Codeforces Round #709 (Div. 1, based on Technocup 2021 Final Round)

A. Basic Diplomacy

2 seconds, 512 megabytes

Aleksey has n friends. He is also on a vacation right now, so he has m days to play this new viral cooperative game! But since it's cooperative, Aleksey will need one teammate in each of these m days.

On each of these days some friends will be available for playing, and all others will not. On each day Aleksey must choose one of his available friends to offer him playing the game (and they, of course, always agree). However, if any of them happens to be chosen strictly more than $\left\lceil \frac{m}{2} \right\rceil$ times, then all other friends are offended. Of course, Aleksey doesn't want to offend anyone.

Help him to choose teammates so that nobody is chosen strictly more than $\left\lceil \frac{m}{2} \right\rceil$ times.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10\,000$). Description of the test cases follows.

The first line of each test case contains two integers n and m ($1 \le n, m \le 100\,000$) standing for the number of friends and the number of days to play, respectively.

The i-th of the following m lines contains an integer k_i ($1 \le k_i \le n$), followed by k_i distinct integers f_{i1}, ..., f_{ik_i} ($1 \le f_{ij} \le n$), separated by spaces — indices of available friends on the day i.

It is guaranteed that the sums of n and m over all test cases do not exceed $100\,000$. It's guaranteed that the sum of all k_i over all days of all test cases doesn't exceed $200\,000$.

Output

Print an answer for each test case. If there is no way to achieve the goal, print "NO".

Otherwise, in the first line print "YES", and in the second line print m space separated integers c_1, ..., c_m. Each c_i must denote the chosen friend on day i (and therefore must be one of f_{ij}).

No value must occur more than $\left\lceil \frac{m}{2} \right\rceil$ times. If there is more than one possible answer, print any of them.

input

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

output

```
YES
1 2 1 1 2 3
NO
```

B. Playlist

2 seconds, 256 megabytes

Arkady has a playlist that initially consists of n songs, numerated from 1 to n in the order they appear in the playlist. Arkady starts listening to the songs in the playlist one by one, starting from song 1. The playlist is cycled, i. e. after listening to the last song, Arkady will continue listening from the beginning.

Each song has a genre a_i, which is a positive integer. Let Arkady finish listening to a song with genre y, and the genre of the next-to-last listened song be x. If $\gcd(x, y) = 1$, he deletes the last listened song (with genre y) from the playlist. After that he continues listening normally, skipping the deleted songs, and **forgetting** about songs he listened to before. In other words, after he deletes a song, he can't delete the next song immediately.

Here $\gcd(x, y)$ denotes the greatest common divisor (GCD) of integers x and y.

For example, if the initial songs' genres were $[5, 9, 2, 10, 15]$, then the playlist is converted as follows: [**5**, 9, 2, 10, 15] $\to$ [**5**, **9**, 2, 10, 15] $\to$ [5, 2, 10, 15] (because $\gcd(5, 9) = 1$) $\to$ [5, **2**, 10, 15] $\to$ [5, **2**, **10**, 15] $\to$ [5, 2, **10**, **15**] $\to$ [**5**, 2, 10, **15**] $\to$ [**5**, **2**, 10, 15] $\to$ [5, 10, 15] (because $\gcd(5, 2) = 1$) $\to$ [5, **10**, 15] $\to$ [5, **10**, **15**] $\to$ ... The bold numbers represent the two last played songs. Note that after a song is deleted, Arkady forgets that he listened to that and the previous songs.

Given the initial playlist, please determine which songs are eventually deleted and the order these songs are deleted.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10\,000$). Description of the test cases follows.

The first line of each test case contains a single integer n ($1 \le n \le 10^5$) — the number of songs.

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

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

Output

For each test case, print a single line. First, print a single integer k — the number of deleted songs. After that print k distinct integers: deleted songs in the order of their deletion.

input

```
5
5
5 9 2 10 15
6
1 2 4 2 4 2
2
1 2
1
1
1
2
```

output

```
2 2 3
2 2 1
2 2 1
1 1
0
```

Explanation of the first test case is given in the statement.

In the second test case, the playlist is converted as follows: [**1**, 2, 4, 2, 4, 2] → [**1**, **2**, 4, 2, 4, 2] → [1, 4, 2, 4, 2] (because $\gcd(1, 2) = 1$) → [1, **4**, 2, 4, 2] → [1, **4**, **2**, 4, 2] → [1, 4, **2**, **4**, 2] → [1, 4, 2, **4**, **2**] → [**1**, 4, 2, 4, **2**] → [4, 2, 4, 2] (because $\gcd(2, 1) = 1$) → [**4**, 2, 4, 2] → ...

In the third test case, the playlist is converted as follows: [**1**, 2] → [**1**, **2**] → [1] (because $\gcd(1, 2) = 1$) → [**1**] → [**1**] (Arkady listened to the same song twice in a row) → [] (because $\gcd(1, 1) = 1$).

The fourth test case is same as the third after deletion of the second song.

In the fifth test case, the same song is listened to over and over again, but since $\gcd(2, 2) \neq 1$, it is not deleted.

C. Skyline Photo

2.5 seconds, 256 megabytes

Alice is visiting New York City. To make the trip fun, Alice will take photos of the city skyline and give the set of photos as a present to Bob. However, she wants to find the set of photos with maximum beauty and she needs your help.

There are n buildings in the city, the i-th of them has positive height h_i. All n building heights in the city are different. In addition, each building has a beauty value b_i. Note that beauty can be positive or negative, as there are ugly buildings in the city too.

A set of photos consists of one or more photos of the buildings in the skyline. Each photo includes one or more buildings in the skyline that form a contiguous segment of indices. Each building needs to be in **exactly one** photo. This means that if a building does not appear in any photo, or if a building appears in more than one photo, the set of pictures is not valid.

The beauty of a photo is equivalent to the beauty b_i of the shortest building in it. The total beauty of a set of photos is the sum of the beauty of all photos in it. Help Alice to find the maximum beauty a valid set of photos can have.

Input

The first line contains an integer n ($1 \leq n \leq 3 \cdot 10^5$), the number of buildings on the skyline.

The second line contains n distinct integers $h_1, h_2, \ldots, h_n$ ($1 \leq h_i \leq n$). The i-th number represents the height of building i.

The third line contains n integers $b_1, b_2, \ldots, b_n$ ($-10^9 \leq b_i \leq 10^9$). The i-th number represents the beauty of building i.

Output

Print one number representing the maximum beauty Alice can achieve for a valid set of photos of the skyline.

input
5 1 2 3 5 4 1 5 3 2 4
output
15

input
5 1 4 3 2 5 -3 4 -10 2 7
output
10

input
2 2 1 -2 -3
output
-3

input
10 4 7 3 2 5 1 9 10 6 8 -4 40 -46 -8 -16 4 -10 41 12 3
output
96

In the first example, Alice can achieve maximum beauty by taking five photos, each one containing one building.

In the second example, Alice can achieve a maximum beauty of 10 by taking four pictures: three just containing one building, on buildings 1, 2 and 5, each photo with beauty -3, 4 and 7 respectively, and another photo containing building 3 and 4, with beauty 2.

In the third example, Alice will just take one picture of the whole city.

In the fourth example, Alice can take the following pictures to achieve maximum beauty: photos with just one building on buildings 1, 2, 8, 9, and 10, and a single photo of buildings 3, 4, 5, 6, and 7.

D. Useful Edges

5 seconds, 512 megabytes

You are given a weighted undirected graph on n vertices along with q triples (u, v, l), where in each triple u and v are vertices and l is a positive integer. An edge e is called *useful* if there is at least one triple (u, v, l) and a path (not necessarily simple) with the following properties:

- u and v are the endpoints of this path,
- e is one of the edges of this path,
- the sum of weights of all edges on this path doesn't exceed l.

Please print the number of useful edges in this graph.

Input

The first line contains two integers n and m ($2 \leq n \leq 600$, $0 \leq m \leq \frac{n(n-1)}{2}$).

Each of the following m lines contains three integers u, v and w ($1 \leq u, v \leq n$, $u \neq v$, $1 \leq w \leq 10^9$), denoting an edge connecting vertices u and v and having a weight w.

The following line contains the only integer q ($1 \leq q \leq \frac{n(n-1)}{2}$) denoting the number of triples.

Each of the following q lines contains three integers u, v and l ($1 \leq u, v \leq n$, $u \neq v$, $1 \leq l \leq 10^9$) denoting a triple (u, v, l).

It's guaranteed that:

- the graph doesn't contain loops or multiple edges;
- all pairs (u, v) in the triples are also different.

Output

Print a single integer denoting the number of useful edges in the graph.

input

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

output

```
5
```

input

```
4 2
1 2 10
3 4 10
6
1 2 11
1 3 11
1 4 11
2 3 11
2 4 11
3 4 9
```

output

```
1
```

input

```
3 2
1 2 1
2 3 2
1
1 2 5
```

output

```
2
```

In the first example each edge is useful, except the one of weight 5.

In the second example only edge between 1 and 2 is useful, because it belongs to the path $1 - 2$, and $10 \leq 11$. The edge between 3 and 4, on the other hand, is not useful.

In the third example both edges are useful, for there is a path $1 - 2 - 3 - 2$ of length exactly 5. Please note that the path may pass through a vertex more than once.

E. Vabank

2 seconds, 256 megabytes

Gustaw is the chief bank manager in a huge bank. He has unlimited access to the database system of the bank, in a few clicks he can move any amount of money from the bank's reserves to his own private account. However, the bank uses some fancy AI fraud detection system that makes stealing more difficult.

Gustaw knows that the anti-fraud system just detects any operation that exceeds some fixed limit M euros and these operations are checked manually by a number of clerks. Thus, any fraud operation exceeding this limit is detected, while any smaller operation gets unnoticed.

Gustaw doesn't know the limit M and wants to find it out. In one operation, he can choose some integer X and try to move X euros from the bank's reserves to his own account. Then, the following happens.

- If $X \leq M$, the operation is unnoticed and Gustaw's account balance raises by X euros.
- Otherwise, if $X > M$, the fraud is detected and cancelled. Moreover, Gustaw has to pay X euros from his own account as a fine. If he has less than X euros on the account, he is fired and taken to the police.

Initially Gustaw has 1 euro on his account. Help him find the exact value of M in no more than 105 operations without getting him fired.

Input

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

For each test case, there is no input prior to your first query, but you can be sure that M is integer, and $0 \leq M \leq 10^{14}$.

Output

For each test case, when you know the exact value of M, print a single line with format "`!` M". After that your program should proceed to the next test case or terminate, if it is the last one.

Interaction

When you want to make an operation, print a single line with format "`?` X", denoting that you try to move X euros ($1 \leq X \leq 10^{14}$). As a response, read a single line that can take the following values:

- "`Lucky!`", if $X \leq M$. Your balance raises by X.
- "`Fraudster!`", if $X > M$. Your balance decreases by X.
- "`Fired!`", if $X > M$, but you can't pay X euros of fine. In this case your program must terminate immediately. You also get this response if you exceed the number of queries.

You can make at most 105 such queries in each test case.

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

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

Hacks

To make a hack, prepare an input in the following format.

The first line contains a single integer t ($1 \leq t \leq 1000$) — the number of test cases.

Each test case is described with a line containing a single integer M ($0 \leq M \leq 10^{14}$).

input

```
1

Lucky!

Lucky!

Lucky!

Lucky!

Lucky!

Fraudster!
```

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

In the example $M = 5$, so the operation with $X = 6$ is detected. At this moment, Gustaw's balance is 16 euros, so he just pays fine.

F. Exam

2 seconds, 512 megabytes

This year a Chunin Selection Exam is held again in Konoha, and taking part in it are n ninjas named s_1, s_2, ..., s_n. All names are distinct. One of the exam stages consists of fights between the participants. This year the rules determining the ninjas for each fight are the following: ninjas i and j fight against each other if the following conditions are held:

- $i \neq j$;
- s_j is a substring of s_i;
- there is no k except i and j that s_j is a substring of s_k, and s_k is a substring of s_i.

A string a is a substring of a string b if a can be obtained from b by deletion of several (possibly, zero or all) characters from the beginning and several (possibly, zero or all) characters from the end.

Your task is to find out how many fights are going to take place this year.

Input

The first line consists of the only integer n ($1 \leq n \leq 10^6$) standing for the number of examinees.

The following n lines contain their names. No two names coincide, all names are non-empty and consist of lowercase English letters. The total length of all names doesn't exceed 10^6.

Output

Print the only integer standing for the number of fights.

input
5 hidan dan hanabi bi nabi
output
3

input
4 abacaba abaca acaba aca
output
4

In the first example `hidan` fights against `dan`, and `hanabi` fights against `nabi`, who also fights `bi`. Ninjas named `hanabi` and `bi` don't fight each other since there is the ninja called `nabi` who breaks the third condition for them.

In the second example the fights are held between `abacaba` and `acaba`, `abacaba` and `abaca`, `acaba` and `aca`, `abaca` and `aca`.