Educational Codeforces Round 35 (Rated for Div. 2)

A. Nearest Minimums

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

You are given an array of n integer numbers $a_0, a_1, \ldots, a_{n-1}$. Find the distance between two closest (nearest) minimums in it. It is guaranteed that in the array a minimum occurs at least two times.

Input

The first line contains positive integer n ($2 \le n \le 10^5$) — size of the given array. The second line contains n integers $a_0, a_1, \ldots, a_{n-1}$ ($1 \le a_i \le 10^9$) — elements of the array. It is guaranteed that in the array a minimum occurs at least two times.

Output

Print the only number — distance between two nearest minimums in the array.

Examples

input
2 3 3
output
1

input
3 5 6 5
output
2

input
9 2 1 3 5 4 1 2 3 1
output
3

B. Two Cakes

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

It's New Year's Eve soon, so Ivan decided it's high time he started setting the table. Ivan has bought two cakes and cut them into pieces: the first cake has been cut into a pieces, and the second one — into b pieces.

Ivan knows that there will be n people at the celebration (including himself), so Ivan has set n plates for the cakes. Now he is thinking about how to distribute the cakes between the plates. Ivan wants to do it in such a way that all following conditions are met:

1. Each piece of each cake is put on some plate;
2. Each plate contains at least one piece of cake;
3. No plate contains pieces of both cakes.

To make his guests happy, Ivan wants to distribute the cakes in such a way that the minimum number of pieces on the plate is maximized. Formally, Ivan wants to know the maximum possible number x such that he can distribute the cakes according to the aforementioned conditions, and each plate will contain at least x pieces of cake.

Help Ivan to calculate this number x!

Input

The first line contains three integers n, a and b ($1 \le a, b \le 100$, $2 \le n \le a + b$) — the number of plates, the number of pieces of the first cake, and the number of pieces of the second cake, respectively.

Output

Print the maximum possible number x such that Ivan can distribute the cake in such a way that each plate will contain at least x pieces of cake.

Examples

input
5 2 3
output
1

input
4 7 10
output
3

Note

In the first example there is only one way to distribute cakes to plates, all of them will have 1 cake on it.

In the second example you can have two plates with 3 and 4 pieces of the first cake and two plates both with 5 pieces of the second cake. Minimal number of pieces is 3.

C. Three Garlands

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

Mishka is decorating the Christmas tree. He has got three garlands, and all of them will be put on the tree. After that Mishka will switch these garlands on.

When a garland is switched on, it periodically changes its state — sometimes it is lit, sometimes not. Formally, if i-th garland is switched on during x-th second, then it is lit only during seconds $x, x + k_i, x + 2k_i, x + 3k_i$ and so on.

Mishka wants to switch on the garlands in such a way that during each second after switching the garlands on there would be at least one lit garland. Formally, Mishka wants to choose three integers x_1, x_2 and x_3 (not necessarily distinct) so that he will switch on the first garland during x_1-th second, the second one — during x_2-th second, and the third one — during x_3-th second, respectively, and during each second starting from $max(x_1, x_2, x_3)$ at least one garland will be lit.

Help Mishka by telling him if it is possible to do this!

Input

The first line contains three integers k_1, k_2 and k_3 $(1 \le k_i \le 1500)$ — time intervals of the garlands.

Output

If Mishka can choose moments of time to switch on the garlands in such a way that each second after switching the garlands on at least one garland will be lit, print `YES`.

Otherwise, print `NO`.

Examples

input
2 2 3
output
YES

input
4 2 3
output
NO

Note

In the first example Mishka can choose $x_1 = 1, x_2 = 2, x_3 = 1$. The first garland will be lit during seconds $1, 3, 5, 7, \ldots$, the second — $2, 4, 6, 8, \ldots$, which already cover all the seconds after the 2-nd one. It doesn't even matter what x_3 is chosen. Our choice will lead third to be lit during seconds $1, 4, 7, 10, \ldots$, though.

In the second example there is no way to choose such moments of time, there always be some seconds when no garland is lit.

D. Inversion Counting

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

A permutation of size n is an array of size n such that each integer from 1 to n occurs exactly once in this array. An inversion in a permutation p is a pair of indices (i, j) such that $i > j$ and $a_i < a_j$. For example, a permutation $[4, 1, 3, 2]$ contains 4 inversions: $(2, 1)$, $(3, 1)$, $(4, 1)$, $(4, 3)$.

You are given a permutation a of size n and m queries to it. Each query is represented by two indices l and r denoting that you have to reverse the segment $[l, r]$ of the permutation. For example, if $a = [1, 2, 3, 4]$ and a query $l = 2, r = 4$ is applied, then the resulting permutation is $[1, 4, 3, 2]$.

After each query you have to determine whether the number of inversions is odd or even.

Input

The first line contains one integer n ($1 \le n \le 1500$) — the size of the permutation.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$) — the elements of the permutation. These integers are pairwise distinct.

The third line contains one integer m ($1 \le m \le 2 \cdot 10^5$) — the number of queries to process.

Then m lines follow, i-th line containing two integers l_i, r_i ($1 \le l_i \le r_i \le n$) denoting that i-th query is to reverse a segment $[l_i, r_i]$ of the permutation. All queries are performed one after another.

Output

Print m lines. i-th of them must be equal to `odd` if the number of inversions in the permutation after i-th query is odd, and `even` otherwise.

Examples

input
3 1 2 3 2 1 2 2 3

output
odd even

input
4 1 2 4 3 4 1 1 1 4 1 4 2 3

output
odd odd odd even

Note

The first example:

1. after the first query $a = [2, 1, 3]$, inversion: $(2, 1)$;
2. after the second query $a = [2, 3, 1]$, inversions: $(3, 1)$, $(3, 2)$.

The second example:

1. $a = [1, 2, 4, 3]$, inversion: $(4, 3)$;
2. $a = [3, 4, 2, 1]$, inversions: $(3, 1)$, $(4, 1)$, $(3, 2)$, $(4, 2)$, $(4, 3)$;
3. $a = [1, 2, 4, 3]$, inversion: $(4, 3)$;
4. $a = [1, 4, 2, 3]$, inversions: $(3, 2)$, $(4, 2)$.

E. Stack Sorting

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

Let's suppose you have an array a, a stack s (initially empty) and an array b (also initially empty).

You may perform the following operations until both a and s are empty:

- Take the first element of a, push it into s and remove it from a (if a is not empty);
- Take the top element from s, append it to the end of array b and remove it from s (if s is not empty).

You can perform these operations in arbitrary order.

If there exists a way to perform the operations such that array b is sorted in non-descending order in the end, then array a is called *stack-sortable*.

For example, $[3, 1, 2]$ is *stack-sortable*, because b will be sorted if we perform the following operations:

1. Remove 3 from a and push it into s;
2. Remove 1 from a and push it into s;
3. Remove 1 from s and append it to the end of b;
4. Remove 2 from a and push it into s;
5. Remove 2 from s and append it to the end of b;
6. Remove 3 from s and append it to the end of b.

After all these operations $b = [1, 2, 3]$, so $[3, 1, 2]$ is *stack-sortable*. $[2, 3, 1]$ is not *stack-sortable*.

You are given k first elements of some permutation p of size n (recall that a permutation of size n is an array of size n where each integer from 1 to n occurs exactly once). You have to restore the remaining $n - k$ elements of this permutation so it is *stack-sortable*. If there are multiple answers, choose the answer such that p is lexicographically maximal (an array q is lexicographically greater than an array p iff there exists some integer k such that for every $i < k$ $q_i = p_i$, and $q_k > p_k$). **You may not swap or change any of first k elements of the permutation**.

Print the lexicographically maximal permutation p you can obtain.

If there exists no answer then output `-1`.

Input

The first line contains two integers n and k ($2 \le n \le 200000$, $1 \le k < n$) — the size of a desired permutation, and the number of elements you are given, respectively.

The second line contains k integers $p_1, p_2, ..., p_k$ ($1 \le p_i \le n$) — the first k elements of p. These integers are pairwise distinct.

Output

If it is possible to restore a *stack-sortable* permutation p of size n such that the first k elements of p are equal to elements given in the input, print lexicographically maximal such permutation.

Otherwise print `-1`.

Examples

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

input
5 3 2 3 1
output
-1

input
5 1 3
output
3 2 1 5 4

input
5 2

3 4
output
-1

F. Tree Destruction

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

You are given an unweighted tree with n vertices. Then $n - 1$ following operations are applied to the tree. A single operation consists of the following steps:

1. choose two leaves;
2. add the length of the simple path between them to the answer;
3. remove one of the chosen leaves from the tree.

Initial answer (before applying operations) is 0. Obviously after $n - 1$ such operations the tree will consist of a single vertex.

Calculate the maximal possible answer you can achieve, and construct a sequence of operations that allows you to achieve this answer!

Input

The first line contains one integer number n ($2 \le n \le 2 \cdot 10^5$) — the number of vertices in the tree.

Next $n - 1$ lines describe the edges of the tree in form a_i, b_i ($1 \le a_i, b_i \le n$, $a_i \ne b_i$). It is guaranteed that given graph is a tree.

Output

In the first line print one integer number — maximal possible answer.

In the next $n - 1$ lines print the operations in order of their applying in format a_i, b_i, c_i, where a_i, b_i — pair of the leaves that are chosen in the current operation ($1 \le a_i, b_i \le n$), c_i ($1 \le c_i \le n$, $c_i = a_i$ or $c_i = b_i$) — choosen leaf that is removed from the tree in the current operation.

See the examples for better understanding.

Examples

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

input
5 1 2 1 3 2 4 2 5
output
9 3 5 5 4 3 3 4 1 1 4 2 2

G. Mass Change Queries

time limit per test: 3 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

You are given an array a consisting of n integers. You have to process q queries to this array; each query is given as four numbers l, r, x and y, denoting that for every i such that $l \le i \le r$ and $a_i = x$ you have to set a_i equal to y.

Print the array after all queries are processed.

Input

The first line contains one integer n ($1 \le n \le 200000$) — the size of array a.

The second line contains n integers a_1, a_2, ..., a_n ($1 \le a_i \le 100$) — the elements of array a.

The third line contains one integer q ($1 \le q \le 200000$) — the number of queries you have to process.

Then q lines follow. i-th line contains four integers l, r, x and y denoting i-th query ($1 \le l \le r \le n$, $1 \le x, y \le 100$).

Output

Print n integers — elements of array a after all changes are made.

Example

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