single integer — the maximum number of districts Álvaro can win by optimally dividing the houses into districts.

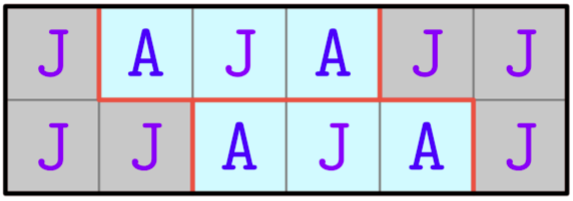
Standard Input	Standard Output
4 3 AAA AJJ 6 JAJAJJ JJAJAJ 6	2 2 3 2

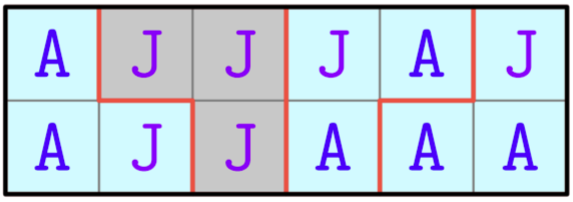
AJJJAJ AJJAAA 9 AJJJJAJAJ JAAJJJJJA	

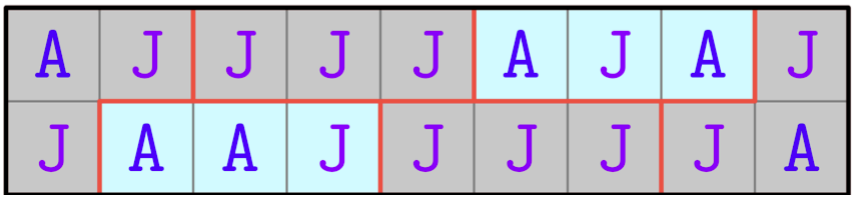
Note

The image below showcases the optimal arrangement of districts Álvaro can use for each test case in the example.









,

D1. Asesino (Easy Version)

```
Input file:     standard input
Output file:    standard output
Time limit:     2.5 seconds
Memory limit:   256 megabytes
```

This is the easy version of the problem. In this version, you can ask at most $n + 69$ questions. You can make hacks only if both versions of the problem are solved.

This is an interactive problem.

It is a tradition in Mexico's national IOI trainings to play the game "*Asesino*", which is similar to "*Among Us*" or "*Mafia*".

Today, n players, numbered from 1 to n, will play "*Asesino*" with the following three roles:

- **Knight**: a Knight is someone who always tells the truth.
- **Knave**: a Knave is someone who always lies.
- **Impostor**: an Impostor is someone everybody thinks is a Knight, but is secretly a Knave.

Each player will be assigned a role in the game. There will be **exactly one** Impostor but there can be any (possible zero) number of Knights and Knaves.

As the game moderator, you have accidentally forgotten the roles of everyone, but you need to determine the player who is the Impostor.

To determine the Impostor, you will ask some questions. In each question, you will pick two players i and j ($1 \le i, j \le n; i \ne j$) and ask if player i thinks that player j is a Knight. The results of the question is shown in the table below.

	Knight	Knave	Impostor
Knight	Yes	No	Yes
Knave	No	Yes	No
Impostor	No	Yes	—

The response of the cell in row a and column b is the result of asking a question when i has role a and j has row b. For example, the "Yes" in the top right cell belongs to row "Knight" and column "Impostor", so it is the response when i is a Knight and j is an Impostor.

Find the Impostor in at most $n + 69$ questions.

Note: the grader is adaptive: the roles of the players are not fixed in the beginning and may change depending on your questions. However, it is guaranteed that there exists an assignment of roles that is consistent with all previously asked questions under the constraints of this problem.

Input

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

The first line of each test case contains a single integer n ($3 \le n \le 10^5$) — the number of people playing the game.

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

Interaction

To ask a question, output a line in the following format:

- "? i j" ($1 \le i, j \le n, i \ne j$) — to ask player i if they think player j is a Knight.

The jury will output a "1" if player i thinks player j is a Knight, and "0" otherwise.

When you have determined which player the Impostor is, output a line in the following format:

- "! i" ($1 \le i \le n$) — the Impostor is player i.

Note that answering does not count to your limit of $n + 69$ questions.

If you have made an invalid output, used more than $n + 69$ questions or wrongly determined the Impostor, the jury will respond with "-1" and you will receive a **Wrong Answer** verdict. Upon receiving "-1", your program must terminate immediately. Otherwise, you may receive an arbitrary verdict because your solution might be reading from a closed stream.

After printing a query do not forget to output the end of the line and flush the output. Otherwise, you will get Idleness limit exceeded. To do this, use:

- fflush(stdout) or cout.flush() in C++;
- System.out.flush() in Java;
- sys.stdout.flush() in Python;
- std::io::stdout().flush() in Rust;
- see the documentation for other languages.

Hack format

For hacks, use the following format.

The first line should contain a single integer t — the number of test cases.

The first line of each test case should contain the integer n followed by the string "manual".

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($-1 \le a_i \le 1$) — the roles of each player. 1 denotes a Knight, 0 denotes a Knave, and -1 dentoes an Impostor. There must be exactly one Impostor, that is there must be exactly one index i such that $a_i = -1$.

As an example, the hack format for the example input is:

```
2
7 manual
0 1 0 -1 0 1 0
4 manual
0 1 -1 0
```

Standard Input	Standard Output
2 7 1	 ? 1 3 ? 7 6

0 0 1 1 0 0 1 4 0 1 1 1	? 2 5 ? 6 2 ? 4 5 ? 4 6 ? 1 4 ? 2 4 ! 4 ? 1 2 ? 2 3 ? 3 4 ? 4 1 ! 3

Note

Note that the example test cases do not represent an optimal strategy for asking questions and are only shown for the sake of demonstrating the interaction format. Specifically, we cannot determine which player is the Impostor from the questions asked in the examples.

In the first test case of the example, players at indices 2 and 6 are Knights, players at indices 1, 3, 5, and 7 are Knaves, and the Impostor is at index 4. The following is an explanation of the questions asked:

- In the first query, player i is a Knave and player j is a Knave. The answer is "yes" since Knaves always lie.
- In the second query, player i is a Knave and player j is a Knight. The answer is "no" since Knaves always lie.
- In the third query, player i is a Knight and player j is a Knave. The answer is "no" since Knights always tell the truth.
- In the fourth query, player i is a Knight and player j is a Knight. The answer is "yes" since Knights always tell the truth.
- In the fifth query, player i is a Impostor and player j is a Knave. The answer is "yes" since the Impostor always lies.
- In the sixth query, player i is a Impostor and player j is a Knight. The answer is "no" since the Impostor always lies.
- In the seventh query, player i is a Knave and player j is a Impostor. The answer is "no" since Knaves always lie and Knaves thinks that the Impostor is a Knight.
- In the eighth query, player i is a Knight and player j is a Impostor. The answer is "yes" since Knights always tell the truth and Knights think that the Impostor is a Knight.

,

D2. Asesino (Hard Version)

```
Input file:     standard input
Output file:    standard output
Time limit:     2.5 seconds
Memory limit:   256 megabytes
```

This is the hard version of the problem. In this version, you must use the minimum number of queries possible. You can make hacks only if both versions of the problem are solved.

This is an interactive problem.

It is a tradition in Mexico's national IOI trainings to play the game "*Asesino*", which is similar to "*Among Us*" or "*Mafia*".

Today, n players, numbered from 1 to n, will play "*Asesino*" with the following three roles:

- **Knight**: a Knight is someone who always tells the truth.
- **Knave**: a Knave is someone who always lies.
- **Impostor**: an Impostor is someone everybody thinks is a Knight, but is secretly a Knave.

Each player will be assigned a role in the game. There will be **exactly one** Impostor but there can be any (possible zero) number of Knights and Knaves.

As the game moderator, you have accidentally forgotten the roles of everyone, but you need to determine the player who is the Impostor.

To determine the Impostor, you will ask some questions. In each question, you will pick two players i and j ($1 \leq i, j \leq n; i \neq j$) and ask if player i thinks that player j is a Knight. The results of the question is shown in the table below.

	Knight	Knave	Impostor
Knight	Yes	No	Yes
Knave	No	Yes	No
Impostor	No	Yes	—

The response of the cell in row a and column b is the result of asking a question when i has role a and j has row b. For example, the "Yes" in the top right cell belongs to row "Knight" and column "Impostor", so it is the response when i is a Knight and j is an Impostor.

Find the Impostor in the minimum number of queries possible. That is, let $f(n)$ be the minimum integer such that for n players, there exists a strategy that can determine the Impostor using at most $f(n)$ questions. Then, you should use at most $f(n)$ questions to determine the Impostor.

Note: the grader is adaptive: the roles of the players are not fixed in the beginning and may change depending on your questions. However, it is guaranteed that there exists an assignment of roles that is consistent with all previously asked questions under the constraints of this problem.

Input

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

The first line of each test case contains a single integer n ($3 \le n \le 10^5$) — the number of people playing the game.

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

Interaction

To ask a question, output a line in the following format:

- "? i j" ($1 \le i, j \le n, i \ne j$) — to ask player i if they think player j is a Knight.

The jury will output a "1" if player i thinks player j is a Knight, and "0" otherwise.

When you have determined which player the Impostor is, output a line in the following format:

- "! i" ($1 \le i \le n$) — the Impostor is player i.

Note that answering does not count to your limit of $f(n)$ questions.

If you have made an invalid output, used more than $f(n)$ questions or wrongly determined the Impostor, the jury will respond with "-1" and you will receive a **Wrong Answer** verdict. Upon receiving "-1", your program must terminate immediately. Otherwise, you may receive an arbitrary verdict because your solution might be reading from a closed stream.

After printing a query do not forget to output the end of the line and flush the output. Otherwise, you will get `Idleness limit exceeded`. To do this, use:

- `fflush(stdout)` or `cout.flush()` in C++;
- `System.out.flush()` in Java;
- `sys.stdout.flush()` in Python;
- `std::io::stdout().flush()` in Rust;
- see the documentation for other languages.

Hack format

For hacks, use the following format.

The first line should contain a single integer t — the number of test cases.

The first line of each test case should contain the integer n followed by the string "manual".

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($-1 \le a_i \le 1$) — the roles of each player. 1 denotes a Knight, 0 denotes a Knave, and -1 dentoes an Impostor. There must be exactly one Impostor, that is there must be exactly one index i such that $a_i = -1$.

As an example, the hack format for the example input is:

```
2
7 manual
0 1 0 -1 0 1 0
4 manual
0 1 -1 0
```

Standard Input	Standard Output

2 7 1 0 0 1 1 0 0 4 0 1 1 1	 ? 1 3 ? 7 6 ? 2 5 ? 6 2 ? 4 5 ? 4 6 ? 1 4 ! 4 ? 1 2 ? 2 3 ? 3 4 ? 4 1 ! 3

Note

Note that the example test cases do not represent an optimal strategy for asking questions and are only shown for the sake of demonstrating the interaction format. Specifically, we cannot determine which player is the Impostor from the questions asked in the examples.

In the first test case of the example, players at indices 2 and 6 are Knights, players at indices 1, 3, 5, and 7 are Knaves, and the Impostor is at index 4. The following is an explanation of the questions asked:

- In the first query, player i is a Knave and player j is a Knave. The answer is "yes" since Knaves always lie.
- In the second query, player i is a Knave and player j is a Knight. The answer is "no" since Knaves always lie.
- In the third query, player i is a Knight and player j is a Knave. The answer is "no" since Knights always tell the truth.
- In the fourth query, player i is a Knight and player j is a Knight. The answer is "yes" since Knights always tell the truth.
- In the fifth query, player i is a Impostor and player j is a Knave. The answer is "yes" since the Impostor always lies.
- In the sixth query, player i is a Impostor and player j is a Knight. The answer is "no" since the Impostor always lies.
- In the seventh query, player i is a Knave and player j is a Impostor. The answer is "no" since Knaves always lie and Knaves thinks that the Impostor is a Knight.

,

E1. Billetes MX (Easy Version)

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

This is the easy version of the problem. In this version, it is guaranteed that $q = 0$. You can make hacks only if both versions of the problem are solved.

An integer grid A with p rows and q columns is called *beautiful* if:

- All elements of the grid are integers between 0 and $2^{30} - 1$, and
- For any subgrid, the XOR of the values at the corners is equal to 0. Formally, for any four integers i_1, i_2, j_1, j_2 ($1 \le i_1 < i_2 \le p$; $1 \le j_1 < j_2 \le q$), $A_{i_1,j_1} \oplus A_{i_1,j_2} \oplus A_{i_2,j_1} \oplus A_{i_2,j_2} = 0$, where $\oplus$ denotes the bitwise XOR operation.

There is a partially filled integer grid G with n rows and m columns where only k cells are filled. Polycarp wants to know how many ways he can assign integers to the unfilled cells so that the grid is beautiful.

However, Monocarp thinks that this problem is too easy. Therefore, he will perform q updates on the grid. In each update, he will choose an unfilled cell and assign an integer to it. Note that these updates are **persistent**. That is, changes made to the grid will apply when processing future updates.

For each of the $q + 1$ states of the grid, the initial state and after each of the q queries, determine the number of ways Polycarp can assign integers to the unfilled cells so that the grid is beautiful. Since this number can be very large, you are only required to output their values modulo $10^9 + 7$.

Input

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

The first line of each test case contains four integers n, m, k and q ($2 \le n, m \le 10^5$; $0 \le k \le 10^5$; $q = 0$) — the number of rows, the number of columns, the number of fixed cells, and the number of updates.

The following k lines contain three integers r, c and v ($1 \le r \le n, 1 \le c \le m$; $0 \le v < 2^{30}$) — indicating that $G_{r,c}$ is assigned the integer v.

The following q lines contain three integers r, c and v ($1 \le r \le n, 1 \le c \le m$; $0 \le v < 2^{30}$) — indicating that $G_{r,c}$ is assigned the integer v.

It is guaranteed that the pairs (r, c) over all assignments are distinct.

It is guaranteed that the sum of n, m, k and q over all test cases does not exceed 10^5 respectively.

Output

For each test case, output $q + 1$ lines. The i-th line of output should contain the answer of the i-th state of the grid modulo $10^9 + 7$.

Standard Input	Standard Output
2 3 3 8 0 2 1 6 3 2 12	1 489373567

1 2 6 2 2 0 1 3 10 1 1 0 2 3 12 3 1 10 2 5 2 0 1 1 10 1 2 30	

Note

In the first test case of the example, we have the following grid:

0	6	10
6	0	12
10	12	$?$

It can be proven that the only valid value for tile $(3, 3)$ is 0.

,

E2. Billetes MX (Hard Version)

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

This is the hard version of the problem. In this version, it is guaranteed that $q \leq 10^5$. You can make hacks only if both versions of the problem are solved.

An integer grid A with p rows and q columns is called *beautiful* if:

- All elements of the grid are integers between 0 and $2^{30}-1$, and
- For any subgrid, the XOR of the values at the corners is equal to 0. Formally, for any four integers i_1, i_2, j_1, j_2 ($1 \leq i_1 < i_2 \leq p$; $1 \leq j_1 < j_2 \leq q$), $A_{i_1,j_1} \oplus A_{i_1,j_2} \oplus A_{i_2,j_1} \oplus A_{i_2,j_2} = 0$, where $\oplus$ denotes the bitwise XOR operation.

There is a partially filled integer grid G with n rows and m columns where only k cells are filled. Polycarp wants to know how many ways he can assign integers to the unfilled cells so that the grid is beautiful.

However, Monocarp thinks that this problem is too easy. Therefore, he will perform q updates on the grid. In each update, he will choose an unfilled cell and assign an integer to it. Note that these updates are **persistent**. That is, changes made to the grid will apply when processing future updates.

For each of the $q+1$ states of the grid, the initial state and after each of the q queries, determine the number of ways Polycarp can assign integers to the unfilled cells so that the grid is beautiful. Since this number can be very large, you are only required to output their values modulo 10^9+7.

Input

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

The first line of each test case contains four integers n, m, k and q ($2 \leq n, m \leq 10^5$; $0 \leq k, q \leq 10^5$) — the number of rows, the number of columns, the number of fixed cells, and the number of updates.

The following k lines contain three integers r, c and v ($1 \leq r \leq n, 1 \leq c \leq m$; $0 \leq v < 2^{30}$) indicating that $G_{r,c}$ is assigned the integer v.

The following q lines contain three integers r, c and v ($1 \leq r \leq n, 1 \leq c \leq m$; $0 \leq v < 2^{30}$) indicating that $G_{r,c}$ is assigned the integer v.

It is guaranteed that the pairs (r, c) over all assignments are distinct.

It is guaranteed that the sum of n, m, k and q over all test cases does not exceed 10^5 respectively.

Output

For each test case, output $q+1$ lines. The i-th line of output should contain the answer of the i-th state of the grid modulo 10^9+7.

Standard Input	Standard Output
3 3 3 8 1 2 1 6 3 2 12	1 0 489373567 651321892

1 2 6 2 2 0 1 3 10 1 1 0 2 3 12 3 1 10 3 3 1 2 5 2 0 1 1 10 1 2 30 2 5 0 2 1 1 10 1 2 30	769740174 489373567

Note

In the first test case of the example, we initially have the following grid:

0	6	10
6	0	12
10	12	$?$

It can be proven that the only valid value for tile $(3, 3)$ is 0, so the first answer is 1. For the second query, the grid does not satisfy the condition, and thus the answer is 0.