Good Bye 2019

A. Card Game

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Two players decided to play one interesting card game.

There is a deck of n cards, with values from 1 to n. The values of cards are **pairwise different** (this means that no two different cards have equal values). At the beginning of the game, the deck is completely distributed between players such that each player has at least one card.

The game goes as follows: on each turn, each player chooses one of their cards (whichever they want) and puts on the table, so that the other player doesn't see which card they chose. After that, both cards are revealed, and the player, value of whose card was larger, takes both cards in his hand. Note that as all cards have different values, one of the cards will be strictly larger than the other one. Every card may be played any amount of times. The player loses if he doesn't have any cards.

For example, suppose that $n = 5$, the first player has cards with values 2 and 3, and the second player has cards with values 1, 4, 5. Then one possible flow of the game is:

- The first player chooses the card 3. The second player chooses the card 1. As $3 > 1$, the first player gets both cards. Now the first player has cards 1, 2, 3, the second player has cards 4, 5.
- The first player chooses the card 3. The second player chooses the card 4. As $3 < 4$, the second player gets both cards. Now the first player has cards 1, 2. The second player has cards 3, 4, 5.
- The first player chooses the card 1. The second player chooses the card 3. As $1 < 3$, the second player gets both cards. Now the first player has only the card 2. The second player has cards 1, 3, 4, 5.
- The first player chooses the card 2. The second player chooses the card 4. As $2 < 4$, the second player gets both cards. Now the first player is out of cards and loses. Therefore, the second player wins.

Who will win if both players are playing optimally? It can be shown that one of the players has a winning strategy.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 100$). The description of the test cases follows.

The first line of each test case contains three integers n, k_1, k_2 ($2 \le n \le 100, 1 \le k_1 \le n - 1, 1 \le k_2 \le n - 1, k_1 + k_2 = n$) — the number of cards, number of cards owned by the first player and second player correspondingly.

The second line of each test case contains k_1 integers $a_1, \ldots, a_{k_1}$ ($1 \le a_i \le n$) — the values of cards of the first player.

The third line of each test case contains k_2 integers $b_1, \ldots, b_{k_2}$ ($1 \le b_i \le n$) — the values of cards of the second player.

It is guaranteed that the values of all cards are different.

Output

For each test case, output "YES" in a separate line, if the first player wins. Otherwise, output "NO" in a separate line. You can print each letter in any case (upper or lower).

Example

input
2 2 1 1 2 1 5 2 3 2 3 1 4 5
output
YES NO

Note

In the first test case of the example, there is only one possible move for every player: the first player will put 2, the second player will put 1. $2 > 1$, so the first player will get both cards and will win.

In the second test case of the example, it can be shown that it is the second player who has a winning strategy. One possible flow of the game is illustrated in the statement.

B. Interesting Subarray

time limit per test: 2 seconds

memory limit per test: 256 megabytes
input: standard input
output: standard output

For an array a of integers let's denote its maximal element as $\max(a)$, and minimal as $\min(a)$. We will call an array a of k integers **interesting** if $\max(a) - \min(a) \geq k$. For example, array $[1, 3, 4, 3]$ isn't interesting as $\max(a) - \min(a) = 4 - 1 = 3 < 4$ while array $[7, 3, 0, 4, 3]$ is as $\max(a) - \min(a) = 7 - 0 = 7 \geq 5$.

You are given an array a of n integers. Find some interesting **nonempty** subarray of a, or tell that it doesn't exist.

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

Input
The first line contains integer number t ($1 \leq t \leq 10\,000$). Then t test cases follow.

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

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

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

Output
For each test case, output "NO" in a separate line if there is no interesting nonempty subarray in a.

Otherwise, output "YES" in a separate line. In the next line, output two integers l and r ($1 \leq l \leq r \leq n$) — bounds of the chosen subarray. If there are multiple answers, print any.

You can print each letter in any case (upper or lower).

Example

input
3 5 1 2 3 4 5 4 2 0 1 9 2 2019 2020

output
NO YES 1 4 NO

Note
In the second test case of the example, one of the interesting subarrays is $a = [2, 0, 1, 9]$: $\max(a) - \min(a) = 9 - 0 = 9 \geq 4$.

C. Make Good

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Let's call an array $a_1, a_2, \ldots, a_m$ of nonnegative integer numbers **good** if $a_1 + a_2 + \cdots + a_m = 2 \cdot (a_1 \oplus a_2 \oplus \cdots \oplus a_m)$, where $\oplus$ denotes the bitwise XOR operation.

For example, array $[1, 2, 3, 6]$ is good, as $1 + 2 + 3 + 6 = 12 = 2 \cdot 6 = 2 \cdot (1 \oplus 2 \oplus 3 \oplus 6)$. At the same time, array $[1, 2, 1, 3]$ isn't good, as $1 + 2 + 1 + 3 = 7 \neq 2 \cdot 1 = 2 \cdot (1 \oplus 2 \oplus 1 \oplus 3)$.

You are given an array of length n: $a_1, a_2, \ldots, a_n$. Append at most 3 elements to it to make it good. Appended elements don't have to be different. It can be shown that the solution always exists under the given constraints. If there are different solutions, you are allowed to output any of them. Note that **you don't have to minimize the number of added elements!**. So, if an array is good already you are allowed to not append elements.

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

The first line of each test case contains a single integer n ($1 \leq n \leq 10^5$) — the size of the array.

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

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

Output
For each test case, output two lines.

In the first line, output a single integer s ($0 \leq s \leq 3$) — the number of elements you want to append.

In the second line, output s integers $b_1, \ldots, b_s$ ($0 \leq b_i \leq 10^{18}$) — the elements you want to append to the array.

If there are different solutions, you are allowed to output any of them.

Example

input
3 4 1 2 3 6 1 8 2 1 1

output
0 2 4 4 3 2 6 2

Note

In the first test case of the example, the sum of all numbers is 12, and their $\oplus$ is 6, so the condition is already satisfied.

In the second test case of the example, after adding $4, 4$, the array becomes $[8, 4, 4]$. The sum of numbers in it is 16, $\oplus$ of numbers in it is 8.

D. Strange Device

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

This problem is interactive.

We have hidden an array a of n **pairwise different** numbers (this means that no two numbers are equal). You can get some information about this array using a new device you just ordered on Amazon.

This device can answer queries of the following form: in response to the positions of k different elements of the array, it will return the position and value of the m-th among them in the ascending order.

Unfortunately, the instruction for the device was lost during delivery. However, you remember k, but don't remember m. Your task is to find m using queries to this device.

You can ask **not more than n queries**.

Note that the array a and number m are fixed before the start of the interaction and don't depend on your queries. In other words, **interactor is not adaptive**.

Note that you don't have to minimize the number of queries, and you don't need to guess array a. You just have to guess m.

Input

The first line contains two integers n and k ($1 \le k < n \le 500$) — the length of the array and the number of the elements in the query.

It is guaranteed that number m satisfies $1 \le m \le k$, elements $a_1, a_2, \ldots, a_n$ of the array satisfy $0 \le a_i \le 10^9$, and all of them are different.

Interaction

You begin the interaction by reading n and k.

To ask a question about elements on positions $x_1, x_2, \ldots, x_k$, in a separate line output

$? \; x_1 \; x_2 \; x_3 \; ... \; x_k$

Numbers in the query have to satisfy $1 \le x_i \le n$, and all x_i have to be different. Don't forget to 'flush', to get the answer.

In response, you will receive two integers pos and a_{pos} — the position in the array a of the m-th in ascending order element among $a_{x_1}, a_{x_2}, \ldots, a_{x_k}$, and the element on this position.

In case your query is invalid or you asked more than n queries, the program will print -1 and will finish interaction. You will receive a **Wrong answer** verdict. Make sure to exit immediately to avoid getting other verdicts.

When you determine m, output

$! \; m$

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.

Hack format

For the hacks use the following format:

The first line has to contain three integers n, k, m ($1 \le m \le k < n \le 500$) — the length of the array, number of elements in the query, and which in the ascending order number the device returns.

In the next line output n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 10^9$) — the elements of the array. They have to be pairwise different.

Example

input
4 3 4 9 4 9 4 9 1 2
output
? 2 3 4 ? 1 3 4 ? 1 2 4 ? 1 2 3 ! 3

Note

In the example, $n = 4$, $k = 3$, $m = 3$, $a = [2, 0, 1, 9]$.

E. Divide Points

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given a set of $n \ge 2$ **pairwise different** points with integer coordinates. Your task is to partition these points into two **nonempty** groups A and B, such that the following condition holds:

For every two points P and Q, write the Euclidean distance between them on the blackboard: if they belong to the **same** group — with a **yellow** pen, and if they belong to **different** groups — with a **blue** pen. **Then no yellow number is equal to any blue number**.

It is guaranteed that such a partition exists for any possible input. If there exist multiple partitions, you are allowed to output any of them.

Input

The first line contains one integer n ($2 \le n \le 10^3$) — the number of points.

The i-th of the next n lines contains two integers x_i and y_i ($-10^6 \le x_i, y_i \le 10^6$) — the coordinates of the i-th point.

It is guaranteed that all n points are pairwise different.

Output

In the first line, output a ($1 \le a \le n - 1$) — the number of points in a group A.

In the second line, output a integers — the indexes of points that you include into group A.

If there are multiple answers, print any.

Examples

input
3 0 0 0 1 1 0
output
1 1

input
4 0 1 0 -1 1 0 -1 0
output
2 1 2

input
3 -2 1 1 1

-1 0

output

1
2

input

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

output

1
6

input

2
-1000000 -1000000
1000000 1000000

output

1
1

Note

In the first example, we set point $(0, 0)$ to group A and points $(0, 1)$ and $(1, 0)$ to group B. In this way, we will have 1 yellow number $\sqrt{2}$ and 2 blue numbers 1 on the blackboard.

In the second example, we set points $(0, 1)$ and $(0, -1)$ to group A and points $(-1, 0)$ and $(1, 0)$ to group B. In this way, we will have 2 yellow numbers 2, 4 blue numbers $\sqrt{2}$ on the blackboard.

F. Awesome Substrings

time limit per test: 8 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Let's call a binary string s **awesome**, if it has at least 1 symbol 1 and length of the string is divisible by the number of 1 in it. In particular, `1`, `1010`, `111` are **awesome**, but `0`, `110`, `01010` aren't.

You are given a binary string s. Count the number of its **awesome** substrings.

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.

Input

The first line contains the string s ($1 \leq |s| \leq 200\,000$) consisting only of zeros and ones.

Output

Output a single number — the number of **awesome** substrings of s.

Examples

input

111

output

6

input

01010

output

10

input

0000

output

0

input

1111100000

output

25

Note
In the first sample, all substrings of s are **awesome**.

In the second sample, we have the following **awesome** substrings of s: 1 (2 times), 01 (2 times), 10 (2 times), 010 (2 times), 1010, 0101

In the third sample, no substring is **awesome**.

G. Subset with Zero Sum

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given n integers $a_1, a_2, \ldots, a_n$, such that for each $1 \le i \le n$ holds $i - n \le a_i \le i - 1$.

Find some nonempty subset of these integers, whose sum is equal to 0. It can be shown that such a subset exists under given constraints. If there are several possible subsets with zero-sum, you can find any of them.

Input
Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^6$). The description of the test cases follows.

The first line of each test case contains a single integer n ($1 \le n \le 10^6$).

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($i - n \le a_i \le i - 1$).

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

Output
For each test case, output two lines.

In the first line, output s ($1 \le s \le n$) — the number of elements in your subset.

In the second line, output s integers $i_1, i_2, \ldots, i_s$ ($1 \le i_k \le n$). All integers have to be pairwise different, and $a_{i_1} + a_{i_2} + \cdots + a_{i_s}$ has to be equal to 0. If there are several possible subsets with zero-sum, you can find any of them.

Example

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

Note
In the first example, we get sum is $a_1 = 0$.

In the second example, we get sum is $a_1 + a_4 + a_3 + a_2 = 0$.

H. Number of Components

time limit per test: 8 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Suppose that we have an array of n distinct numbers $a_1, a_2, \ldots, a_n$. Let's build a graph on n vertices as follows: for every pair of vertices $i < j$ let's connect i and j with an edge, if $a_i < a_j$. Let's define **weight** of the array to be the number of connected components in this graph. For example, weight of array $[1, 4, 2]$ is 1, weight of array $[5, 4, 3]$ is 3.

You have to perform q queries of the following form — change the value at some position of the array. After each operation, output the weight of the array. Updates are not independent (the change stays for the future).

Input
The first line contains two integers n and q ($1 \le n, q \le 5 \cdot 10^5$) — the size of the array and the number of queries.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^6$) — the initial array.

Each of the next q lines contains two integers pos and x ($1 \le pos \le n$, $1 \le x \le 10^6$, $x \ne a_{pos}$). It means that you have to make $a_{pos} = x$.

It's guaranteed that at every moment of time, all elements of the array are different.

Output
After each query, output the weight of the array.

Example

input
5 3 50 40 30 20 10 1 25 3 45 1 48

output
3 3 4

Note

After the first query array looks like $[25, 40, 30, 20, 10]$, the weight is equal to 3.

After the second query array looks like $[25, 40, 45, 20, 10]$, the weight is still equal to 3.

After the third query array looks like $[48, 40, 45, 20, 10]$, the weight is equal to 4.

I. Xor on Figures

time limit per test: 5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

There is given an integer k and a grid $2^k \times 2^k$ with some numbers written in its cells, cell (i, j) initially contains number a_{ij}. Grid is considered to be a torus, that is, the cell to the right of $(i, 2^k)$ is $(i, 1)$, the cell below the $(2^k, i)$ is $(1, i)$ There is also given a lattice figure F, consisting of t cells, where t is **odd**. F doesn't have to be connected.

We can perform the following operation: place F at some position on the grid. (Only translations are allowed, rotations and reflections are prohibited). Now choose any nonnegative integer p. After that, for each cell (i, j), covered by F, replace a_{ij} by $a_{ij} \oplus p$, where $\oplus$ denotes the bitwise XOR operation.

More formally, let F be given by cells $(x_1, y_1), (x_2, y_2), \ldots, (x_t, y_t)$. Then you can do the following operation: choose any x, y with $1 \le x, y \le 2^k$, any nonnegative integer p, and for every i from 1 to n replace number in the cell $(((x + x_i - 1) \bmod 2^k) + 1, ((y + y_i - 1) \bmod 2^k) + 1)$ with $a_{((x+x_i-1) \bmod 2^k)+1,((y+y_i-1) \bmod 2^k)+1} \oplus p$.

Our goal is to make all the numbers equal to 0. Can we achieve it? If we can, find the smallest number of operations in which it is possible to do this.

Input

The first line contains a single integer k ($1 \le k \le 9$).

The i-th of the next 2^k lines contains 2^k integers $a_{i1}, a_{i2}, \ldots, a_{i2^k}$ ($0 \le a_{ij} < 2^{60}$) — initial values in the i-th row of the grid.

The next line contains a single integer t ($1 \le t \le \min(99, 4^k)$, t is odd) — number of cells of figure.

i-th of next t lines contains two integers x_i and y_i ($1 \le x_i, y_i \le 2^k$), describing the position of the i-th cell of the figure.

It is guaranteed that all cells are different, but **it is not guaranteed that the figure is connected**.

Output

If it is impossible to make all numbers in the grid equal to 0 with these operations, output -1.

Otherwise, output a single integer — the minimal number of operations needed to do this. It can be shown that if it is possible to make all numbers equal 0, it is possible to do so in less than 10^{18} operations.

Example

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

output
3

Note

The figure and the operations for the example are shown above: