Codeforces Round #440 (Div. 2, based on Technocup 2018 Elimination Round 2)

A. Search for Pretty Integers

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

You are given two lists of non-zero digits.

Let's call an integer pretty if its (base 10) representation has at least one digit from the first list and at least one digit from the second list. What is the smallest positive pretty integer?

Input

The first line contains two integers n and m ($1 \le n, m \le 9$) — the lengths of the first and the second lists, respectively.

The second line contains n distinct digits $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 9$) — the elements of the first list.

The third line contains m distinct digits $b_1, b_2, \ldots, b_m$ ($1 \le b_i \le 9$) — the elements of the second list.

Output

Print the smallest pretty integer.

Examples

input
2 3 4 2 5 7 6
output
25

input
8 8 1 2 3 4 5 6 7 8 8 7 6 5 4 3 2 1
output
1

Note

In the first example $25, 46, 24567$ are pretty, as well as many other integers. The smallest among them is 25. 42 and 24 are not pretty because they don't have digits from the second list.

In the second example all integers that have at least one digit different from 9 are pretty. It's obvious that the smallest among them is 1, because it's the smallest positive integer.

B. Maximum of Maximums of Minimums

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

You are given an array $a_1, a_2, \ldots, a_n$ consisting of n integers, and an integer k. You have to split the array into exactly k non-empty subsegments. You'll then compute the minimum integer on each subsegment, and take the maximum integer over the k obtained minimums. What is the maximum possible integer you can get?

Definitions of subsegment and array splitting are given in notes.

Input

The first line contains two integers n and k ($1 \le k \le n \le 10^5$) — the size of the array a and the number of subsegments you have to split the array to.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($-10^9 \le a_i \le 10^9$).

Output

Print single integer — the maximum possible integer you can get if you split the array into k non-empty subsegments and take maximum of minimums on the subsegments.

Examples

input
5 2 1 2 3 4 5
output
5

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

Note

A subsegment $[l, r]$ ($l \le r$) of array a is the sequence $a_l, a_{l+1}, \ldots, a_r$.

Splitting of array a of n elements into k subsegments $[l_1, r_1], [l_2, r_2], \ldots, [l_k, r_k]$ ($l_1 = 1, r_k = n, l_i = r_{i-1} + 1$ for all $i > 1$) is k sequences $(a_{l_1}, \ldots, a_{r_1}), \ldots, (a_{l_k}, \ldots, a_{r_k})$.

In the first example you should split the array into subsegments $[1, 4]$ and $[5, 5]$ that results in sequences $(1, 2, 3, 4)$ and (5). The minimums are $min(1, 2, 3, 4) = 1$ and $min(5) = 5$. The resulting maximum is $max(1, 5) = 5$. It is obvious that you can't reach greater result.

In the second example the only option you have is to split the array into one subsegment $[1, 5]$, that results in one sequence $(-4, -5, -3, -2, -1)$. The only minimum is $min(-4, -5, -3, -2, -1) = -5$. The resulting maximum is -5.

C. Maximum splitting

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

You are given several queries. In the i-th query you are given a single positive integer n_i. You are to represent n_i as a sum of maximum possible number of composite summands and print this maximum number, or print -1, if there are no such splittings.

An integer greater than 1 is composite, if it is not prime, i.e. if it has positive divisors not equal to 1 and the integer itself.

Input

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

q lines follow. The $(i+1)$-th line contains single integer n_i ($1 \le n_i \le 10^9$) — the i-th query.

Output

For each query print the maximum possible number of summands in a valid splitting to composite summands, or -1, if there are no such splittings.

Examples

input
1 12
output
3

input
2 6 8
output
1 2

input
3 1 2 3
output
-1 -1 -1

Note

$12 = 4 + 4 + 4 = 4 + 8 = 6 + 6 = 12$, but the first splitting has the maximum possible number of summands.

$8 = 4 + 4$, 6 can't be split into several composite summands.

1, 2, 3 are less than any composite number, so they do not have valid splittings.

D. Something with XOR Queries

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

This is an interactive problem.

Jury has hidden a permutation p of integers from 0 to n - 1. You know only the length n. Remind that in permutation all integers are distinct.

Let b be the inverse permutation for p, i.e. $p_{b_i} = i$ for all i. The only thing you can do is to ask `xor` of elements p_i and b_j, printing two indices i and j (not necessarily distinct). As a result of the query with indices i and j you'll get the value $p_i \oplus b_j$, where $\oplus$ denotes the `xor` operation. You can find the description of `xor` operation in notes.

Note that some permutations can remain indistinguishable from the hidden one, even if you make all possible n^2 queries. You have to compute the number of permutations indistinguishable from the hidden one, and print one of such permutations, making no more than $2n$ queries.

The hidden permutation does not depend on your queries.

Input

The first line contains single integer n ($1 \le n \le 5000$) — the length of the hidden permutation. You should read this integer first.

Output

When your program is ready to print the answer, print three lines.

In the first line print "`!`".

In the second line print single integer *answers_cnt* — the number of permutations indistinguishable from the hidden one, including the hidden one.

In the third line print n integers $p_0, p_1, \ldots, p_{n-1}$ ($0 \le p_i < n$, all p_i should be distinct) — one of the permutations indistinguishable from the hidden one.

Your program should terminate after printing the answer.

Interaction

To ask about `xor` of two elements, print a string "`? i j`", where i and j — are integers from 0 to n - 1 — the index of the permutation element and the index of the inverse permutation element you want to know the `xor`-sum for. After that print a line break and make `flush` operation.

After printing the query your program should read single integer — the value of $p_i \oplus b_j$.

For a permutation of length n your program should make no more than $2n$ queries about `xor`-sum. Note that printing answer doesn't count as a query. Note that you can't ask more than $2n$ questions. If you ask more than $2n$ questions or at least one incorrect question, your solution will get "`Wrong answer`".

If at some moment your program reads `-1` as an answer, it should immediately exit (for example, by calling `exit(0)`). You will get "`Wrong answer`" in this case, it means that you asked more than $2n$ questions, or asked an invalid question. If you ignore this, you can get other verdicts since your program will continue to read from a closed stream.

Your solution will get "`Idleness Limit Exceeded`", if you don't print anything or forget to flush the output, including for the final answer .

To flush you can use (just after printing line break):

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

Hacking

For hacking use the following format:

n

$p_0 p_1 \cdots p_{n-1}$

Contestant programs will not be able to see this input.

Examples

input
3 0 0 3 2 3

```
2
```

output

```
? 0 0
? 1 1
? 1 2
? 0 2
? 2 1
? 2 0
!
1
0 1 2
```

input

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

output

```
? 0 1
? 1 2
? 2 3
? 3 3
? 3 2
? 2 1
? 1 0
? 0 0
!
2
3 1 2 0
```

Note

`xor` operation, or bitwise exclusive OR, is an operation performed over two integers, in which the i-th digit in binary representation of the result is equal to 1 if and only if exactly one of the two integers has the i-th digit in binary representation equal to 1. For more information, see here.

In the first example $p = [0, 1, 2]$, thus $b = [0, 1, 2]$, the values $p_i \oplus b_j$ are correct for the given i, j. There are no other permutations that give the same answers for the given queries.

The answers for the queries are:

- $p_0 \oplus b_0 = 0 \oplus 0 = 0$,
- $p_1 \oplus b_1 = 1 \oplus 1 = 0$,
- $p_1 \oplus b_2 = 1 \oplus 2 = 3$,
- $p_0 \oplus b_2 = 0 \oplus 2 = 2$,
- $p_2 \oplus b_1 = 2 \oplus 1 = 3$,
- $p_2 \oplus b_0 = 2 \oplus 0 = 2$.

In the second example $p = [3, 1, 2, 0]$, and $b = [3, 1, 2, 0]$, the values $p_i \oplus b_j$ match for all pairs i, j. But there is one more suitable permutation $p = [0, 2, 1, 3]$, $b = [0, 2, 1, 3]$ that matches all n^2 possible queries as well. All other permutations do not match even the shown queries.

E. Points, Lines and Ready-made Titles

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

You are given n distinct points on a plane with integral coordinates. For each point you can either draw a vertical line through it, draw a horizontal line through it, or do nothing.

You consider several coinciding straight lines as a single one. How many distinct pictures you can get? Print the answer modulo $10^9 + 7$.

Input

The first line contains single integer n ($1 \le n \le 10^5$) — the number of points.

n lines follow. The $(i + 1)$-th of these lines contains two integers x_i, y_i ($-10^9 \le x_i, y_i \le 10^9$) — coordinates of the i-th point.

It is guaranteed that all points are distinct.

Output

Print the number of possible distinct pictures modulo $10^9 + 7$.

Examples

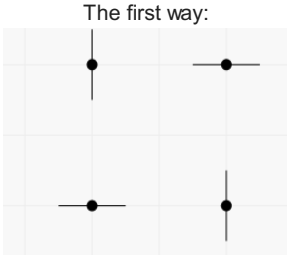
input
4 1 1 1 2 2 1 2 2
output
16

input
2 -1 -1 0 1
output
9

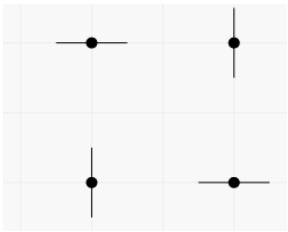
Note

In the first example there are two vertical and two horizontal lines passing through the points. You can get pictures with any subset of these lines. For example, you can get the picture containing all four lines in two ways (each segment represents a line containing it).

The first way:



The second way:



In the second example you can work with two points independently. The number of pictures is $3^2 = 9$.