Day 2. Gaussian elimination, XOR, XOR-basis

1A. Gaussian elimination

2 s., 64 MB

Solve a system of linear equations $n \times n$ of the form $\sum_{j=1}^{n} a_{ij}x_j = b_i$. It is guaranteed that the solution exists and is unique.

Input
The first line contains the number n. The next n lines contain $n + 1$ numbers — coefficients of the variables and the right hand side.

Output
Print the values of the variables with an accuracy of at least 10^{-4}.

input
2 2.0 3.0 19.0 4.0 1.0 23.0
output
5.0 3.0

1B. Binary Gaussian

2 s., 64 MB

Given a set of n bit vectors of size n and another vector. Get this vector as xor of the original.

Input
The first line contains the number n ($1 \le n \le 300$). The next n lines contain the initial vectors, the last line contains the vector to be obtained.

Output
If there are no solutions, print `No solution`. If there are multiple solutions, print `Multiple solutions`. If there is only one solution, print the numbers of vectors (vectors are numbered starting from 0) that need to be added to get this vector. Print the numbers in ascending order.

input
3 100 111 101 010
output
1 2

input
3 100 111 011 010
output
No solution

input
3 111 010 101 000
output
Multiple solutions

1C. The Road to Berland is Paved With Good Intentions

1 second, 256 megabytes

Berland has n cities, some of them are connected by bidirectional roads. For each road we know whether it is asphalted or not.

The King of Berland Valera II wants to asphalt all roads of Berland, for that he gathered a group of workers. Every day Valera chooses exactly one city and orders the crew to asphalt all roads that come from the city. The valiant crew fulfilled the King's order in a day, then workers went home.

Unfortunately, not everything is as great as Valera II would like. The main part of the group were gastarbeiters — illegal immigrants who are enthusiastic but not exactly good at understanding orders in Berlandian. Therefore, having received orders to asphalt the roads coming from some of the city, the group asphalted all non-asphalted roads coming from the city, and vice versa, took the asphalt from the roads that had it.

Upon learning of this progress, Valera II was very upset, but since it was too late to change anything, he asked you to make a program that determines whether you can in some way asphalt Berlandian roads in at most n days. Help the king.

Input

The first line contains two space-separated integers n, m $(1 \le n \le 100;\ 1 \le m \le \frac{n \cdot (n-1)}{2})$ — the number of cities and roads in Berland, correspondingly. Next m lines contain the descriptions of roads in Berland: the i-th line contains three space-separated integers a_i, b_i, c_i $(1 \le a_i, b_i \le n;\ a_i \ne b_i;\ 0 \le c_i \le 1)$. The first two integers (a_i, b_i) are indexes of the cities that are connected by the i-th road, the third integer (c_i) equals 1, if the road was initially asphalted, and 0 otherwise.

Consider the cities in Berland indexed from 1 to n, and the roads indexed from 1 to m. It is guaranteed that between two Berlandian cities there is not more than one road.

Output

In the first line print a single integer x $(0 \le x \le n)$ — the number of days needed to asphalt all roads. In the second line print x space-separated integers — the indexes of the cities to send the workers to. Print the cities in the order, in which Valera send the workers to asphalt roads. If there are multiple solutions, print any of them.

If there's no way to asphalt all roads, print "`Impossible`" (without the quotes).

input

```
4 4
1 2 1
2 4 0
4 3 1
3 2 0
```

output

```
4
3 2 1 3
```

input

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

output

```
Impossible
```

2A. Xor queries

5 seconds, 256 megabytes

You're given an array a of n elements. You should answer q queries. Each query contains 2 numbers l and r and you want to find $a_l \oplus a_{l+1} \oplus \ldots \oplus a_r$.

Input

The first line on input contains a single integer n $(1 \le n \le 10^5)$.

The second lines contains n integers a_i ($1 \le a_i \le 10^9$) — the elements of the array.

The third line contains a single integer q ($1 \le n \le 10^5$) — the number of queries.

The next q lines contain 2 integers l_i and r_i each ($1 \le l_i \le r_i \le n$) – the segment for which the answer should be found.

Output

For each query print the answer in a separate line.

input
3 1 2 2 3 2 3 1 3 1 2
output
0 1 3

2B. Dima and a Bad XOR

1 second, 256 megabytes

Student Dima from Kremland has a matrix a of size $n \times m$ filled with non-negative integers.

He wants to select exactly one integer from each row of the matrix so that the `bitwise exclusive OR` of the selected integers is strictly greater than zero. Help him!

Formally, he wants to choose an integers sequence $c_1, c_2, \ldots, c_n$ ($1 \le c_j \le m$) so that the inequality $a_{1,c_1} \oplus a_{2,c_2} \oplus \ldots \oplus a_{n,c_n} > 0$ holds, where $a_{i,j}$ is the matrix element from the i-th row and the j-th column.

Here $x \oplus y$ denotes the bitwise XOR operation of integers x and y.

Input

The first line contains two integers n and m ($1 \le n, m \le 500$) — the number of rows and the number of columns in the matrix a.

Each of the next n lines contains m integers: the j-th integer in the i-th line is the j-th element of the i-th row of the matrix a, i.e. $a_{i,j}$ ($0 \le a_{i,j} \le 1023$).

Output

If there is no way to choose one integer from each row so that their bitwise exclusive OR is strictly greater than zero, print "`NIE`".

Otherwise print "`TAK`" in the first line, in the next line print n integers $c_1, c_2, \ldots c_n$ ($1 \le c_j \le m$), so that the inequality $a_{1,c_1} \oplus a_{2,c_2} \oplus \ldots \oplus a_{n,c_n} > 0$ holds.

If there is more than one possible answer, you may output any.

input
3 2 0 0 0 0 0 0
output
NIE

input
2 3 7 7 7 7 7 10
output
TAK 1 3

In the first example, all the numbers in the matrix are 0, so it is impossible to select one number in each row of the table so that their bitwise exclusive OR is strictly greater than zero.

In the second example, the selected numbers are 7 (the first number in the first line) and 10 (the third number in the second line), $7 \oplus 10 = 13$, 13 is more than 0, so the answer is found.

2C. Make Good

2 seconds, 256 megabytes

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 ($1 \le t \le 10\,000$). The 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 size of the array.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 10^9$) — 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 ($0 \le s \le 3$) — the number of elements you want to append.

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

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

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

output
0 2 4 4 3 2 6 2

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.

2D. Little Xor

2 seconds, 256 megabytes

Little Petya likes arrays that consist of non-negative integers a lot. Recently his mom has presented him one such array consisting of n elements. Petya immediately decided to find there a segment of consecutive elements, such that the xor of all numbers from this segment was maximal possible. Help him with that.

The xor operation is the bitwise exclusive "OR", that is denoted as "xor" in Pascal and "^" in C/C++/Java.

Input

The first line contains integer n ($1 \le n \le 100$) — the number of elements in the array. The second line contains the space-separated integers from the array. All numbers are non-negative integers strictly less than 2^{30}.

Output

Print a single integer — the required maximal xor of a segment of consecutive elements.

input
5 1 2 1 1 2
output
3

input
3 1 2 7
output
7

input
4 4 2 4 8
output
14

In the first sample one of the optimal segments is the segment that consists of the first and the second array elements, if we consider the array elements indexed starting from one.

The second sample contains only one optimal segment, which contains exactly one array element (element with index three).

2E. Mahmoud and Ehab and the xor

2 seconds, 256 megabytes

Mahmoud and Ehab are on the third stage of their adventures now. As you know, Dr. Evil likes sets. This time he won't show them any set from his large collection, but will ask them to create a new set to replenish his beautiful collection of sets.

Dr. Evil has his favorite evil integer x. He asks Mahmoud and Ehab to find a set of n distinct non-negative integers such the bitwise-xor sum of the integers in it is exactly x. Dr. Evil doesn't like big numbers, so any number in the set shouldn't be greater than 10^6.

Input

The only line contains two integers n and x ($1 \le n \le 10^5$, $0 \le x \le 10^5$) — the number of elements in the set and the desired bitwise-xor, respectively.

Output

If there is no such set, print "NO" (without quotes).

Otherwise, on the first line print "YES" (without quotes) and on the second line print n distinct integers, denoting the elements in the set is any order. If there are multiple solutions you can print any of them.

input
5 5
output
YES 1 2 4 5 7

input
3 6
output
YES 1 2 5

You can read more about the bitwise-xor operation here: https://en.wikipedia.org/wiki/Bitwise_operation#XOR

For the first sample $1 \oplus 2 \oplus 4 \oplus 5 \oplus 7 = 5$.

For the second sample $1 \oplus 2 \oplus 5 = 6$.

3A. Max XOR

2 s., 64 MB

Given a set of n numbers. Find the maximum number that can be obtained as a bit xor of the data.

Input
The first line contains the number n ($1 \le n \le 1000$). The next n lines contain the initial numbers a_i ($0 \le a_i \le 2^{63} - 1$).

Output
Print the maximum number you can get.

input
2 1 2
output
3

input
3 6 2 8
output
14

input
3 15 3 8
output
15

3B. Count different

2 s., 64 MB

Given a set of m bit vectors of size n ($1 \le n, m \le 50$). How many different vectors can be obtained as xor of these vectors.

Input
The first line contains numbers n and m. The next m lines contain vectors.

Output
Output the number of vectors that can be obtained

input
2 2 10 11
output
4

input
3 4 100 111 011 010

output

```
8
```

input

```
3 4
111
111
111
000
```

output

```
2
```

3C. Mahmoud and Ehab and yet another xor task

1 second, 512 megabytes

Ehab has an array a of n integers. He likes the bitwise-xor operation and he likes to bother Mahmoud so he came up with a problem. He gave Mahmoud q queries. In each of them, he gave Mahmoud 2 integers l and x, and asked him to find the number of subsequences of the first l elements of the array such that their bitwise-xor sum is x. Can you help Mahmoud answer the queries?

A subsequence can contain elements that are not neighboring.

Input

The first line contains integers n and q $(1 \le n, q \le 10^5)$, the number of elements in the array and the number of queries.

The next line contains n integers a_1, a_2, ..., a_n $(0 \le a_i < 2^{20})$, the elements of the array.

The next q lines, each contains integers l and x $(1 \le l \le n, 0 \le x < 2^{20})$, representing the queries.

Output

For each query, output its answer modulo $10^9 + 7$ in a newline.

input

```
5 5
0 1 2 3 4
4 3
2 0
3 7
5 7
5 8
```

output

```
4
2
0
4
0
```

input

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

output

```
4
2
```

The bitwise-xor sum of the empty set is 0 and the bitwise-xor sum of a set containing one element is that element itself.

3D. Weird Game

2 s., 512 MB

There are n two-sided cards, one side of the i-th card has number a_i written on it, while the other side has number b_i. At the beginning all cards are put on the table, each card only one of its sides up, and this side is chosen independently and uniformly. Thus the sequence $c_1, c_2, \ldots, c_n$ is obtained, where c_i is equal to a_i or b_i. Find the probability that $c_1 \oplus c_2 \oplus \ldots \oplus c_n \neq 0$

Input
The first line of the input contains a single integer n ($1 \le n \le 500\,000$) — the number of cards.

Each of the following n lines contains the description of one card, consisting of two integers a_i and b_i ($0 \le a_i, b_i \le 10^{18}$).

Output
Output the answer as an irreducible fraction p/q. If the probabilityis 0, print `0/1`.

input
2 1 1 1 1
output
0/1

input
2 1 2 1 2
output
1/2

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

3E. Ivan and Burgers

3 seconds, 256 megabytes

Ivan loves burgers and spending money. There are n burger joints on the street where Ivan lives. Ivan has q friends, and the i-th friend suggested to meet at the joint l_i and walk to the joint r_i ($l_i \le r_i$). While strolling with the i-th friend Ivan can visit all joints x which satisfy $l_i \le x \le r_i$.

For each joint Ivan knows the cost of the most expensive burger in it, it costs c_i burles. Ivan wants to visit some subset of joints on his way, in each of them he will buy the most expensive burger and spend the most money. But there is a small issue: his card broke and instead of charging him for purchases, the amount of money on it changes as follows.

If Ivan had d burles before the purchase and he spent c burles at the joint, then after the purchase he would have $d \oplus c$ burles, where $\oplus$ denotes the bitwise XOR operation.

Currently Ivan has $2^{2^{100}} - 1$ burles and he wants to go out for a walk. Help him to determine the maximal amount of burles he can spend if he goes for a walk with the friend i. The amount of burles he spends is defined as the difference between the initial amount on his account and the final account.

Input
The first line contains one integer n ($1 \le n \le 500\,000$) — the number of burger shops.

The next line contains n integers $c_1, c_2, \ldots, c_n$ ($0 \le c_i \le 10^6$), where c_i — the cost of the most expensive burger in the burger joint i.

The third line contains one integer q ($1 \le q \le 500\,000$) — the number of Ivan's friends.

Each of the next q lines contain two integers l_i and r_i ($1 \le l_i \le r_i \le n$) — pairs of numbers of burger shops between which Ivan will walk.

Output
Output q lines, i-th of which containing the maximum amount of money Ivan can spend with the friend i.

input

```
4
7 2 3 4
3
1 4
2 3
1 3
```

output

```
7
3
7
```

input

```
5
12 14 23 13 7
15
1 1
1 2
1 3
1 4
1 5
2 2
2 3
2 4
2 5
3 3
3 4
3 5
4 4
4 5
5 5
```

output

```
12
14
27
27
31
14
25
26
30
23
26
29
13
13
7
```

In the first test, in order to spend the maximum amount of money with the first and third friends, Ivan just needs to go into the first burger. With a second friend, Ivan just go to the third burger.

In the second test for a third friend (who is going to walk from the first to the third burger), there are only 8 options to spend money — 0, 12, 14, 23, $12 \oplus 14 = 2$, $14 \oplus 23 = 25$, $12 \oplus 23 = 27$, $12 \oplus 14 \oplus 23 = 20$. The maximum amount of money it turns out to spend, if you go to the first and third burger — $12 \oplus 23 = 27$.