A. Object Identification

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

This is an interactive problem.

You are given an array $x_1, \ldots, x_n$ of integers from 1 to n. The jury also has a fixed but hidden array $y_1, \ldots, y_n$ of integers from 1 to n. The elements of array y are **unknown** to you. Additionally, it is known that for all i, $x_i \neq y_i$, and all pairs (x_i, y_i) are distinct.

The jury has secretly thought of one of two objects, and you need to determine which one it is:

- **Object A**: A directed graph with n vertices numbered from 1 to n, and with n edges of the form $x_i \to y_i$.
- **Object B**: n points on a coordinate plane. The i-th point has coordinates (x_i, y_i).

To guess which object the jury has thought of, you can make queries. In one query, you must specify two numbers i, j $(1 \leq i, j \leq n, i \neq j)$. In response, you receive one number:

- If the jury has thought of **Object A**, you receive the length of the shortest path (in edges) from vertex i to vertex j in the graph, or 0 if there is no path.
- If the jury has thought of **Object B**, you receive the Manhattan distance between points i and j, that is $|x_i - x_j| + |y_i - y_j|$.

You have 2 queries to determine which of the objects the jury has thought of.

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.

Interaction

The interaction begins with reading n ($3 \leq n \leq 2 \cdot 10^5$) — the length of the arrays x and y at the start of each test case.

Next, read n integers: $x_1, x_2, \ldots, x_n$ ($1 \leq x_i \leq n$) — the elements of array x.

It is guaranteed that the sum of n across all test cases does not exceed $2 \cdot 10^5$.

The array $y_1, y_2, \ldots, y_n$ is **fixed** for each test case. In other words, the interactor is not adaptive.

It is guaranteed that $x_i \neq y_i$ for all $1 \leq i \leq n$, and all pairs (x_i, y_i) are distinct.

To make a query, output "? i j" (without quotes, $1 \leq i, j \leq n, i \neq j$). Then you must read the response to the query, which is a non-negative integer.

To give an answer, output "! A" if **Object A** is hidden or "! B" if **Object B** is hidden (without quotes). Note that printing the answer does not count as one of the 2 queries.

After printing each query do not forget to output the end of line and flush* the output. Otherwise, you will get `Idleness limit exceeded` verdict. If, at any interaction step, you read -1 instead of valid data, your solution must exit immediately. This means that your solution will reveive `Wrong answer` because of an

invalid query or any other mistake. Failing to exit can result in an arbitrary verdict because your solution will continue to read from a closed stream. Note that if the query is correct, the response will never be -1.

Hacks

The first line must contain an integer t ($1 \le t \le 1000$) — the number of test cases.

The first line of each test case must contain an integer n ($3 \le n \le 2 \cdot 10^5$) — the length of the hidden array.

The second line of each test case must contain n integers — $x_1, x_2, \ldots, x_n$ ($1 \le x_i \le n$).

The third line of each test case must contain n integers — $y_1, y_2, \ldots, y_n$ ($1 \le y_i \le n$).

It must be ensured that $x_i \neq y_i$ for all $1 \le i \le n$, and all pairs (x_i, y_i) must be distinct.

The fourth line must contain one character: A or B, indicating which of the objects you want to hide.

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

---

*To flush, use:

- `fflush(stdout)` or `cout.flush()` in C++;
- `sys.stdout.flush()` in Python;
- see the documentation for other languages.

Standard Input	Standard Output
2 3 2 2 3 1 0 5 5 1 4 2 3 4 4	? 2 3 ? 1 2 ! A ? 1 5 ? 5 1 ! B

Note

In the first test case, $x = [2, 2, 3]$, $y = [1, 3, 1]$ and Object A is guessed.

In the second test case, $x = [5, 1, 4, 2, 3]$, $y = [3, 3, 2, 4, 1]$ and Object B is guessed.

,

B. White Magic

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

We call a sequence $a_1, a_2, \ldots, a_n$ *magical* if for all $1 \leq i \leq n-1$ it holds that: $\min(a_1, \ldots, a_i) \geq \operatorname{mex}(a_{i+1}, \ldots, a_n)$. In particular, any sequence of length 1 is considered *magical*.

The minimum excluded (MEX) of a collection of integers $a_1, a_2, \ldots, a_k$ is defined as the smallest non-negative integer t which does not occur in the collection a.

You are given a sequence a of n non-negative integers. Find the maximum possible length of a *magical* subsequence* of the sequence a.

*A sequence a is a subsequence of a sequence b if a can be obtained from b by the deletion of several (possibly, zero or all) element from arbitrary positions.

Input

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

The first line of each test case contains an integer n ($1 \leq n \leq 2 \cdot 10^5$) — the length of the sequence a.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($0 \leq a_i \leq 10^9$) — the elements of the sequence a.

It is guaranteed that the sum of n across all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output a single number — the maximum possible length of a *magical* subsequence of the sequence a.

Standard Input	Standard Output
8 5 4 3 2 1 0 6 4 3 3 2 1 0 4 2 0 1 2 1 777 4 1000000000 1 7 9 2 0 1 2 1 2 4	5 5 3 1 4 2 2 3

0 1 0 1	

Note

In the first test case, the sequence $[4, 3, 2, 1, 0]$ is *magical*, since:

- $\min(4) = 4, \operatorname{mex}(3, 2, 1, 0) = 4$. $4 \geq 4$
- $\min(4, 3) = 3, \operatorname{mex}(2, 1, 0) = 3$. $3 \geq 3$
- $\min(4, 3, 2) = 2, \operatorname{mex}(1, 0) = 2$. $2 \geq 2$
- $\min(4, 3, 2, 1) = 1, \operatorname{mex}(0) = 1$. $1 \geq 1$

In the second test case, the sequence $[4, 3, 3, 2, 1, 0]$ is not *magical*, since $\min(4, 3) = 3, \operatorname{mex}(3, 2, 1, 0) = 4$, $3 < 4$. However, the subsequence $[4, 3, 2, 1, 0]$ of this sequence is *magical*, so the answer is 5.

,

C. Bitwise Slides

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

You are given an array $a_1, a_2, \ldots, a_n$. Also, you are given three variables P, Q, R, initially equal to zero.

You need to process all the numbers $a_1, a_2, \ldots, a_n$, **in the order from** 1 **to** n. When processing the next a_i, you must perform **exactly** one of the three actions of your choice:

1. $P := P \oplus a_i$
2. $Q := Q \oplus a_i$
3. $R := R \oplus a_i$

$\oplus$ denotes the bitwise XOR operation.

When performing actions, you must follow the *main rule*: it is necessary that after each action, all three numbers P, Q, R are **not** pairwise distinct.

There are a total of 3^n ways to perform all n actions. How many of them do not violate the *main rule*? Since the answer can be quite large, find it modulo $10^9 + 7$.

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 an integer n ($1 \le n \le 2 \cdot 10^5$) — the length of the array a.

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 elements of the array a.

It is guaranteed that the sum of the values of n for all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output the number of ways to perform all n actions without violating the *main rule*, modulo $10^9 + 7$.

Standard Input	Standard Output
5 3 1 7 9 4 179 1 1 179 5 1 2 3 3 2 12 8 2 5 3 9 1 8 12 9 9 9 4 1 1000000000	3 9 39 123 3

Note

In the first test case, there are 3 valid sequences of operations: **PPP**, **QQQ**, **RRR**.

In the second test case, there are 9 valid sequences of operations: **PPPP**, **PPPQ**, **PPPR**, **QQQP**, **QQQQ**, **QQQR**, **RRRP**, **RRRQ**, **RRRR**.

,

D1. Club of Young Aircraft Builders (easy version)

Input file: standard input
Output file: standard output
Time limit: 4 seconds
Memory limit: 1024 megabytes

This is the easy version of the problem. The difference between the versions is that in this version, all $a_i = 0$. You can hack only if you solved all versions of this problem.

There is an n-story building, with floors numbered from 1 to n from bottom to top. There is exactly one person living on each floor.

All the residents of the building have a very important goal today: to launch at least c paper airplanes collectively. The residents will launch the airplanes in turn. When a person from the i-th floor launches an airplane, all residents on the floors from 1 to i can see it as it descends to the ground.

If, from the perspective of the resident on the i-th floor, at least c airplanes have already been launched, they will **not** launch any more airplanes themselves. It is also known that by the end of the day, from the perspective of each resident in the building, at least c airplanes have been launched, and a total of m airplanes were thrown.

You carefully monitored this flash mob and recorded which resident from which floor threw each airplane. Unfortunately, the information about who exactly threw some airplanes has been lost. Find the number of ways to fill in the gaps so that the information could be credible. Since the answer can be quite large, output it modulo $10^9 + 7$.

In this version of the problem, all information has been lost, and the entire array consists of gaps.

It is also possible that you made a mistake in your records, and there is no possible way to restore the gaps. In that case, the answer is considered to be 0.

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 three integers n, c, m ($1 \le n \le 100$, $1 \le c \le 100$, $c \le m \le n \cdot c$) — the number of floors in the building, the minimum required number of airplanes, and the number of airplanes actually launched.

The second line of each test case contains m integers $a_1, a_2, \ldots, a_m$ ($0 \le a_i \le n$) — a_i indicates the resident from which floor launched the i-th airplane; $a_i = 0$ indicates a gap.

In this version of the problem, it is guaranteed that all $a_i = 0$.

It is guaranteed that the sum of the values of m across all test cases does not exceed 10^4.

Output

For each test case, output the number of ways to fill in the gaps with numbers from 1 to n, so that the chronology of the airplane launches could be credible, modulo $10^9 + 7$.

Standard Input	Standard Output

2 3 2 4 0 0 0 0 5 5 7 0 0 0 0 0 0 0	6 190

Note

In the first test example, all six possible ways to fill in the gaps are as follows:

1. $[1, 1, 3, 3]$
2. $[1, 2, 3, 3]$
3. $[1, 3, 2, 3]$
4. $[2, 1, 3, 3]$
5. $[2, 2, 3, 3]$
6. $[3, 1, 2, 3]$

Note that the array $[2, 3, 1, 3]$ **is not** a valid way to fill in the gaps, as the third airplane could not have been launched by the person on the 1st floor, since from their perspective, $c = 2$ airplanes had already been launched.

Also, the array $[1, 1, 2, 3]$ **is not** a valid way to fill in the gaps, as from the perspective of the person on the 3rd floor, only 1 airplane has been launched, while $c = 2$.

,

D2. Club of Young Aircraft Builders (hard version)

Input file: standard input
Output file: standard output
Time limit: 4 seconds
Memory limit: 1024 megabytes

This is the hard version of the problem. The difference between the versions is that in this version, not necessary $a_i = 0$. You can hack only if you solved all versions of this problem.

There is a building with n floors, numbered from 1 to n from bottom to top. There is exactly one person living on each floor.

All the residents of the building have an important goal today: to launch at least c paper airplanes collectively. The residents will launch the airplanes in turn. When a person from the i-th floor launches an airplane, all residents on floors from 1 to i can see it as it descends to the ground. If, from the perspective of the resident on the i-th floor, at least c airplanes have already been launched, they will no longer launch airplanes themselves. It is also known that by the end of the day, from the perspective of each resident in the building, at least c airplanes have been launched, and a total of m airplanes were thrown.

You have been carefully monitoring this flash mob, and for each airplane, you recorded which resident from which floor threw it. Unfortunately, the information about who exactly threw some of the airplanes has been lost. Find the number of ways to fill in the gaps so that the information could be credible. Since the answer could be quite large, output it modulo $10^9 + 7$.

It is also possible that you made a mistake in your records, and there is no possible way to restore the gaps. In that case, the answer is considered to be 0.

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 three integers n, c, m ($1 \le n \le 100$, $1 \le c \le 100$, $c \le m \le n \cdot c$) — the number of floors in the building, the minimum required number of airplanes, and the number of airplanes actually launched.

The second line of each test case contains m integers $a_1, a_2, \ldots, a_m$ ($0 \le a_i \le n$) — a_i indicates the resident from which floor launched the i-th airplane; $a_i = 0$ indicates a gap.

It is guaranteed that the sum of the values of m across all test cases does not exceed 10^4.

Output

For each test case, output the number of ways to fill in the gaps with numbers from 1 to n, so that the chronology of the airplane launches could be credible, modulo $10^9 + 7$.

Standard Input	Standard Output
8 3 2 4 0 0 0 0 5 5 7 0 0 0 0 0 0 0 6 1 3	6 190 3 2 0 0

2 0 0 2 3 5 0 0 1 0 2 3 3 4 3 3 3 0 2 1 2 0 1 2 1 2 0 2 5 3 12 0 0 1 0 2 4 0 0 0 5 0 5	1 14

Note

In the first test example, all six possible ways to fill in the gaps are as follows:

1. $[1, 1, 3, 3]$
2. $[1, 2, 3, 3]$
3. $[1, 3, 2, 3]$
4. $[2, 1, 3, 3]$
5. $[2, 2, 3, 3]$
6. $[3, 1, 2, 3]$

Note that the array $[2, 3, 1, 3]$ **is not** a valid way to fill in the gaps, as the third airplane could not have been launched by the person on the 1st floor, since from their perspective, $c = 2$ airplanes had already been launched.

Also, the array $[1, 1, 2, 3]$ **is not** a valid way to fill in the gaps, as from the perspective of the person on the 3rd floor, only 1 airplane has been launched, while $c = 2$.

,

E. Tropical Season

Input file: standard input
Output file: standard output
Time limit: 4 seconds
Memory limit: 1024 megabytes

You have n barrels of infinite capacity. The i-th barrel initially contains a_i kilograms of water. In this problem, we assume that all barrels weigh the same.

You know that **exactly** one of the barrels has a small amount of tropical poison distributed on its surface, with a total weight of 0.179 kilograms. However, you do not know which barrel contains the poison. Your task is to identify this *poisonous* barrel.

All the barrels are on scales. Unfortunately, the scales do not show the exact weight of each barrel. Instead, for each pair of barrels, they show the result of a comparison between the weights of those barrels. Thus, for any two barrels, you can determine whether their weights are equal, and if not, which barrel is heavier. The poison and water are included in the weight of the barrel.

The scales are always turned on, and the information from them can be used an unlimited number of times.

You also have the ability to pour water. You can pour water from any barrel into any other barrel in any amounts.

However, to pour water, you must physically handle the barrel from which you are pouring, so if that happens to be the poisonous barrel, you will die. This outcome must be avoided.

However, you can pour water into the poisonous barrel without touching it.

In other words, you can choose the numbers i, j, x ($i \neq j, 1 \leq i, j \leq n, 0 < x \leq a_i$, the barrel numbered i is **not** poisonous) and execute $a_i := a_i - x, a_j := a_j + x$. Where x is not necessarily an integer.

Is it possible to guarantee the identification of which barrel contains the poison and remain alive using pouring and the information from the scales? You know that the poison is located on **exactly** one of the barrels.

Additionally, we ask you to process q queries. In each query, either one of the existing barrels is removed, or an additional barrel with a certain amount of water is added. After each query, you need to answer whether it is possible to guarantee the identification of the poisonous barrel, given that there is exactly one.

Input

The first line contains two integers n and q ($1 \leq n, q \leq 2 \cdot 10^5$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 10^6$) — the volumes of water in the existing barrels.

The next q lines contain queries, one per line. Each query is either of the form `+ x` or `- x`, which means, respectively, adding and removing a barrel with x kilograms of water. It is guaranteed that when the query `- x` is made, one of the existing barrels has a volume of x, and that there is always at least one barrel remaining throughout the queries. In all queries, $1 \leq x \leq 10^6$.

Output

Output $q + 1$ lines, the answer to the problem before all queries, and after each query. If it is possible to identify the *poisonous* barrel, output "`Yes`"; otherwise, output "`No`". 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
4 7 2 2 4 11 - 2 + 4 + 30 + 40 - 4 + 2 + 2	Yes No Yes No Yes No No Yes
6 7 5000 1000 400 400 100 99 + 1 - 5000 - 1 - 400 - 400 - 100 - 99	No Yes Yes Yes No No No Yes

Note

In the first test, before any queries, the weights of the barrels are $[2, 2, 4, 11]$. You can compare the values of the first and second barrels. If the result is not equal, you can definitively conclude that the barrel with the greater weight is poisonous. If the result is equal, both barrels are non-poisonous. Then you can pour all the contents of the first barrel into the second. After that, both the second and third barrels will contain 4 kilograms of water. Now let's compare the values of the second and third barrels. If the result is not equal, the barrel with the greater weight is poisonous. Otherwise, both barrels are non-poisonous. The only barrel that could be poisonous is barrel 4. Thus, with this strategy, we can safely determine the poisonous barrel.

,

F. Curse

Input file: standard input
Output file: standard output
Time limit: 3 seconds
Memory limit: 1024 megabytes

You are given two arrays of integers: $a_1, a_2, \ldots, a_n$ and $b_1, b_2, \ldots, b_m$.

You need to determine if it is possible to transform array a into array b using the following operation several (possibly, zero) times.

- Among all non-empty subarrays* of a, choose any with the maximum sum, and replace this subarray with an arbitrary non-empty integer array.

If it is possible, you need to construct any possible sequence of operations. Constraint: in your answer, the sum of the lengths of the arrays used as replacements must not exceed $n + m$ across all operations. The numbers must not exceed 10^9 in absolute value.

*An array a is a subarray of an array b if a can be obtained from b by the deletion of several (possibly, zero or all) elements from the beginning and several (possibly, zero or all) elements from the end.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 200$). The description of the test cases follows.

The first line of each test case contains two integers n, m ($1 \le n, m \le 500$) — the lengths of arrays a and b.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($-10^6 \le a_i \le 10^6$) — the elements of array a.

The third line of each test case contains m integers $b_1, b_2, \ldots, b_m$ ($-10^6 \le b_i \le 10^6$) — the elements of array b.

It is guaranteed that the sum of the values of n across all test cases does not exceed 500.

It is guaranteed that the sum of the values of m across all test cases does not exceed 500.

Output

For each test case, output -1 if it is impossible to transform array a into array b.

Otherwise, in the first line, output the number of operations $0 \le q \le n + m$. Then output the operations in the following format in the order they are performed.

In the first line of each operation, print three numbers l, r, k ($1 \le l \le r \le |a|$). In the second line, print k integers $c_1 \ldots c_k$, which means replacing the segment $a_l, \ldots, a_r$ with the array $c_1, \ldots, c_k$.

The sum of k across all q operations must not exceed $n + m$. Additionally, it must hold that $-10^9 \le c_i \le 10^9$.

You do not need to minimize the number of operations.

Standard Input	Standard Output

3 4 3 2 -3 2 0 -3 -7 0 2 1 -2 -2 2 5 4 -5 9 -3 5 -9 -6 6 -1 -9	4 3 4 1 -3 1 1 1 -3 2 2 1 -7 3 3 1 0 -1 3 2 4 1 -5 1 1 1 -6 2 2 2 6 -1

Note

In the first test, the initial array is modified as follows:

$$[2, -3, 2, 0] \to [2, -3, -3] \to [-3, -3, -3] \to [-3, -7, -3] \to [-3, -7, 0]$$

You may choose to output empty lines or not. Empty lines in the example are added for convenience.