Problem A. Matrix Minors

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

You're given an $n \times n$ matrix A. The minor M_{ij} of A in (i, j) is the determinant of the matrix obtained from A by deleting the i-th row and the j-th column. Your task is to find all minors of A.

Input

First line of input contains a single integer n ($2 \le n \le 500$).

The i-th of the following n lines contains n numbers $A_{i1}, \ldots, A_{in}$, the i-th row of A ($0 \le A_{ij} < 10^9 + 7$).

Output

Print n lines, containing n numbers each.

The i-th line should contain $M_{i1}, \ldots, M_{in}$, the minors of A.

Since the resulting numbers can be very big, print them modulo $10^9 + 7$.

Examples

standard input	standard output

2
0 1
1 1
	1 1
	1 0

3
0 0 0
0 0 0
0 0 0
	0 0 0
	0 0 0
	0 0 0

3
1 2 3
4 5 6
7 8 9
	1000000004 1000000001 1000000004
	1000000001 999999995 1000000001
	1000000004 1000000001 1000000004

Note

In the third example, the matrix M is given as follows:

$$M = -\begin{pmatrix} 3 & 6 & 3 \\ 6 & 12 & 6 \\ 3 & 6 & 3 \end{pmatrix}$$

Problem B. Digital Products

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1.5 seconds
Memory limit:	256 megabytes

Define $d(x)$ as the product of digits of x. For example, $d(5442) = 5 \cdot 4 \cdot 4 \cdot 2 = 160$.

For a given integer n, define the set $D = \{d(x) | 1 \le x \le n\}$. Find $|D|$.

In other words, find the number of different products of digits for all integers between 1 and n.

Input

The first line contains one integer t $(1 \le t \le 1\,000)$ — the number of test cases.

Each test case consists of one line containing the integer n $(1 \le n \le 10^{18})$.

Output

For each test case, print the answer on a separate line.

Example

standard input	standard output
6 1 5 10 20 100 1000	1 5 10 10 37 101

Problem C. Concert Lineup

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

You are a manager for a popular concert tour, and you are supposed to plan the lineup for each concert. The marketing department uses polls to give you updates on the optimal lineup.

The initial lineup consists of n different artists numbered 1 through n in a certain order. After each poll, the marketing department will give you an update of the following form:

You will receive an **even** number k. This means you should remove the first $\frac{k}{2}$ artists with **odd** positions (1-indexed) from the lineup, and you should reverse the order of the first $\frac{k}{2}$ artists with **even** positions. The two operations are done simultaneously.

Output the resulting lineup after processing every update from marketing.

Input

In the first line of input, you receive t ($1 \le t \le 100$), the number of test cases. Then t test cases follow.

The first line of each test case contains two integers n and q ($2 \le n \le 10^5$; $1 \le q < n$), the initial number of artists and the number of updates.

The second line contains n distinct integers $a_1, \ldots, a_n$ ($1 \le a_i \le n$), the initial order of artists.

The third line contains q even integers $k_1, \ldots, k_q$ ($2 \le k_i \le n$), the updates from the marketing department.

It is guaranteed that k_i is not greater than the current number of artists for every update. Additionally, it is guaranteed that the sum of n over all test cases is not greater than 10^5.

Output

Print the final lineup after processing all updates.

Example

standard input	standard output

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

Note

In the first update of the first test case, the elements 1 and 3 will get removed, as they are in odd positions; then 2 and 4 will be reversed.

In the second update of the first test case, the elements 4, 5, and 7 will get removed, and 2, 6, 8 reverse their positions.

Problem D. LSB

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

As a lover of all things bitset, you have been invited by the *Loving Society of the Bitset* (or LSB, for short) to be a tester for their new programming language *Language for Straightforward Bitsets* (or LSB, for short). You need to implement a program in LSB that isolates the *least significant bit* (or LSB, for short) of the input bitset.

LSB operates on sequences of bits, so called *bitsets*, of length n. The input of an LSB program is stored in the bitset `B0`. The i-th line of an LSB program computes the bitset Bi, and it has to be one of the following:

- Bi = Bj << k $(0 \le j < i,\ 0 \le k \le n)$ – shift the bits of Bj by k places to the left (empty places get filled with zeros) and store the result to Bi
- Bi = Bj >> k $(0 \le j < i,\ 0 \le k \le n)$ – shift the bits of Bj by k places to the right (empty places get filled with zeros) and store the result to Bi
- Bi = Bj ^ Bk $(0 \le j, k < i)$ – compute the bitwise XOR of Bj and Bk and store the result to Bi
- Bi = Bj | Bk $(0 \le j, k < i)$ – compute the bitwise OR of Bj and Bk and store the result to Bi

The last computed bitset is considered the output of the LSB program. If the program is empty, we consider its output to be `B0`.

For example, consider the following program for $n = 8$:

```
4
B1 = B0 << 2
B2 = B1 >> 3
B3 = B0 ^ B1
B4 = B2 | B3
```

For input `B0 = 10100010`, the example program computes the following bitsets:

```
B1 = 10001000

B2 = 00010001

B3 = 00101010

B4 = 00111011
```

The final output of this program is `00111011`.

Bits of a bitset are indexed from right to left. The LSB of a bitset is the bit with the smallest index that is set to 1 (the right-most 1). The LSB is interested in isolating the LSB. More precisely, the desired output of your program is a bitset that has the LSB of the input bitset set to 1 and all other bits set to 0. For example, for input `10100100`, your program should produce output `00000100`.

Your program needs to **work for all input bitsets** of length n that have at least one bit set to 1.

Can you help the LSB and write a program in LSB that isolates the LSB?

Input

The input is a single integer n $(1 \le n \le 100\,000)$ – the length of the bitsets your LSB program is operating on.

Output

In the first line, output the number of lines of your LSB program. After that, output your LSB program. Your LSB program is allowed to have **at most 100 lines**.

Example

standard input	standard output

`8`	`8`
	`B1 = B0 << 1`
	`B2 = B0 | B1`
	`B3 = B2 << 2`
	`B4 = B2 | B3`
	`B5 = B4 << 4`
	`B6 = B4 | B5`
	`B7 = B6 << 1`
	`B8 = B6 ^ B7`

Problem E. Lighting the Street

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

You've recently been promoted from an apprentice to a Junior City Planner.

Your first task in the new position is to figure out the optimal light bulb type the city should buy to place into the street lights on the side street Felseneggweg. You're not yet senior enough to decide on the light bulb type for the main road.

In this problem, the street is modeled as the segment $[0, L]$ of real numbers. All points on the segment must be illuminated (including its borders and non-integer points). There are n street lights positioned on the street at integer positions p_i, each of which needs a new light bulb.

There are m different light bulb types that illuminate varying distances, and their costs are determined by the distance they cover: the more the distance, the more expensive light bulbs of this type are. When installed in a position x, a light bulb of the j-th type illuminates the distance l_j in each direction along the street. In other words, the whole segment $[x - l_j, x + l_j]$ is illuminated, including its border points.

Since the city would like to buy the light bulbs in bulk, all street lights must use the same light bulb type. You need to find the cheapest light bulb type that allows you to light the entirety of Felseneggweg to show your boss that your promotion was well deserved.

Input

The first line of input contains three integers n, m ($1 \le n, m \le 10^5$), and L ($1 \le L \le 10^9$).

The second line contains n integers $p_1, \ldots, p_n$ ($0 \le p_i \le L$), the positions of the street lights. It is guaranteed that all positions are distinct.

The third line contains m integers $l_1, \ldots, l_m$ ($0 \le l_j \le L$), the illuminating distances of light bulb types. It is guaranteed that all distances are distinct.

Output

Return the optimal illuminating distance l_j, or -1 if it is not possible to cover the entire street.

Examples

standard input	standard output

`4 7 20`
`10 3 7 14`
`2 5 1 7 10 4 20`
	`7`

`3 4 200`
`14 153 22`
`30 2 50 20`
	`-1`

Problem F. Autobahn Optimization

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1.5 seconds
Memory limit:	256 megabytes

You just finished picking the best light bulb for Felseneggweg and are now on your way home on the German Autobahn, which is famous for its absence of a speed limit and super fast cars. Since you are a Junior City Planner, you notice that everyone could arrive at the destination faster and want to stop them from blocking each other. Therefore, you start optimizing which car should go in which lane.

There are n cars driving on the Autobahn, which currently has only one lane. Since not all cars are the same, the i-th car has a maximum speed of s_i.

Now the cars arrive at a fork, at which the Autobahn splits up into two lanes. Each car can pick one of the two lanes and has to stay there since a barrier splits the lanes.

All cars want to go as fast as possible, but they cannot drive faster than the car in front of them. In particular, if, after the lane split, the car j drives right in front of the car i with the speed of c_j, then the i-th car will drive with the speed of $c_i = \min(c_j, s_i)$. If there is no car in front of the i-th car, then $c_i = s_i$.

Your task is to distribute the cars into lanes in such a way that the sum of the speed losses is minimized:

$$\sum_{i=1}^{n}(s_i - c_i) \to \min$$

What is the least possible sum of speed losses?

Input

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

The first line of each test case contains a single integer n ($1 \leq n \leq 1\,000$), the number of cars. The cars enter the lanes in the order they are given in the input, and this order cannot be changed.

The second line of each test case contains n integers $s_1, \ldots, s_n$ ($1 \leq s_i \leq 5\,000$), the maximum speeds of the cars.

It is guaranteed that the sum of n over all test cases does not exceed $1\,000$.

Output

Print one integer, the least possible sum of speed losses.

Examples

standard input	standard output

2
5
4 3 8 2 7
5
3 9 10 5 1
	0
	1

3
6
1 2 3 4 5 6
6
6 1 5 2 4 3
4
1 3 2 4
	6
	1
	2

Problem G. Contrived Intelligence

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

This is an interactive problem.

With the field of artificial intelligence on the rise, you have decided to take a detour and instead look at the human mind again. You have realized that the emotions of your best friend, Adamant, can be encoded as a hidden polynomial with integer coefficients; call it $P(x)$. To know more about $P(x)$, you can ask everyday questions from Adamant while obtaining mathematical values related to $P(x)$ from his answers.

There are N questions that you may ask, numbered 1 through N. Whenever you ask the k-th question, you are told how many of the integers $P(1), P(2), \ldots, P(k)$ are divisible by k.

Of course, it is impossible to fully understand a person simply from talking to them, but you would like to predict how Adamant might behave in certain scenarios.

There are N possible scenarios, numbered 1 through N. In the k-th scenario, Adamant's behavior is fully determined by the number of integers among $P(1), P(2), \ldots, P(k)$, which are co-prime with k.

You have to predict Adamant's behavior in each scenario while asking him at most 10^4 questions.

Interaction Protocol

The interaction begins with the interactor providing you with the number N $(1 \le N \le 10^5)$, which serves as the upper limit on the questions and the number of scenarios. You can then begin asking questions.

To ask the k-th question $(1 \le k \le N)$, print "`? k`", without quotes. Afterward, you should read a single integer, the answer to the k-th question. Don't forget to **flush** the output after printing a question!

You are only allowed to ask **at most 10^4 questions**.

When you have fully determined Adamant's behavior in all possible scenarios, print "`! `", followed by N integers. The k-th integer must be the number of integers j, such that $1 \le j \le k$ and $\gcd(P(j), k) = 1$.

For each test, the polynomial $P(x)$ is chosen in advance and does not change.

Examples

standard input	standard output
3 1 0	 ? 2 ? 3 ! 1 1 3
6 1 1 0	 ? 1 ? 5 ? 6 ! 1 1 3 2 4 3

Note

In the first sample, the polynomial is $P(x) = x^2 + 1$. In the first query, the answer is 1, as among $1^2 + 1$

and $2^2 + 1$, exactly one is divisible by 2.

In the second example, the hidden polynomial $P(x)$ is $3x^3 + 2$.

To flush the output, use:

- `fflush(stdout)` or `cout.flush()` in C++;
- `System.out.flush()` in Java;
- `flush(output)` in Pascal;
- `stdout.flush()` in Python;
- see the documentation for other languages.

Problem H. Zürich Trams

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	1024 megabytes

It is a very rare sight when public transportation systems are as well designed as they are in Zürich. Residents of Zürich often take this for granted, as they reach any destination in time by following a path that is built for them in a transportation app.

In this problem, we are going to make you appreciate it. To make it simpler, we will assume that Zürich only has trams as its public transport, and we will model the connections between the stations of the Zürich tram network as a tree, meaning that the tram network, which consists of n stations numbered 1 through n, is connected and has no loops.

In total, there are m trams; the i-th tram can be represented by three integers s_i, e_i, and k_i: the starting location of the tram, the ending location of the tram, and the pace of the tram, respectively. In the morning, the i-th tram starts at the location s_i and travels to the location e_i; after reaching e_i, it travels back to s_i. This process repeats indefinitely; the tram always takes the shortest path, and it takes the time k_i to travel a direct connection between two stations.

Bjarki lives at the station a and has been working until the early morning. He just realized that his favorite rave has already started. He quickly put his clothes on, opened his travel app, and put in his destination, which is located at the station b. When a tram enters the station that Bjarki is located in, he can enter it; this takes a negligible amount of time. Then he can leave the tram at any station, which also takes a negligible amount of time.

Help Bjarki get to the rave as quickly as possible!

Input

The first line contains four integers n, m, a, and b ($2 \le n \le 1\,000$; $1 \le m \le 1\,000$; $1 \le a, b \le n$): the number of stations in the Zürich tram network, the number of trams, and Bjarki's home station and his favorite rave's station, respectively. It is guaranteed that a and b are distinct.

Each of the next $n - 1$ lines contains two integers u, v ($1 \le u, v \le n$), indicating a direct connection between the stations u and v in the tram network. It is guaranteed that it is possible to reach any station from any other station via a sequence of direct connections.

The i-th of the following m lines contains three integers s_i, e_i, k_i ($1 \le s_i, e_i \le n$; $1 \le k_i \le 10^5$), representing the starting station, ending station, and the pace of the i-th tram.

It is guaranteed that s_i and e_i are distinct. It is possible that a connection is used by more than one tram, and some of the connections might not be used by any tram, meaning that Bjarki can't use them.

Assume that all trams simultaneously start moving from their starting stations the moment Bjarki reaches the station near his home.

Output

Output a single integer, the minimum amount of time it takes for Bjarki to travel to the rave. If there is no possible way for Bjarki to travel to the rave via the tram network, output -1.

Examples

standard input	standard output
3 2 1 3	15
1 2	
2 3	
2 1 5	
2 3 3	
2 1 2 1	10
1 2	
2 1 10	

Problem I. Periodic Recurrence

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

You are given an integer sequence $a_0, a_1, a_2, \ldots,$ in which the first two elements a_0 and a_1 are given, and for $i \geq 2$, the elements are specified by the recurrent formula $a_i = (A \cdot a_{i-1} + B \cdot a_{i-2}) \bmod M$.

The period of a sequence is a positive integer T such that, starting with some i_0, for any $i \geq i_0$, it holds that $a_i = a_{i+T}$. Find the **smallest** period of the given sequence.

Input

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

Each test case contains 5 integers M, a_0, a_1, A, B on a single line ($2 \leq M \leq 10^9$; $0 \leq a_0, a_1, A, B < M$).

Output

For each test case, print a single positive integer, the smallest period of the given sequence.

It can be shown that under the given constraints, the answer always exists and is not greater than 10^{18}.

Example

standard input	standard output

`9`
`3 1 1 1 1`
`3 0 1 2 2`
`7 0 0 3 4`
`10 1 2 3 0`
`10 1 2 0 3`
`11 1 1 6 6`
`12 4 3 2 1`
`15 3 5 5 2`
`19 8 2 18 14`
	`8`
	`3`
	`1`
	`4`
	`8`
	`1`
	`8`
	`24`
	`171`

Note

Sequences in the first 6 test cases (repeating subsequence is bolded):

- **1, 1, 2, 0, 2, 2, 1, 0**, 1, 1, 2, 0, 2, 2, 1, 0, ...
- **0, 1, 2**, 0, 1, 2, 0, 1, 2, ...
- **0**, 0, 0, ...
- 1, **2, 6, 8, 4**, 2, 6, 8, 4, ...
- **1, 2, 3, 6, 9, 8, 7, 4**, 1, 2, 3, 6, ...
- **1**, 1, 1, ...

Problem J. Gibberish

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

This is an interactive problem.

While roaming through the Swiss Alps one weekend, you stumble upon a curious creature. At first, it appears the creature is speaking ~~Swiss German~~ gibberish. However, after some time, you begin to notice a pattern in its speech!

Whenever you say a word consisting of exactly n letters, the creature applies a fixed permutation to the letters and says the permuted word back.

More formally, the creature has a hidden permutation p of size n. When you say a word $w_1w_2...w_n$, the creature constructs a new word by placing letter w_1 at position p_1, letter w_2 at position p_2, ..., letter w_n at position p_n, and says the new word back to you.

Your task is to find the permutation p.

Interaction Protocol

You should begin by reading a single integer n ($1 \le n \le 100\,000$) – the length of the permutation.

After that, you can print queries of the format "`? w`", where w is a string consisting of exactly n lowercase letters. Don't forget to **flush** the output after printing a query!

After each query, you should read a string, the result of applying p to w.

You are allowed to make **at most** 4 **queries**.

When you have figured out the permutation p, print your guess for p as "`! p1 p2 ... pn`" and terminate.

Example

standard input	standard output
`5`	
	`? hello`
`hello`	
	`? world`
`wolrd`	
	`! 1 2 4 3 5`

Note

To flush the output, use:

- `fflush(stdout)` or `cout.flush()` in C++;
- `System.out.flush()` in Java;
- `flush(output)` in Pascal;
- `stdout.flush()` in Python;
- see the documentation for other languages.

Problem K. Cheater Detector

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

There have been reports of cheating in the ICPC (Interdisciplinary Course for Programming and Cooking) at ETH Zürich! After checking all submitted assignments, a list of cheating reports has been made.

Each report is a pair (X, Y), meaning that the student with legi X and the student with legi Y shared some of their homework. For simplicity, we assume that legi are consecutively indexed, starting with 1 (if there are 5 students taking the course, they will have a legi of $1, 2, 3, 4$, and 5).

The general consensus is that most ETH students follow the code of conduct, and the current hypothesis is that the students can be divided into 2 types:

- Cheaters, who cheat whenever they can and with whoever they can.
- Non-cheaters, who generally don't cheat but might be influenced by cheaters to share some solutions.

Specifically, the hypothesis is that the set $V = \{1, 2, \ldots, N\}$ of students can be partitioned into the sets A and B such that:

- For any 2 students x and y in A, a report with the pair (x, y) or (y, x) has been made.
- For any 2 students x and y in B, **no** report with the pair (x, y) or (y, x) has been made.

Your task is to verify if such a division of V into A and B exists.

Input

The first line contains the numbers N $(1 \le N \le 10^5)$ and M $(0 \le M \le 10^6)$, representing the number of students taking the course and the number of reports.

Each of the following M lines contains two numbers x_i and y_i $(1 \le x_i, y_i \le N, x_i \ne y_i)$, the i-th report.

For any two distinct reports, the corresponding pairs of students are distinct as well.

Output

If a valid partition of the students satisfying the hypothesis exists, print `YES`. Otherwise, print `NO`.

Examples

standard input	standard output

`4 4`
`1 2`
`2 3`
`3 4`
`4 1`
	`NO`

`5 6`
`1 2`
`2 3`
`3 1`
`1 4`
`1 5`
`3 5`
	`YES`

Problem L. Drawing Rectangles

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

As an aspiring artist, you prepare to impress the world with another masterpiece on an $n \times m$ canvas.

To make it more impactful, you will close your eyes and paint k non-degenerate rectangles one after another in a way that, for each rectangle, its edges are parallel to the sides of the canvas and have integer distances from them, and the rectangles themselves are chosen uniformly at random from the set of all valid rectangles, independently from each other.

You are certain that the impact of the painting on the audience will be equal to the number of rectangles (out of the k that you have painted) that are fully visible and unobscured by other rectangles.

You want to find the expected value of the impact of your painting over all possible ways to select k rectangles. Assume that if two rectangles only share an edge, they do not obscure each other.

Input

The only line of the input contains three integers n, m, and k ($1 \le n, m, k \le 42$).

Output

Let $\frac{p}{q}$ be the reduced fraction representing the expected impact of your painting.

Print $p \cdot q^{-1} \pmod{10^9+7}$. It can be proven that $q \not\equiv 0 \pmod{10^9+7}$.

Examples

standard input	standard output
`1 2 2`	`222222225`
`3 4 5`	`51141916`

Note

In the first example, the answer is $\frac{11}{9} \approx 1.22$, as there are $3 \cdot 3 = 9$ ways to paint two rectangles, but only two of such ways would allow presenting both rectangles unobscured.