

Another Boring Problem (boring)


Luca has an array a of length N . Luca need to process Q queries. Each query is one of two types:

- 1 $x \rightarrow$ what is the value of a_x modulo $10^9 + 7$?
- 2 $x, y, b, c \rightarrow$ for every i in the range $[x, y]$, a_i becomes $\max(a_i, b \cdot i^c)$.



Figure 1: She's bored as well.

Help Luca find the answer to each type 1 query!

 Among the attachments of this task you may find a template file `boring.*` with a sample incomplete implementation.


Input

The first line contains the only integer N . The second line contains N integers a_i .

The third line contains the only integer Q . The following Q lines can be of the kind 1 x or 2 $x y b c$, depending on which kind of query they represent.

Output

For every 1 x query in the input, you should write a line with an integer: the value of a_x modulo $10^9 + 7$.







 The *modulo* operation ($a \bmod m$) can be written in C/C++/Python as `(a % m)` and in Pascal as `(a mod m)`. To avoid the *integer overflow* error, remember to reduce all partial results through the modulus, and not just the final result!
 Notice that if $x < 10^9 + 7$, then $2x$ fits into a C/C++ `int` and Pascal `longint`.

Constraints

- $1 \leq N \leq 100\,000$.
- $1 \leq a_i \leq 100\,000$ for each $i = 1 \dots N$.
- $1 \leq Q \leq 200\,000$.
- $1 \leq x \leq N$ for each query of the first kind.
- $1 \leq x, y \leq N$ and $1 \leq b, c \leq 100\,000$ for each query of the second kind.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

- **Subtask 1** (0 points) Examples.

- **Subtask 2** (20 points) Every number, except for Q , is at most 10.

- **Subtask 3** (25 points) $c = 1$.

- **Subtask 4** (15 points) $N \leq 1000$ and $Q \leq 100\,000$.

- **Subtask 5** (20 points) $N \leq 50\,000$ and $Q \leq 100\,000$.

- **Subtask 6** (20 points) No additional limitations.


Examples

input	output
10 5 3 7 8 3 9 10 10 1 2 15 1 7 2 1 5 3 3 1 5 1 4 1 1 2 6 10 2 2 1 6 1 7 1 10 2 1 10 10 10 1 1 1 2 1 10 1 7 1 3	10 375 192 5 72 98 200 10 10240 999999307 824752476 590490

Explanation

In the **first sample case** the array contains 10 elements, which initially are:

5, 3, 7, 8, 3, 9, 10, 10, 1, 2

The first query asks the value of the 7th element (which is the first 10).

The next query updates the values of the elements in positions 1 to 5. After the update the array contains:

5, 24, 81, 192, 375, 9, 10, 10, 1, 2

After the second query of type 2, the array contains:

5, 24, 81, 192, 375, 72, 98, 128, 162, 200

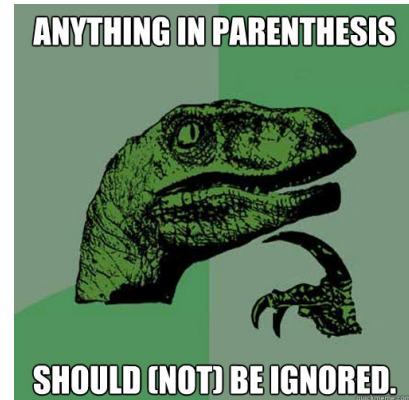
After the last update of type 2, the array contains:

10, 10240, 590490, 10485760, 97656250, 604661760, 2824752490, 10737418240, 34867844010, 100000000000

Quantum Brackets (brackets)

Dario is experimenting with a new kind of brackets: the *quantum* brackets. A *quantum* bracket is in a *superposition* of (meaning, being at the same time) open and closed, and will collapse to either option when observed. Dario is working with K kinds of *quantum* brackets and he managed to get a sequence of N of those. He is wondering whether there exists a way in which they can *collapse* (meaning, become definitely open or closed) to form a well-parenthesized expression.

An expression of K kinds of brackets is well-parenthesized if it is possible to one-to-one match each bracket with one of the same kind such that the first is open, the second is closed and taken any two intervals between matched brackets they are either disjoint or contained in one another.



📎 Among the attachments of this task you may find a template file `brackets.*` with a sample incomplete implementation.

Input

The first line contains an integer T , the number of test cases in the input file. Then $2T$ lines follow, describing each test case, one after the other. The j -th test case is described by two lines: the first one contains two integers N_j and K_j , the number of brackets and the number of kinds of brackets, respectively; the second line contains N_j integers P_{ji} , the kinds of the N_j brackets.

Output






You need to write T lines, one for each test case. If there is a well-parenthesized collapse for the *quantum* brackets print 1, otherwise 0.

Constraints

- $1 \leq T \leq 10$.
- $\sum_{j=1}^T N_j \leq 400\,000$.
- $1 \leq K_j \leq N_j$ for each $j = 1 \dots T$.
- $0 \leq P_{ji} < K_j$ for each $i = 1 \dots N_j, j = 1 \dots T$.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

- **Subtask 1** (0 points) Examples.

- **Subtask 2** (20 points) $\sum_{j=1}^T N_j \leq 20$.

- **Subtask 3** (15 points) $K_j = 1$ for $j = 1 \dots T$.

- **Subtask 4** (30 points) $\sum_{j=1}^T N_j \leq 3000$.

- **Subtask 5** (35 points) No additional limitations.


Examples

input	output
3 8 2 0 1 0 0 1 1 1 0 6 3 2 1 0 2 0 1 16 4 0 1 0 2 3 3 2 1 1 0 0 1 1 0 1 0	1 0 1

Explanation

In the **first example** we have 3 cases.

In the *first case*, if we associate 0 with round brackets and 1 with square brackets, the quantum bracket can collapse to the following expression: $([()] [])$. This way the first bracket is matched with the eighth one, the second with the fifth one, the third with the fourth one, and lastly the sixth with the seventh. All intervals bounded by pairs of corresponding brackets are disjoint or contained in one another. Therefore the solution is 1.

In the *second case* the quantum bracket can never collapse to a well-parenthesized expression. Indeed, since there are only two brackets of each kind, they necessarily need to be matched with one another. Now if we associate 0 with round brackets, 1 with square brackets, and 2 with curly brackets, and impose that for each pair of brackets, the first one should be open while the second should be closed, our quantum bracket collapses to: $([\{ \}])$. From this expression we can easily notice that the second condition is not satisfied: the interval bounded by the round brackets overlaps the interval bounded by the square brackets, and this happens taking into account curly brackets and any other pair of brackets as well. Thus the solution is 0.

In the *third case*, if we associate 0 with round brackets, 1 with square brackets, 2 with curly brackets, and 3 with angle brackets, the quantum bracket can collapse to: $([(\{ < > \} [[()]])])$. Every open bracket has a corresponding closed bracket of the same type and all intervals bounded by pairs of corresponding brackets are disjoint or contained in one another. Therefore the solution is 1.

Sand Buckets (buckets)

Marco likes very much to play with the sand. He has many tools he uses to build sand castles, like shovels, rakes and molds. He also has N buckets, the i -th of which having a diameter of D_i centimeters.



Figure 1: The buckets Marco uses to play with the sand.

Marco would like to pile the buckets one inside the other to transport them more easily. Of course, one bucket A can be put inside another bucket B only if the diameter of A is **strictly smaller** than that of B , that is, if $D_A < D_B$. Help Marco find out if it is possible to put all the buckets in a single pile!

Among the attachments of this task you may find a template file `buckets.*` with a sample incomplete implementation.

Input

The first line contains the only integer N , the number of buckets. The second line contains N integers D_i , the diameter of every bucket.

Output






If all the buckets can be put in a single pile, you need to write a single line containing the string `Ok`. Otherwise, you should print the string `Impossible`.

Constraints

- $1 \leq N \leq 100\,000$.
- $1 \leq D_i \leq 2 \cdot 10^9$ for each $i = 0 \dots N - 1$.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

- **Subtask 1** (0 points) Examples.

- **Subtask 2** (15 points) $N \leq 3, D_i \leq 100\,000$ for each $i = 0 \dots N - 1$.

- **Subtask 3** (30 points) $N \leq 10\,000, D_i \leq 100\,000$ for each $i = 0 \dots N - 1$.

- **Subtask 4** (35 points) $D_i \leq 100\,000$ for each $i = 0 \dots N - 1$.

- **Subtask 5** (20 points) No additional limitations.


Examples

input	output
3 10 2 5	Ok
3 5 10 5	Impossible

Explanation

In the **first sample case**, the second bucket (with diameter of 2 centimeters) can be placed inside the third bucket (with diameter of 5 centimeters). These two buckets can then be put inside the first bucket (with diameter of 10 centimeters).

In the **second sample case**, it is not possible to place all the buckets in the same pile.

Disk Failure (disks)

Giorgio is busy setting up the servers that will host the first OIS round. However, he notices that they are very slow, so he starts investigating to find out what is wrong. It turns out some of the disks are broken and need to be replaced!



Giorgio knows that the servers are used everyday from hour A to hour B . He also knows that he will need T contiguous hours to replace the broken disks.

If Giorgio chooses when to begin the maintenance in an optimal way, what is the minimum number of hours of downtime?

📎 Among the attachments of this task you may find a template file `disks.*` with a sample incomplete implementation.

Input

The first line contains three integers A , B and T : the hour at which the servers start to be used, the hour at which the servers stop being used and the hours needed by Giorgio to replace the disk.

Output





You need to write a single line with an integer: the minimum number of hours of downtime of the servers.

Constraints

- $0 \leq A < B \leq 24$.
- $1 \leq T \leq 10^{12}$.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

- **Subtask 1** (0 points) Examples.

- **Subtask 2** (50 points) $T \leq 1000$.

- **Subtask 3** (30 points) $T \leq 10^9$.

- **Subtask 4** (20 points) No additional limitations.


Examples

input	output
7 21 42	22
0 24 69	69

Explanation

In the **first sample case**, the servers are used everyday from 7:00 to 21:00 and Giorgio needs 42 hours to replace the disks. If the maintenance starts at 19:00, it will end at 13:00 two days later. In total, the servers will be down for 22 hours (2 hours the first day, 14 hours the second day, 6 hours the third day).

In the **second sample case**, the servers are used all day long, from 0:00 to 24:00. In this case, it is not important when Giorgio will start fixing the problem, as there will be 69 hours of downtime anyway.

License Key Generator (keygen)

Edoardo just found a beautiful 4D rendering software. Unfortunately, this software is not free, and a valid license key must be entered to use it. Edoardo, however, reverse engineered the program and found the function that checks the key!

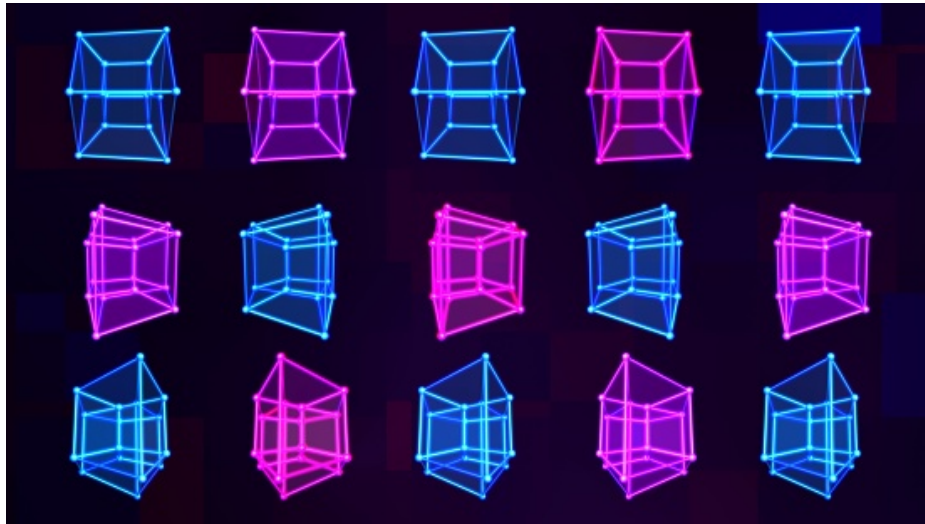



Figure 1: Edoardo illegally using the software.

The function takes K boolean variables, named as the first K lowercase letters of the English alphabet (i.e. “a”, “b”, etc...). These variables are combined in a boolean formula represented by a string S .

The formula is a **XOR** of multiple clauses, where each clause is the **AND** of some literals: a variable (positive literal) or its negation (negative literal). In order to generate a valid license, Edoardo needs to find a way to assign each variable 0 or 1 such that the given formula evaluates to 1. Edoardo wants to generate as many valid licenses as possible to sell them on the dark web. Help Edoardo find the number of different licenses he can sell!

 Among the attachments of this task you may find a template file `keygen.*` with a sample incomplete implementation.

Input

The first line contains the only integer T . The description of T testcases follow. For each testcase, the first line contains the only integer K , and the second line contains the string S .

Output






You need to write T lines, where each line contains the number of different valid licenses (i.e., assignments to variables that satisfy the given formula).

Constraints

- $1 \leq T \leq 20$.
- $1 \leq K \leq 15$.
- $1 \leq |S| \leq 250\,000$, where $|S|$ is the length of S .
- S is a concatenation of clauses.
- Clauses are separated by a XOR: “^”.
- Each clause is bounded by round brackets: “(” and “)”.
- Each clause is a concatenation of literals separated by AND: “&”.
- Each clause contains at least 1 literal.
- A literal is just a variable or a variable preceded by a NOT: “!”.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

- **Subtask 1** (0 points) Examples.

- **Subtask 2** (20 points) There are no negative literals and $1 \leq K \leq 10$.

- **Subtask 3** (30 points) There are no negative literals.

- **Subtask 4** (20 points) $1 \leq K \leq 12$.

- **Subtask 5** (30 points) No additional limitations.


Examples

input	output
3 4 (c&d)^(!a&b&!c)^(b) 4 (!c) 4 (!c&!d)^(!a&c&!d)^(a&!b&c)	6 8 8
3 5 (d)^(!b&e) 5 (b&d)^(b&!e)^(c&!d&e) 5 (b&!c&d&e)^(!a&d)^(!b&c)^(c&d&!e)	16 12 12

Explanation

In the **first testcase** of **first sample case** there are 6 possible valid combinations:

- a=0, b=1, c=1, d=0.
- a=1, b=1, c=1, d=0.
- a=1, b=1, c=0, d=1.
- a=0, b=0, c=1, d=1.
- a=1, b=0, c=1, d=1.

In the **second testcase** of **first sample case** there are 8 possible valid combinations:

- a=0, b=0, c=0, d=0.
- a=1, b=0, c=0, d=0.
- a=0, b=1, c=0, d=0.
- a=1, b=1, c=0, d=0.
- a=0, b=0, c=0, d=1.
- a=1, b=0, c=0, d=1.
- a=0, b=1, c=0, d=1.
- a=1, b=1, c=0, d=1.

In the **third testcase** of **first sample case** there are 8 possible valid combinations:

- a=0, b=0, c=0, d=0.
- a=1, b=0, c=0, d=0.
- a=0, b=1, c=0, d=0.
- a=1, b=1, c=0, d=0.
- a=0, b=0, c=1, d=0.
- a=1, b=0, c=1, d=0.
- a=0, b=1, c=1, d=0.
- a=1, b=0, c=1, d=1.

Martian War (martianwar)


Recent studies have shown that there is indeed intelligent life on Mars. The problem is that now humanity is at war with the Martians! Our best bet for survival is to strike first.



On Mars there is a complex railroad system of N cities connected by M bidirectional railroads. Earth has called upon the best bomber there is, mister RANDy, to strike down those railroads. Because he is a maniac, he only has one bomb left, so he can only strike down a single one of those railroads.

RANDy will only target strategic railroads. A railroad is strategic if and only if there exists a pair of cities (x, y) such that you can reach x from y before the bombing, and after bombing the given railroad you can no longer reach x from y .

The Martians are starting to pick up on our plan, so they are now constructing Q additional railroads. After the construction of each new railroad, RANDy wants to know how many strategic railroads there are. RANDy now asks you to help him find the answer he seeks.

 Among the attachments of this task you may find a template file `martianwar.*` with a sample incomplete implementation.

Input

The first line contains three integers N , M and Q : the number of cities on the Martian surface, the number of railroads connecting those cities, and the number of additional railroads that the Martians will be constructing.

The next M lines each contain two integers x and y , meaning that there is a bidirectional railroad between cities x and y .

The next Q lines each contain two integers x and y , meaning that the Martians are constructing a railroad from city x to city y .

Output





The output will contain Q lines, where the i -th line will contain a single number s , representing the number of strategic railroads after the Martians have constructed the first i new railroads.

Constraints

- $2 \leq N \leq 250\,000$.
- $0 \leq M \leq 250\,000$.
- $1 \leq Q \leq 250\,000$.
- $1 \leq x, y \leq N$.
- The graph is not guaranteed to be connected.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

- **Subtask 1** (0 points) Examples.

- **Subtask 2** (30 points) $N, M, Q \leq 1000$.

- **Subtask 3** (40 points) $N, M, Q \leq 100\,000$.

- **Subtask 4** (30 points) No additional limitations.


Examples

input	output
5 4 2 5 1 3 2 2 5 4 2 1 2 5 3	2 1
6 5 2 5 1 3 2 2 5 4 2 1 2 4 3 6 5	0 1

Explanation

In the **first sample case** there are 5 nodes and initially 4 railroads. After adding the link 1–2, the only strategic links are 2–3 and 2–4. After adding the link 3–5, only the 2–4 link is strategic.

In the **second sample case** there are 6 nodes, and node 6 is not connected. After adding the 3–4 link there are no strategic links. After adding the link 5–6, there is only one strategic link, the 5–6 link.

Police Investigation 3 (police3)

After his traditional Friday's bank robbery, fearsome William is running away trying to reach his nest. As the police is already wary on Terror Street and Crime Avenue, he decides to go through Heist Road instead, which unfortunately is full of traffic lights.



Figure 1: Traffic lights on Heist Road.

Being so unlucky, he knows that he will have to stop to all the N traffic lights in his way, spending T_i minutes at each of them. The police is on his way, and he really needs to hurry up, so he decides to jump a few red lights. Jumping a red light is both dangerous, and it can attract unwanted attention to him, therefore he decides to never skip two consecutive traffic lights. This means that when he reaches a traffic light, he can go through it without stopping, but then he will have to stop to the next one.

What is the minimum amount of time he has to spend waiting at the traffic lights?

Among the attachments of this task you may find a template file `police3.*` with a sample incomplete implementation.

Input

The first line contains the only integer N . The second line contains N integers T_i .

Output






You need to write a single line with an integer: the minimum time William has to wait.

Constraints

- $1 \leq N \leq 100\,000$.
- $1 \leq T_i \leq 10\,000$ for each $i = 0 \dots N - 1$.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

- **Subtask 1** (0 points) Examples.

- **Subtask 2** (15 points) $T = 1$.

- **Subtask 3** (20 points) $N \leq 10$.

- **Subtask 4** (30 points) $N \leq 1000$.

- **Subtask 5** (35 points) No additional limitations.


Examples

input	output
6 3 2 2 6 2 3	6
7 1 6 1 1 4 2 4	5

Explanation

In the **first sample case** the best solution is to stop only at the three traffic lights with value 2.

In the **second sample case** the best solution is to skip the second, fifth and seventh traffic light.

City Redevelopment (renovations)

Alessandro has finally been elected as the new mayor of Pordenone! With great power comes great responsibilities, and so he now has to keep his promises and redevelop part of the city to create a new clubbing centre for Pordenone's youngsters.



Figure 1: Artist's impression of the hypothetical Pordenone's night life. 2021, colourized.

The city of Pordenone is composed of N buildings, numbered from 1 to N all aligned along a single road, and each building i has a *beauty value* V_i . Pordenone's youngsters will only accept to use a clubbing centre encompassing a contiguous sequence of buildings from l to r ($1 \leq l \leq r \leq N$), consisting only of buildings with a beauty value of at least K , and having a *total beauty* (sum of the values of each building) of **exactly** s .¹ Since Alessandro is particularly indecisive, before proceeding with the necessary renovations he wants to calculate in **how many** different ways the buildings in the $[l; r]$ interval could be renovated (thus, increasing their beauty value V_i) in order to achieve a total beauty of exactly s . For instance, assume that Pordenone is composed of $N = 6$ buildings, with the following beauty values:

13 0 30 5 15 25

and assume that Pordenone's youngsters want a clubbing centre in the interval between $l = 2$ and $r = 4$, with a minimal beauty of $K = 20$ and a total beauty of $s = 73$. There are 10 possible ways for Alessandro to meet their conditions, which would produce the following beauty values:

13	20	30	23	15	25	13	20	31	22	15	25	13	20	32	21	15	25
13	20	33	20	15	25	13	21	30	22	15	25	13	21	31	21	15	25
13	21	32	20	15	25	13	22	30	21	15	25	13	22	31	20	15	25
						13	23	30	20	15	25						

¹Youngsters from Pordenone are *very* picky: they don't like both too ugly nor too classy neighbourhoods.

However, Alessandro indecisiveness is going to cost him the success of his quest! While he is pondering on how to perform the renovations, the beauty values of buildings and the youngsters' preferences l, r, s keep changing. In particular, you are given Q different queries to handle, each of them of one of two possible kinds:

- 1 $a\ b \rightarrow$ building a has changed its beauty value to $V_a = b$;
- 2 $l\ r\ s \rightarrow$ youngsters now want a clubbing centre in the interval $[l; r]$ with a total beauty of s .

Help Alessandro by keeping track of the number of different ways the buildings could be renovated to meet the youngsters' requirements, every time they change their mind!


 Among the attachments of this task you may find a template file `renovations.*` with a sample incomplete implementation.

Input

The first line contains the three integers N, Q, K . The second line contains N integers V_i . The following Q lines can be of the kind 1 $a\ b$ or 2 $l\ r\ s$, depending on which kind of query they represent.

Output

For every 2 $l\ r\ s$ query in the input, you should write a line with an integer: the number of different ways buildings could be renovated to meet the youngsters' criteria **modulo** $10^9 + 7$.




 The *modulo* operation ($a \bmod m$) can be written in C/C++/Python as `(a % m)` and in Pascal as `(a mod m)`. To avoid the *integer overflow* error, remember to reduce all partial results through the modulus, and not just the final result!
Notice that if $x < 10^9 + 7$, then $2x$ fits into a C/C++ `int` and Pascal `longint`.




Constraints

- $1 \leq N \leq 100\,000$.
- $1 \leq Q \leq 200\,000$.
- $0 \leq K \leq 1000$.
- $0 \leq V_i \leq 100$ for each $i = 1 \dots N$.
- $1 \leq a \leq N$ and $0 \leq b \leq 100$ for each query of the first kind.
- $1 \leq l \leq r \leq N$ and $0 \leq s \leq 2\,000\,000$ for each query of the second kind.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

- **Subtask 1** (0 points) Examples.

- **Subtask 2** (15 points) $N, Q \leq 10$ and $s \leq 10$ in every query.

- **Subtask 3** (25 points) $N \leq 2000, Q \leq 5000, K = 0$.


- **Subtask 4** (20 points) $N \leq 2000, Q \leq 5000$.

- **Subtask 5** (20 points) $K \leq 20$.

- **Subtask 6** (20 points) No additional limitations.


Examples

input	output
6 1 20 13 0 30 5 15 25 2 2 4 73	10
4 5 2 0 0 0 3 2 1 4 10 1 1 2 2 2 3 3 1 2 4 2 1 2 6	4 0 1

Explanation

The **first sample case** is described in the problem statement.

In the **second sample case**, Pordenone is composed of four buildings with the following initial beauty values:

0 0 0 3

And the minimum beauty accepted by youngsters is 2. Firstly, Alessandro needs to build a clubbing centre encompassing the whole city, and this can be done in 4 ways:

2 2 3 3 2 3 2 3 3 2 2 3 2 2 2 4

While Alessandro is pondering, the beauty of building 1 changes:

2 0 0 3

The youngsters' preferences also change, and now Alessandro needs to build the centre on the two middle buildings with a total beauty of 3. However, this is not possible: both buildings would need to have a minimum beauty of 2, their total beauty is going to be at least 4. Then, building 2 also changes:

2 4 0 3

Now Alessandro needs to build the centre on the leftmost two buildings with a total beauty of 6. This is possible in a single way: by just not doing anything.

UEFA Champions League (ucl)

Stefan is a big football fan! He just read about some past competitions where teams are divided into groups, and each team plays against every other team in its group twice. A team is awarded three points for a win, one point for a draw and zero points for a loss.

Given the final standings of T groups, help Stefan find out whether there is a unique way of assigning the results for each match such that we end up having the final standings given. However, since the data can be faulty, there can also be a chance that the standings of some groups are not valid, so you will also have to help Stefan find them out.



Figure 1: Stefan being a huge football fan.

In particular, the teams in the standings of each group are given in **decreasing order of points**, and if two or more teams have the same number of points, they are ordered in **decreasing order of wins**.

Given the number of wins, draws and losses of each team, you have to print one of the following strings:

- “**Unique**” if there is **only one way** of assigning the result (win, draw or loss) to each match (also distinguishing the two matches between a same pair of teams);
- “**Not unique**” if instead there is **more than one way** of assigning the results to get the given standings;
- “**Invalid**” if the group results are invalid, that is, if it is **impossible to assign results** to each match so that they are coherent with the given standings, or if the teams are **not in the correct order**.

Note that Stefan is not interested in the exact result of each match: he only wants to know which team won or if the match ended in a draw.

Among the attachments of this task you may find a template file `ucl.*` with a sample incomplete implementation.

Input

The first line contains the only integer T , the number of groups. Then the T groups are described, one after the other. For each group description, the first line contains one integer N , the number of teams in the group. Then N lines follow, each containing a 3-digit number: the first digit is the number of wins, the second digit is the number of draws and the last digit is the number of losses of a single team.

Output







You need to write T lines each containing either the string “Unique”, “Not unique” or “Invalid”.

Constraints

- $1 \leq T \leq 100\,000$.
- $2 \leq N \leq 4$ for each group.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

- **Subtask 1** (0 points) Examples.

- **Subtask 2** (10 points) $N = 2$ for each group.

- **Subtask 3** (20 points) $2 \leq N \leq 3$ for each group.

- **Subtask 4** (15 points) $T \leq 10$.

- **Subtask 5** (25 points) All the standings are guaranteed to be valid.

- **Subtask 6** (30 points) No additional limitations.


Examples

input	output
6	Unique
4	Not unique
600	Not unique
402	Invalid
204	Unique
006	Invalid
4	
402	
402	
402	
006	
2	
110	
011	
2	
011	
110	
3	
400	
022	
022	
4	
330	
240	
042	
114	

Explanation

The sample case consists of five groups.

In the **first group**, there is only one way to assign the outcomes of each match. In particular, the first team has won all the matches and the last one has lost all of them. The second team has won all the matches, except the ones against the first team. The third team has lost all the matches, except the ones against the last team.

In the **second group**, the last team has lost all the matches he played. However, Stefan cannot determine the results of the matches played among the first three teams.

In the **third group**, the first team has won against the second in one match, and draw in the other match. Since there is no way to determine which of the two matches is a win and which is a draw, the configuration should be marked as “Not unique”.

In the **fourth group**, the results are the same as in the third group, but the standings are in the wrong order. Therefore, the configuration is invalid.

In the **fifth group**, the first team has won against the other two. They, instead, tied with each other in both their matches. The result of each match is then uniquely determined, therefore, the configuration should be marked as “Unique”.

In the **sixth group**, although there are ways to obtain the given standings, they are not presented in the correct order: in particular, the last two teams have the same number of points, but are not ordered by decreasing number of wins.