Problem A: Approaching Hurricane

Aroma lives on an island in the middle of the sea. It's hurricane season and Aroma needs to lead the clean up efforts after each hurricane, so she wants to know the area of the affected region in each hurricane beforehand.

To do so, Aroma has modeled the island as a **simple** polygon that has n vertices and n edges. The polygon may not be convex, but it is not self-intersecting.

There are q upcoming hurricanes. Each hurricane has the form of a *moving* circle with changing radius. Here is how Aroma formally models the hurricane:

- The hurricane is modeled as a circle, with its center representing the eye of the hurricane, and the radius of the circle representing the effective range.
- The hurricane will appear with center at point (x_s, y_s) and a radius of r_s.
- The hurricane's center will gradually move from (x_s, y_s) to (x_t, y_t) **in a straight line**, and its radius also changes gradually from r_s to r_t.

  Formally, at some moment, if the distance between the hurricane's current center and (x_s, y_s) is d_s, and the distance between that and (x_t, y_t) is d_t; then the hurricane's current radius is

$$\frac{d_s \cdot r_t + d_t \cdot r_s}{d_s + d_t}$$

- When the hurricane's center reaches the point (x_t, y_t), it stays there until it dissipates.

If at any moment, a point P on the island is within the circle representing the i-th hurricane, then P is said to be **affected** by this hurricane.

After modeling, Aroma wants to determine the **affected** land area for each hurricane so she can devise a helping plan. Please help Aroma find the area of **affected** land for each hurricane passing by.

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 contains a single integer n ($3 \le n \le 3 \cdot 10^5$) – the number of points that make up the island.
- The i-th of the next n lines contains two integers x_i and y_i– ($|x_i|, |y_i| \le 5000$) – the coordinates of the i-th vertex of the polygon that represents the island.
- The next line contains an integer q ($1 \le q \le \min\left\{3 \cdot 10^5, \frac{10^6}{n}\right\}$) – the number of upcoming hurricanes.
- The i-th of the next q lines contains six integers $x_s, y_s, r_s, x_t, y_t, r_t$ ($|x_s|, |y_s|, |x_t|, |y_t| \le 5000$, $1 \le r_s, r_t \le 100$) – representing the i-th hurricane.

It is guaranteed that:

- vertices are listed in either clockwise or counterclockwise order,
- no two given vertices of the polygon have the same coordinates,
- two edges of the polygon only intersect at their shared end vertex (if any),
- the sum of $n \cdot q$ over all t test cases does not exceed 10^6.

Output

For each of the hurricanes, print the affected area of the island. Your answer is considered correct if its absolute or relative error does not exceed 10^{-6}.

Formally, let your answer be a, and the jury's answer be b. Your answer is accepted if and only if $\frac{|a-b|}{\max(1,|b|)} \le 10^{-6}$.

Sample Input 1

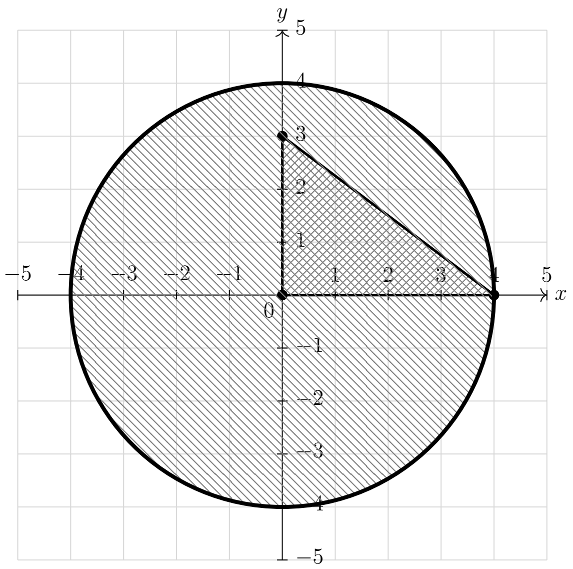
```
3
3
0 0
0 3
4 0
5
0 0 4 0 0 4
0 0 1 0 3 1
0 0 2 4 0 2
1 1 1 5 5 5
-1 -1 1 3 3 3
5
0 0
1 1
2 2
2 0
2 -2
1
2 0 1 6 0 2
3
3 3
2 1
1 1
1
2 2 1 2 2 1
```

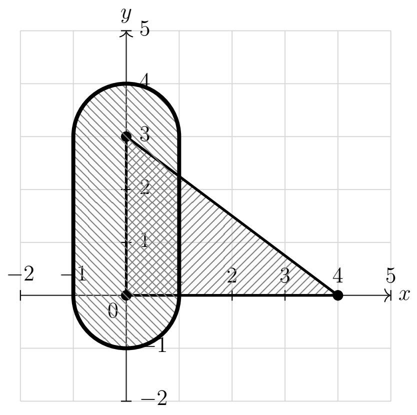
Sample Output 1

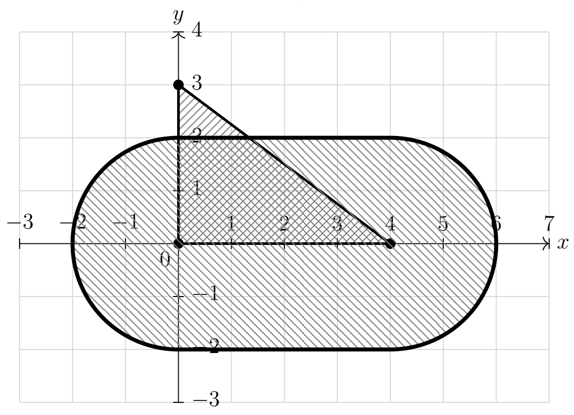
```
6.000000000000
2.625000000000
5.333333333333
5.785398163397
5.982317997582
1.576314961400
0.863647609001
```

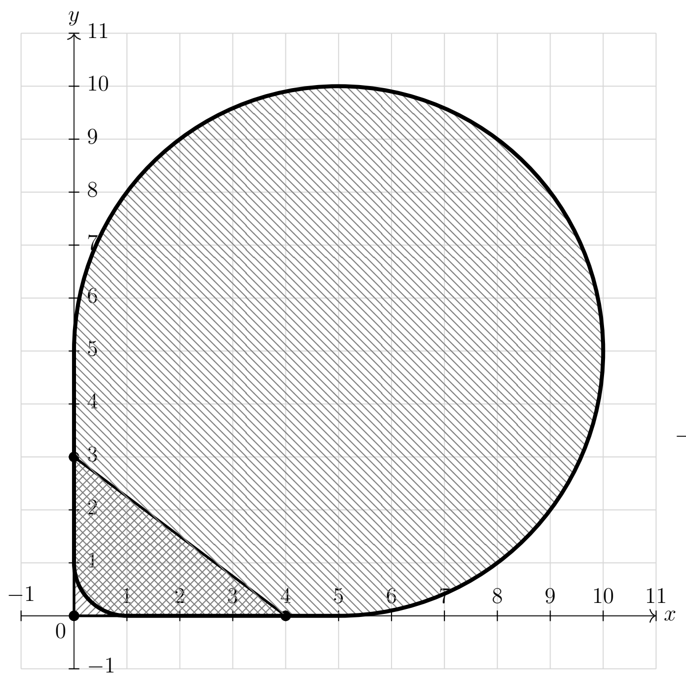
Sample Explanation

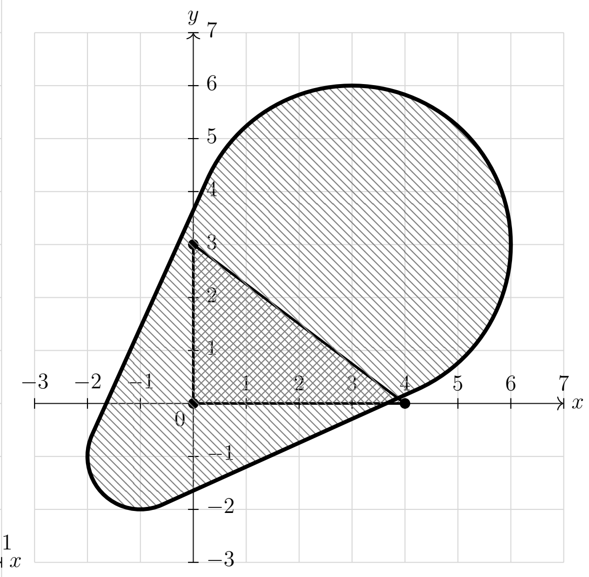
Below are the illustrations for the first test case. The triangle represents the island.


Query 1

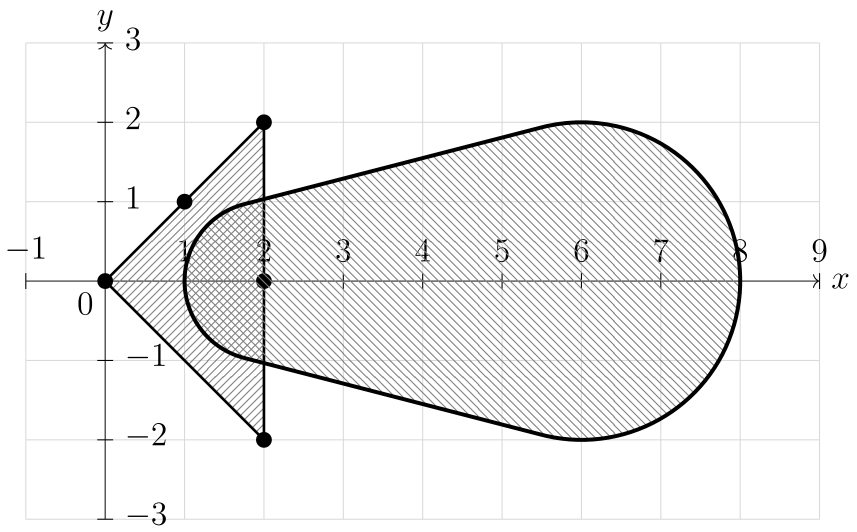

Query 2


Query 3

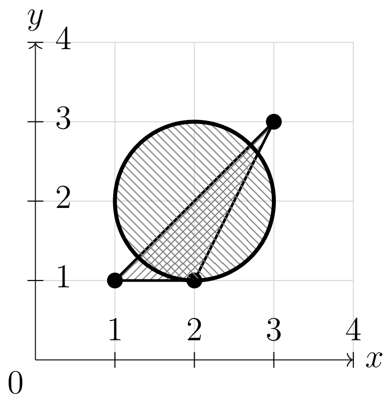

Query 4


Query 5

The following are the illustrations for the second test case.



The following are the illustrations for the third test case.



Problem B: Bullet Train

In the year of 2207, the country PVH has become one of the busiest countries in the world. To satisfy the rapidly increasing demand of of travelling between cities increases rapidly, and to reduce the emission caused by airplanes, the government plans to establish a bullet train system.

There are n cities in the country PVH. These cities are numbered from 1 to n, inclusively. Each city has exactly one central station, where trains depart and arrive. The current rail system in country PVH allows trains to travel directly between every pair of cities. The distance between the central stations of two cities i and j is $d_{i,j}$. Please note that $d_{i,j} = d_{j,i}$ for every pair of cities (i, j) and $d_{i,i} = 0$ for every city i.

The government plans to operate several train routes. Each route departs from some station, passing through several other stations and terminating at some station. In a route, the train must not pass through a station more than once and there should be at most k stations in a route (including the starting and ending ones). The operating cost of a route is the total distance of all pairs of consecutive stations.

Formally, a route can be represented as a sequence of integers $x_1, x_2, \ldots, x_t$ satisfying:

- $1 \leq x_1, x_2, \ldots, x_t \leq n$
- $2 \leq t \leq k$
- $x_1, x_2, \ldots, x_t$ are pairwise distinct.

The operating cost of this route is $d_{x_1,x_2} + d_{x_2,x_3} + \ldots + d_{x_{t-1},x_t}$.

There are m important pairs of cities, where millions of people travel between every year. For each important pair of cities (u, v), the government wants to have at least one route passing through both u and v. Note that the order does not matter, meaning that two important pairs (u, v) and (v, u) are considered the same.

Please help the country PVH to design train routes so as to cover all important pairs of cities, and the total operating cost of these routes is as small as possible.

Input

The first line contains a single integer – the number of test cases. Each test case is presented as follows:

- The first line contains three integers n, k and m $(2 \leq k \leq n \leq 19, 1 \leq m \leq 15)$.
- In the next $n-1$ lines, the i-th one $(1 \leq i \leq n-1)$ contains $n-i$ integers $d_{i,i+1}, d_{i,i+2}, \ldots, d_{i,n}$ $(1 \leq d_{i,j} \leq 1312)$.
- In the last m lines, each contains two integers u and v $(1 \leq u < v \leq n)$ representing an important pair of cities.

It is guaranteed that:

- The sum of n over all test cases does not exceed 95.
- The sum of m over all test cases does not exceed 75.

Output

For each test case:

- The first line contains two integers b and c — the minimum operating cost and the total number of train routes.
- In the last c lines, each contains an integer t $(2 \le t \le k)$ followed by t integers $x_1, x_2, \ldots, x_t$ $(1 \le x_i \le n)$ demonstrating a train route.

If there are multiple optimal solutions, you can output any of them.

Sample Input 1

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

Sample Output 1

```
20 2
3 4 3 5
3 2 1 3
17 2
4 4 1 3 5
3 2 1 3
```

Problem C: Classroom Carnival Chaos

Every year, Chikapu's school organizes a Carnival in March to help students chill before their final exams.

This year, Chikapu is in charge of the Carnival's parade. There are n classes, each sending exactly m students to participate in the parade. During the parade, the students are arranged in an n-row by m-column grid. The cell on the i-th row and j-th column is denoted by (i, j). Each cell in the grid has exactly one student. To promote diversity, Chikapu wants to ensure that no two students from the same class occupy the same column.

Now Chikapu wonders, how many different arrangements are possible? Two arrangements are considered different, if there is at least one cell occupied by different students.

As the number of arrangements be very large, output it modulo $(10^9 + 7)$.

Input

The first line contains a single integer t $(1 \le t \le 10^5)$ – the number of test cases. t test cases follow, each consists of two integers n and m in one line $(1 \le n, m \le 10^7)$.

Output

For each test case, print the answer in one line.

Sample Input 1

```
1
2 2
```

Sample Output 1

```
16
```

Sample Explanation

In the sample test case, there are 2 classes, each with 2 students. Let the classes be **A** and **B**. And let's denote the students in class **A** by **A1** and **A2**, and the students in class **B** with **B1** and **B2**.

There are **16** valid arrangements, as shown below:

B2	A2
A1	B1

B1	A2
A1	B2

B1	B2
A2	A1

A2	B2
B1	A1

A2	B1
B2	A1

A1	A2
B1	B2

A2	A1
B1	B2

A1	A2
B2	B1

B2	B1
A2	A1

A1	B1
B2	A2

B1	B2
A1	A2

B2	B1
A1	A2

B1	A1
A2	B2

A2	A1
B2	B1

B2	A1
A2	B1

A1	B2
B1	A2

Problem D: Distinguished Permutation

Given an array A of length n, we call it a **permutation** if it consists of n distinct integers from 1 to n in arbitrary order. For example, $[2, 3, 1, 5, 4]$ is a permutation, but $[1, 2, 2]$ is not a permutation (2 appears twice in the array), and $[1, 3, 4]$ is also not a permutation ($n = 3$ but there is 4 in the array).

For a permutation P, we define $F(P)$ as the number of contiguous subarrays of P that are permutations. For example, let's consider the permutation $P = (5, 3, 1, 4, 2)$, the following are all of its continuous sub-arrays that are permutations:

1. (1),
2. $(3, 1, 4, 2)$,
3. $(5, 3, 1, 4, 2)$.

Hence, $F(P) = 3$.

A permutation P of length n is called **distinguished** if $F(P)$ is the maximum amongst all permutations of length n.

You are given n and k. Consider all **distinguished** permutations of length n. Find the k-th lexicographically smallest permutation.

Note: A permutation a is lexicographically smaller than a permutation b of the same length if and only if: in the first position where a and b differ, the permutation a has a smaller element than the corresponding element in b.

Input

The first line contains a single integer t $(1 \leq t \leq 10^5)$ – the number of test cases. t test cases follow, each consists of two integers n and k in a single line $(1 \leq n \leq 10^5, 1 \leq k \leq 10^{18})$.

It is guaranteed that:

- k does not exceed the number of distinguished permutations of length n with maximum value of F,
- the sum of n over all test cases does not exceed 10^5.

Output

For each test case, print a single line containing n integers – the k-th lexicographically smallest **distinguished** permutation.

Sample Input 1	Sample Output 1
1 4 2	2 1 3 4

Sample Explanation

With $n = 4$, below are the first few permutations, lexicographically sorted:

- $1, 2, 3, 4$ with $F = 4$,
- $1, 2, 4, 3$ with $F = 3$,
- $1, 3, 2, 4$ with $F = 3$,
- $1, 3, 4, 2$ with $F = 2$,
- $1, 4, 2, 3$ with $F = 2$,
- $1, 4, 3, 2$ with $F = 2$,
- $2, 1, 3, 4$ with $F = 4$,

It can be shown that the maximum value of $F(P)$ over all permutations of length $n = 4$ is 4. The second lexicographically smallest permutation with $F = 4$ is $2, 1, 3, 4$ as per the list above.

Problem E: Easy Game

Bash is teaching Chikapu how to write. In order to make learning more exciting, Bash bought a strip of paper of length n, divided into n cells, and came up with the following game:

- Bash and Chikapu agree on a string s which does not contain three consecutive identical letters.
- They also agree on a string a.
- Bash and Chikapu take turns alternatively with Bash going first.
- On each turn, a player writes a letter on one of the n cells. The chosen cell must be empty before this turn, and the chosen letter must belong to the string a.
- If, after a turn, there are consecutive cells which form the string s, the player making this turn wins.
- If there are no empty cells left and no one wins, the game ends in a draw.

Now Chikapu wonders about the outcome of the game if both players play optimally.

Input

The input consists of several test cases. The first line contains an integer t $(1 \leq t \leq 100)$ – the number of test cases. The description of the test cases follows:

- The first line contains a single integer n $(1 \leq n \leq 100)$,
- The second line contains the string s. The length of s is at most 100.
- The third line contains a string a. The length of a is at most 26.

It is guaranteed that s and a only contain lowercase letters.

Output

For each test case, output:

- `Bash` if Bash wins,
- `Chikapu` if Chikapu wins,
- `Oh no!` if they draw.

Sample Input 1	Sample Output 1

```
2
50
p
abcdefghijklmnopqrstuvwxyz
2
pika
aeioupq
```
	```
	Bash
	Oh no!
	```

Sample Explanation

In the first test case, s is `p`, so Bash writes the letter `p` and wins.

In the second test case, s is `pika`. However, letter `k` is not in a, so neither player can write `k`. As the result, the game is a draw.

Problem F: FizzBuzz

You are likely very familiar with **FizzBuzz** – a basic introductory programming exercise: Given a positive integer n, print the positive integers from 1 to n sequentially. However, for numbers divisible by 3, print the string `fizz` instead of the number; for numbers divisible by 5, print the string `buzz` instead of the number. For numbers divisible by both 3 and 5, print the string `fizzbuzz`. For example, with $n = 16$, the output should be:

```
1 2 fizz 4 buzz fizz 7 8 fizz buzz 11 fizz 13 14 fizzbuzz 16
```

MofK is preparing a programming exercise for an introductory class. Since the class has not yet learned about loops, MofK modifies the FizzBuzz problem as follows:

Given three positive integers n, a, and b, output:

- `fizzbuzz` if n is divisible by both a and b,
- `fizz` if n is divisible by a but not by b,
- `buzz` if n is divisible by b but not by a,
- the value of n itself if n is divisible by neither a nor b.

It is guaranteed that $1 \le n \le 10^9$ and $1 \le a < b \le 10^9$.

After drafting the problem, MofK prepares the test cases and goes to sleep.

The next morning, upon arriving at class, MofK is astonished to discover that the test cases prepared the previous day were faulty! The input only contains the correct number n instead of the three numbers n, a, and b.

Given n and the expected result, MofK needs to quickly find two numbers a and b such that the result matches the expected one. Help MofK write a program that can run before class starts!

Input

The first line contains a single integer t $(1 \le t \le 10^5)$ – the number of test cases. Each of the next t lines contains a positive integer n $(1 \le n \le 10^9)$ and the string representing the expected result. It is guaranteed that the expected result is one of the strings `fizz`, `buzz`, `fizzbuzz`, or the number n.

Output

For each test case, print two positive integers a and b that satisfy $1 \le a < b \le 10^9$ such that running the original program with inputs n, a, and b produces the expected output. If no such a and b exist, print `-1 -1`.

If there are multiple answers, you can print any of them.

Sample Input 1	Sample Output 1

`3`
`3 fizz`
`4 4`
`15 fizzbuzz`
	`3 5`
	`3 5`
	`3 5`

Problem G: Gid-osh

During her next concert, Adobo is planning to perform a live cover of *Gid-osh*, a popular vocaloid song known for its wide vocal range required from the singer.

The song contains n music notes in total, numbered from 1 to n. The i-th note is represented by its pitch value a_i. The **vocal range** of the song is the difference between the pitch of the highest note and the pitch of the lowest note in the song. More formally, it is defined as $\max(a_1, a_2, \ldots, a_n) - \min(a_1, a_2, \ldots, a_n)$.

However, before the day of the concert, Adobo ate a bit too much sweet potato and flying onion. As a consequence, her voice changed, and she cannot perform songs requiring an excessively wide vocal range. Therefore, she decided to skip **exactly one note** from the song, so that the vocal range of the remaining song with $n - 1$ notes is minimized.

Help Adobo find the minimum possible vocal range after skipping exactly one note from the song and save her concert!

Input

The first line contains a integer t $(1 \leq t \leq 1\,000)$ — the number of test cases. The description of each test case is as follows.

- The first line contains a single integer n $(2 \leq n \leq 100)$ — the number of notes in the song.
- The second line contains n integers $a_1, a_2, \ldots, a_n$ $(1 \leq a_i \leq 100)$ — the pitches of the notes in the song.

Output

For each test case, print a single integer — the minimum possible vocal range possible of the remaining song after skipping exactly one note.

Sample Input 1	Sample Output 1

```
3
3
2 8 3
5
2 4 9 6 5
2
1 100
```
	```
	1
	4
	0
	```

Sample Explanation

In the first test case, there are three ways to skip exactly one note from the song:

- Removing the first note. The remaining song consists of two notes 8 and 3 and has the vocal range of $\max(8, 3) - \min(8, 3) = 5$.
- Removing the second note. The remaining song consists of two notes 2 and 3 and has the vocal range of $\max(2, 3) - \min(2, 3) = 1$.
- Removing the third note. The remaining song consists of two notes 2 and 8 and has the vocal range of $\max(2, 8) - \min(2, 8) = 6$.

Therefore, removing the 2^{nd} note is the most optimal way to achieve the minimum vocal range of the remaining song (which is 1).

In the second test case, the most optimal way is to skip the 3^{rd} note from the song. The remaining notes are $[2, 4, 6, 5]$, which has the vocal range of 4.

In the third test case, no matter which note of the two are removed, the vocal range of the remaining song containing a single note is always 0.

Problem H: Hash-shashin

Hashshashin, more commonly known as the *Order of Assassins*, was a prominent religious order best known for their, uh, *covert operations* against their opponents. In order to carry out such *covert operations* without alerting the outside world, the order members, known as *Assassins*, often communicate via rigorously encrypted messages. The sender first converts the message to a string s of length n consisting of only characters `0` and `1`. Then, the sender computes a sequence, aptly called *hash*, defined as follows:

- $hash_0 = 0$
- $hash_i = (2 \cdot hash_{i-1} + s_i) \bmod m, \forall 1 \le i \le n$

Here, s_i is the i-th character (numbered from 1) of the binary message and m is a positive integer agreed beforehand between the recipient and the sender. The sender then sends the sequence *hash* to the intended recipient.

It is easy to see that the message can be reconstructed with the knowledge of m, provided $m \ge 3$. Additionally, to ensure that the message has not been compromised, the sender will separately send another string *key* of length n that contains the comparison between each pair of consecutive elements in *hash*. In other words:

- key_i is "<" if $hash_{i-1} < hash_i$,
- key_i is "=" if $hash_{i-1} = hash_i$, and
- key_i is ">" if $hash_{i-1} > hash_i$.

Your task is to help Altaïr, a senior *Assassin*, on his upcoming *covert operation*. Altaïr expects to receive a message from his mentor Mualim outlining his next objectives. However, he only received the key; the message containing the *hash* has been intercepted by hostile spies. As it is obviously unsafe to send another message at the moment, Altaïr wants to recover as many characters of the original message as possible. More specifically, for each position i ($1 \le i \le n$), Altaïr wants to know whether s_i is constant for every possible message s that is consistent with the received key. It is possible that the key itself has been tampered and there exists no satisfying string s, in which case Altaïr also wants you to let him know. Please help him!

Input

The first line contains a single integer t ($1 \le t \le 10\,000$) — the number of test cases. The description of the test cases follows.

- The first line of each test case contains two integers n, m ($1 \le n \le 200\,000$, $3 \le m \le 10^9$) — the length of the original message and the predetermined modulo used for encryption.
- The second line contains a string key of length n, where:

- key_i is "<" if $hash_{i-1} < hash_i$,
- key_i is "=" if $hash_{i-1} = hash_i$, and
- key_i is ">" if $hash_{i-1} > hash_i$.

It is guaranteed that the sum of n over all test cases does not exceed $200\,000$.

Output

For each test case, let S be the set of all binary strings s that matches the given key. If S is empty, output "`impossible`" (without the quotes). Otherwise, output a string a of length n, where:

- a_i is "0" if $s_i = 0 \; \forall s \in S$,
- a_i is "1" if $s_i = 1 \; \forall s \in S$,
- a_i is "?" otherwise.

Sample Input 1

```
3
5 6
<<<>>
10 998244353
==<<<<<<<<
5 3
<>=<=
```

Sample Output 1

```
10???
001???????
impossible
```

Sample Explanation

In the first test case, consider string `s = 10101`, the corresponding hash and key are as follows:

i	0	1	2	3	4	5
s_i		1	0	1	0	1
$hash_i$	0	1	2	5	4	3
key_i		<	<	<	>	>

Thus, the string `10101` matches the given key. The remaining satisfying strings are `10010`, `10011`, and `10100`.

In the second test case, any string of length 10 and starts with `001` matches the given key.

In the third test case, it can be proven that no string of length 5 matches the given key.

Problem I: Inversland

The Kingdom of Inversland issues n denominations of currency, where the i-th denomination is worth $\frac{1}{p_i}$ dollars. To make exchanging money even more complicated, all p_i values are **distinct prime numbers**.

One drawback of this set of denominations is that in some cases, to pay a certain amount of money, the payer and the payee must exchange bills. For example, with $n = 2$ and $p = \{3, 5\}$, to pay $\frac{7}{15}$ dollars, the payer could give two $\frac{1}{3}$ bills, and receive a $\frac{1}{5}$ bill in change. It can be proven that there exists no way to pay $\frac{7}{15}$ dollars without requiring the payee to give back change. Conversely, to pay $\frac{8}{15}$ dollars, the payer can give a $\frac{1}{3}$ bill and a $\frac{1}{5}$ bill without receiving change.

To evaluate the significance of this drawback, the government wants to calculate how many values satisfy the following conditions:

- it is possible to pay that amount of money using the existing denominations, and
- it is **impossible** to pay that amount without requiring the payee to provide change.

Given n denominations, the government asks you, a programming team who has qualified for a regional competition, to calculate the number of such values and output it modulo $998\,244\,353$. In case there is an *infinite* number of values, please report it as well.

Input

The first line contains a single integer t $(1 \leq t \leq 10\,000)$ — the number of test cases. The description of the test cases follows.

- The first line contains an integer n $(2 \leq n \leq 200\,000)$ — the number of denominations.
- The second line contains n **distinct prime numbers** $p_1, p_2, \ldots, p_n$ $(2 \leq p_i < 998\,244\,353)$, describing the denomination set.
- It is guaranteed that the sum of n over all test cases does not exceed $200\,000$.

Output

For each test case, if the number of such values is infinite, print `infinity`. Otherwise, print the number of values modulo $998\,244\,353$.

Sample Input 1

```
2
2
3 5
10
941 947 953 967 971 977 983 991 997 998244341
```

Sample Output 1

```
4
679849251
```

Sample Explanation

In the first test case, it can be proven that any amount of the form $\frac{x}{15}$, where $x \in \mathbb{N}$ can be paid using $\frac{1}{3}$ and $\frac{1}{5}$ dollar bills. Among these, the following amounts cannot be paid without giving back change: $\frac{1}{15}$, $\frac{2}{15}$, $\frac{4}{15}$, and $\frac{7}{15}$.

Problem J: Jumbled Journey

Hanoi – the capital city of Vietnam has n intersections, connected by m two-way roads. The intersections are numbered from 1 to n, and the roads are numbered from 1 to m. The i-th road connects intersection u_i and intersection v_i with a length of w_i kilometers. The roads guarantee that it is possible to travel between any two intersections.

Tahp and Mei are competing for a position at a delivery company. To evaluate their abilities, the company has designed a problem: There is a package that needs delivering between n intersections, each intersection receives the package exactly once, and these deliveries are made on different days. Formally, the schedule can be represented as a permutation p of integers from 1 to n, the detailed schedule over the days is as follows:

- On day 1, the package is delivered from intersection p_1 to intersection p_2.
- On day 2, the package is delivered from intersection p_2 to intersection p_3.

  ...
- On day i, the package is delivered from intersection p_i to intersection p_{i+1}.

  ...
- On day $n-1$, package is delivered from intersection p_{n-1} to intersection p_n.

However, the schedule is not fixed and may be changed during the process. Hence, the risk is difficult to predict. In this problem, Tahp is responsible for choosing the permutation p, determining the order of the deliveries. Mei must then determine the optimal delivery *route* for each day. A *route* from u to v is a sequence of intersections $u = x_0, x_1, \ldots, x_k = v$, where each pair of consecutive intersections (x_i, x_{i+1}) is connected by a road. Tahp's target is to **maximize** the *risk of the schedule*, while Mei's aim is to **minimize** it.

The company evaluates the risk of the schedule using the following criteria:

- The *risk of a road* is its length.
- The *risk of a route* is the **highest risk** of a road along that route.
- The *risk of a schedule* is the **sum of the risk** over all delivery routes on each day.

You are required to find the risk of the schedule and a corresponding delivery schedule which achieves that risk, assuming both Tahp and Mei make optimal choices.

Input

The first line contains a single integer t $(1 \le t \le 5 \cdot 10^5)$ representing the number of test cases. The description of each test case is as follows:

- The first line contains two integers n and m $(2 \le n \le 5 \cdot 10^5; n-1 \le m \le 5 \cdot 10^5)$ – the number of intersections and the number of roads.
- The j-th line of the next m lines contains three integers u_j, v_j, w_j $(1 \le u_j, v_j \le n; 1 \le w_j \le 10^9)$ describing the j-th road.

It is guaranteed that:

- the sum of n over all test cases does not exceed $5 \cdot 10^5$,
- the sum of m over all test cases does not exceed $5 \cdot 10^5$, and
- it is possible to travel between any two intersections using the given roads.

Output

For each test case, print two lines:

- The first line contains the risk of the schedule.
- The second line contains the permutation $p_1, p_2, \ldots, p_n$ that Tahp has chosen to achieve the above risk.

If there are multiple solutions, print any of them.

Sample Input 1

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

Sample Output 1

```
6
3 2 4 1
3
2 1
20
2 4 1 5 3
```

Problem K: Kangokutou Exodus

After a long battle, the last Nightmare of the divine prison tower is down, and the way for the Blood Maidens and their friends to escape its underground jail is laid bare.

And with everyone fully packed up, it's time to go. Known as the one with the best agility of them all, Cinderella is tasked to spearhead the exodus of the tower that she and her friends had just cleared.

Now malformed, the divine prison tower consists of n corridors, linked together as a chain; that is, for $1 < i < n$, the i-th corridor only touches the $(i-1)$-th and the $(i+1)$-th one. The i-th corridor is a_i units long, and any two adjacent corridors are **perpendicular**. At the end of the last corridor, there is an exit.

To move as fast as possible — not just for her but everyone else — Cinderella has a fast sprinting ability that was once called "Twelve Dash", but through years of training, it's now a customizable ability named "m-dash", characterized by a positive integer m. This ability functions as follows.

- For each second, Cinderella and her friends sprint for m units of length.
- If in any second, Cinderella reaches the end of a corridor without sprinting exactly m units of length, she stops at the end of the corridor and suffers an extra second of delay, as the bump onto the wall or the exit will cause everyone to be in slight disorder, and they will take that delay second to reorganize.
- Whenever reaching the end of a corridor and not being stunned (or fully recovered from being stunned), Cinderella and her friends turn to the direction of the next corridor and prepare to sprint again, or escape the maze if they reached the exit. This action takes 0 seconds.

The value m cannot be changed when she starts running, but she can choose m at will before that. With that in mind, she wonders how fast she could escape this maze, but her focus is too drifted towards her glass pendant that complex calculations are not in the cards at the moment. Will you help her with that?

Input

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

- The first line of each test case contains an integer n ($1 \le n \le 10^6$) — the number of corridors in the maze.
- 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 lengths of the corridors.

It is guaranteed that the sum of n over all test cases does not exceed 10^6.

Output

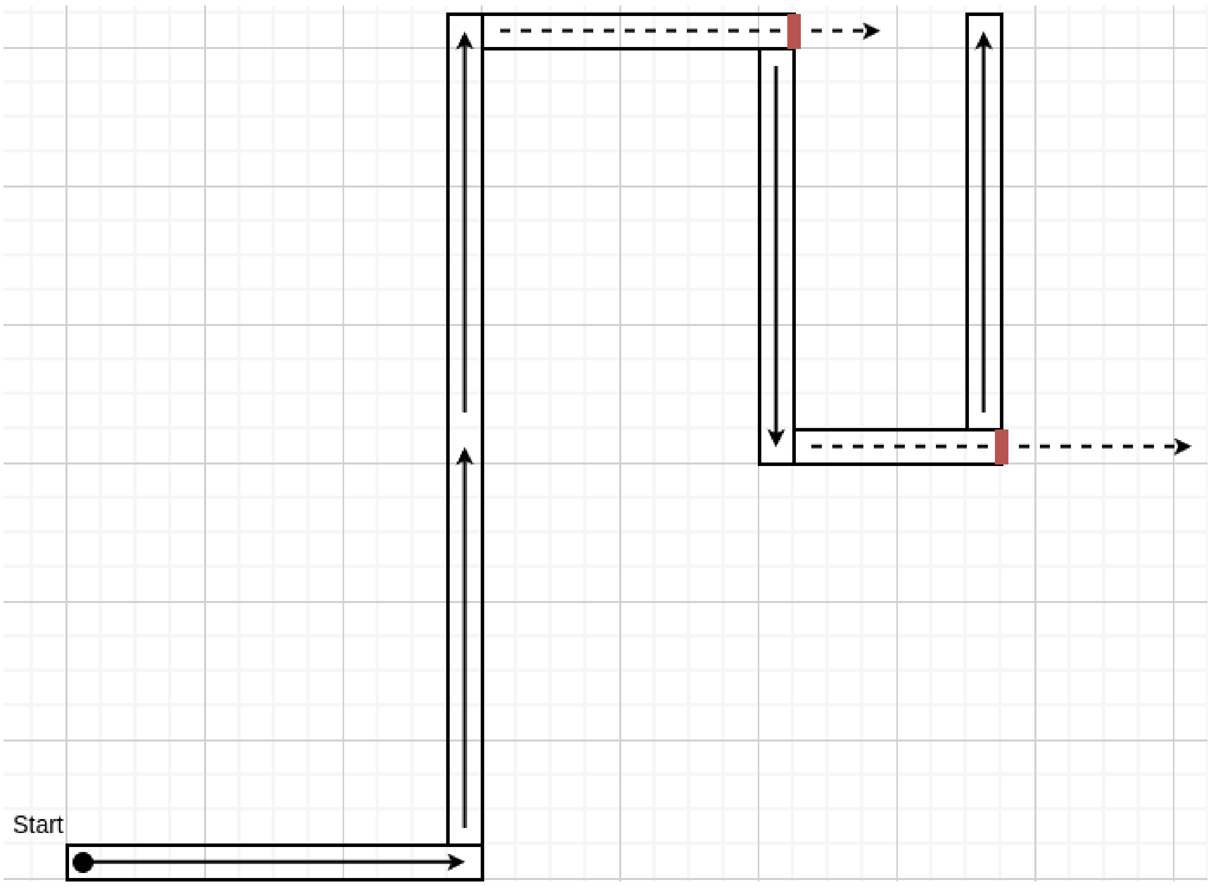
For each test case, output a single integer — the minimum time (in seconds) that Cinderella and her friends would take to escape the maze, given she could choose any value m before starting.

Sample Input 1	Sample Output 1

2
6
12 24 9 12 6 12
4
27 12 38 36
	9
	7

Sample Explanation

In the first test case, the best possible m is 12. The illustration below shows how Cinderella and her friends dashes through the corridors. Each arrow represents one sprint of length $m = 12$. The dashed arrow represents a dash with one extra second delay when bumping into the wall.



There are 7 sprints, with 2 delays, the total time to escape the tower is $7 + 2 = 9$ seconds.

Problem L: Locomotive Lane Logistics

Dooby is an energetic and meticulous train enthusiast, recently hired as a train operator. However, her first assignment is far from simple.

Dooby is responsible for managing a new train track connecting *Primstation* and *Krustown*. The train track is a straight line, with a total length of s meters. Each day, Dooby must create a schedule for the trains running on this track, and the list of trains varies day by day.

Initially, there are no trains. Dooby's job spans q days, and on the i-th day, one of the following events occurs:

- "$+ \ w_i \ v_i$" – a train with a length of w_i meters and a constant speed of v_i meters per hour is **added** to Dooby's train list.
- "$- \ w_i \ v_i$" – a train with a length of w_i meters and a constant speed of v_i meters per hour is **removed** from Dooby's train list. It is guaranteed that at least one such train exists in the current list.

Each day, all trains in Dooby's list start at *Primstation* and must travel to *Krustown*. Since there is only one track, the trains must be arranged in a specific order and start at specific times to **avoid collisions**. Once a train reaches *Krustown*, it is **immediately** moved to a shed, allowing the next train to proceed.

Before departure, each train's **front** end is aligned with the start of the track. A train is considered to have reached *Krustown* when its **rear** end crosses the far end of the track.

To demonstrate her efficiency as a train operator, Dooby must find the starting times for all trains each day so that:

- there must be no collision between two trains, and
- the time t_i required for all trains to reach *Krustown* is **minimized**.

Your task is to assist Dooby in her scheduling mission. For each of the q days, please determine the **minimum** possible time t_i required for all trains currently in the list to safely reach *Krustown*. Output t_i modulo $998\,244\,353$.

Formally, let $M = 998\,244\,353$. It can be shown that the answer can be expressed as an irreducible fraction $\frac{p}{q}$, where p and q are integers and $q \not\equiv 0 \pmod{M}$. Output the integer equal to $p \cdot q^{-1} \bmod M$. In other words, output such an integer x that $0 \le x < M$ and $x \cdot q \equiv p \pmod{M}$.

Collision condition

For the i-th train, let $\text{Rear}_i(t)$ and $\text{Front}_i(t)$ represent the positions of the **rear** end and the **front** end of the i-th train at time t, respectively.

By definition, we have $\text{Front}_i(t) = \text{Rear}_i(t) + w_i$.

Two trains i and j are considered to have **collided** if there exists a moment t such that:

- both the i-th and j-th trains have left *Primstation*,
- neither the i-th nor the j-th train has reached *Krustown*,
- $\min\{\text{Front}_i(t), \text{Front}_j(t)\} > \max\{\text{Rear}_i(t), \text{Rear}_j(t)\}$

Input

The first line contains two integers s and q ($1 \le s \le 10^9$, $1 \le q \le 200\,000$) – the length of the track in meters and the number of days Dooby's job lasts.

Each of the following q lines describes an event on the i-th day. It is one of the following types:

- "$+\ \ w_i\ v_i$" ($1 \le w_i, v_i \le 10^6$) – a train with a length of w_i meters and a constant speed of v_i meters per hour is **added** to Dooby's train list.
- "$-\ \ w_i\ v_i$" ($1 \le w_i, v_i \le 10^6$) – a train with a length of w_i meters and a constant speed of v_i meters per hour is **removed** from Dooby's train list. It is guaranteed that at least one such train exists in the current list.

Output

Print q lines. The i-th line should contain the **minimum** possible time required for all trains in the list on the i-th day to safely reach *Krustown*, modulo $998\,244\,353$. If on the i-th day, there is no train in Dooby's list, output `0`.

Sample Input 1

```
20 4
+ 2 2
+ 3 3
+ 5 2
- 2 2
```

Sample Output 1

```
11
12
499122191
499122190
```

Sample Explanation

Let (w, v) denote a train with length of w meters and speed of v meters per hour.

In the sample test, we have a train track of length 20 meters. Even though this is a small track, Dooby's efficient scheduling ensures every train reaches safely and quickly!

1. On the first day, there is only one train $(2, 2)$. The train takes $20/2 = 10$ hours for the **front** to reach *Krustown*, and $2/2 = 1$ hours for the rear to clear the track.
2. On the second day, a new train $(3, 3)$ was added.

   This train can start at $4\frac{1}{3}$ hours. It will reach *Krustown* at exactly 12 hours.

   Note that if the train $(3, 3)$ starts sooner, it can collide with the train $(2, 2)$. For example, suppose that the train $(3, 3)$ starts at 3 hours, the collision will happen at 7 hours.
3. On the third day, another train $(5, 2)$ was added. Here is one schedule that ensure the minimal time.

   - Train $(2, 2)$ starts at 0 hours;
   - Train $(5, 2)$ starts at 1 hours;
   - Train $(3, 3)$ starts at $6\frac{5}{6}$ hours.

   The time for the train $(3, 3)$ to reach *Krustown* is $\frac{29}{2}$ hours. The output is 499122191 because $499122191 \cdot 2 \equiv 29 \pmod{998\,244\,353}$.
4. On the forth day, the train $(2, 2)$ is removed. One optimal schedule is as follows:

   - Train $(3, 3)$ starts at 0 hours;
   - Train $(5, 2)$ starts at 1 hours.

   The time for the train $(5, 2)$ to reach *Krustown* is $\frac{27}{2}$ hours.

This page is intentionally left blank.

Problem M: Minotaur's Mysterious Maze

This is an interactive problem.

The Cretan Maze was an elaborate and confusing structure designed and built by the legendary artificer Daedalus for King Minos of Crete at Knossos. Its purpose was to imprison the Minotaur, a mythical creature with the head and tail of a bull and the body of a man.

Though the Minotaur is long gone, the Maze remains, with its magical walls shifting endlessly to confuse anyone who dares to enter. You, a bold adventurer, have become trapped in this ancient maze, surrounded by its eerie and ever-changing corridors.

The Maze consists of a series of **rooms**, each connected in a circular arrangement. Inside each room lies a single **enchanted torch**, which is either **lit** or **extinguished**. However, the number of rooms, n, is **unknown** to you. At any given moment, you can only observe the state of the torch in your current room and have no knowledge of the other rooms' torches.

To escape the Maze, you must deduce n – the number of rooms in the circle. With each step, you can move one room clockwise or counterclockwise, observe the torch's state, and change it if you wish. But be warned: you have only $n + 55$ **steps** to determine the number of torches before the maze shifts again, trapping you inside forever.

Interaction

The state of each torch is encoded by a number.

- 0 represents an **extinguished** torch;
- 1 represents a **lit** torch.

At the beginning, your program will read the number t ($1 \le t \le 555$), the number of test cases. For each test case, your program will interact with the jury's program in a loop. The steps of the loop are as follows:

- First, the jury's program will print a number x ($x = 0$ or $x = 1$) – the state of the torch at your current room.
- Then, your program can make a query by printing one of the following:
  - "**>** y" ($y = 0$ or $y = 1$) – set the current torch to the state corresponding to y and move one room clockwise. You continue the interaction loop.
  - "**<** y" ($y = 0$ or $y = 1$) – set the current torch to the state corresponding to y and move one room counterclockwise. You continue the interaction loop.
  - "**!** n" – answer that the number of rooms in the maze is n, and conclude the interaction for the current test case.

For a test case, your answer will be accepted if your program satisfies the following conditions:

- your program answers n correctly, and
- before answering, your program has not made more than $n + 55$ queries in total. The answer query does not count toward this limit.

It is guaranteed that the sum of n over all test cases does not exceed $10\,000$.

The number of torches n, as well as the initial state of each torch, are **fixed** before the interaction process, and they will only be changed according to your program's queries during the interaction process.

After outputting, do not forget to print a newline and flush the output. To do this, use:

- `fflush(stdout)` or `cout.flush()` in C++;
- `System.out.flush()` in Java;
- `stdout.flush()` in Python.

Sample interaction

The sample interaction is only for demonstrating the interaction process, and does not show how the answer is deduced.

stdin	stdout	Description
2		The number of test cases is 2.
1		The torch in the current room is lit (1).
	> 0	Set the torch to extinguished (0) and moves one room clockwise.
0		The torch in the current room is extinguished (0).
	> 1	Set the torch to lit (1) and moves one room clockwise.
1		The torch in the current room is lit (1).
	< 0	Set the torch to extinguished (0) and moves one room counterclockwise.
1		The torch in the current room is lit (1).
	< 1	Set the torch to lit (1) and moves one room counterclockwise.
0		The torch in the current room is extinguished (0).
	> 0	Set the torch to extinguished (0) and moves one room clockwise.
1		The torch in the current room is lit (1).
	! 3	Guesses that there are 3 rooms in the maze.
1		The torch in the current room is lit (1).
	< 0	Set the torch to extinguished (0) and moves one room counterclockwise.
0		The torch in the current room is extinguished (0).
	< 0	Set the torch to extinguished (0) and moves one room counterclockwise.
0		The torch in the current room is extinguished (0).
	! 4	Guesses that there are 4 rooms in the maze.

In the first test case, the initial state of the torches is $\underline{1}01$ in the clockwise order, and initially you are at the first torch. Here are the state of the torches after each of the query.

$$\underline{1}01 \xrightarrow{>\ 0} 0\underline{0}1 \xrightarrow{>\ 1} 01\underline{1} \xrightarrow{<\ 0} 0\underline{1}0 \xrightarrow{<\ 1} \underline{0}10 \xrightarrow{>\ 0} 0\underline{1}0$$