Codeforces Round #381 (Div. 1)

A. Alyona and mex

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

Alyona's mother wants to present an array of n non-negative integers to Alyona. The array should be special.

Alyona is a capricious girl so after she gets the array, she inspects m of its subarrays. Subarray is a set of some subsequent elements of the array. The i-th subarray is described with two integers l_i and r_i, and its elements are $a[l_i], a[l_i + 1], ..., a[r_i]$.

Alyona is going to find *mex* for each of the chosen subarrays. Among these m *mexes* the girl is going to find the smallest. She wants this minimum *mex* to be as large as possible.

You are to find an array a of n elements so that the minimum *mex* among those chosen by Alyona subarrays is as large as possible.

The *mex* of a set S is a minimum possible non-negative integer that is not in S.

Input

The first line contains two integers n and m ($1 \le n, m \le 10^5$).

The next m lines contain information about the subarrays chosen by Alyona. The i-th of these lines contains two integers l_i and r_i ($1 \le l_i \le r_i \le n$), that describe the subarray $a[l_i], a[l_i + 1], ..., a[r_i]$.

Output

In the first line print single integer — the maximum possible minimum *mex*.

In the second line print n integers — the array a. All the elements in a should be between 0 and 10^9.

It is guaranteed that there is an optimal answer in which all the elements in a are between 0 and 10^9.

If there are multiple solutions, print any of them.

Examples

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

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

Note

The first example: the *mex* of the subarray $(1, 3)$ is equal to 3, the *mex* of the subarray $(2, 5)$ is equal to 3, the *mex* of the subarray $(4, 5)$ is equal to 2 as well, thus the minumal *mex* among the subarrays chosen by Alyona is equal to 2.

B. Alyona and a tree

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

Alyona has a tree with n vertices. The root of the tree is the vertex 1. In each vertex Alyona wrote an positive integer, in the vertex i she wrote a_i. Moreover, the girl wrote a positive integer to every edge of the tree (possibly, different integers on different edges).

Let's define $dist(v, u)$ as the sum of the integers written on the edges of the simple path from v to u.

The vertex v controls the vertex u ($v \neq u$) if and only if u is in the subtree of v and $dist(v, u) \leq a_u$.

Alyona wants to settle in some vertex. In order to do this, she wants to know for each vertex v what is the number of vertices u such that v controls u.

Input

The first line contains single integer n ($1 \leq n \leq 2 \cdot 10^5$).

The second line contains n integers $a_1, a_2, ..., a_n$ ($1 \leq a_i \leq 10^9$) — the integers written in the vertices.

The next $(n - 1)$ lines contain two integers each. The i-th of these lines contains integers p_i and w_i ($1 \leq p_i \leq n, 1 \leq w_i \leq 10^9$) — the parent of the $(i + 1)$-th vertex in the tree and the number written on the edge between p_i and $(i + 1)$.

It is guaranteed that the given graph is a tree.

Output

Print n integers — the i-th of these numbers should be equal to the number of vertices that the i-th vertex controls.

Examples

input
5 2 5 1 4 6 1 7 1 1 3 5 3 6
output
1 0 1 0 0

input
5 9 7 8 6 5 1 1 2 1 3 1 4 1
output
4 3 2 1 0

Note

In the example test case the vertex 1 controls the vertex 3, the vertex 3 controls the vertex 5 (note that is doesn't mean the vertex 1 controls the vertex 5).

C. Alyona and towers

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

Alyona has built n towers by putting small cubes some on the top of others. Each cube has size $1 \times 1 \times 1$. A tower is a non-zero amount of cubes standing on the top of each other. The towers are next to each other, forming a row.

Sometimes Alyona chooses some segment towers, and put on the top of each tower several cubes. Formally, Alyouna chooses some segment of towers from l_i to r_i and adds d_i cubes on the top of them.

Let the sequence $a_1, a_2, ..., a_n$ be the heights of the towers from left to right. Let's call as a segment of towers $a_l, a_{l+1}, ..., a_r$ a hill if the following condition holds: there is integer k ($l \le k \le r$) such that $a_l < a_{l+1} < a_{l+2} < ... < a_k > a_{k+1} > a_{k+2} > ... > a_r$.

After each addition of d_i cubes on the top of the towers from l_i to r_i, Alyona wants to know the maximum width among all hills. The width of a hill is the number of towers in it.

Input

The first line contain single integer n ($1 \le n \le 3 \cdot 10^5$) — the number of towers.

The second line contain n integers $a_1, a_2, ..., a_n$ ($1 \le a_i \le 10^9$) — the number of cubes in each tower.

The third line contain single integer m ($1 \le m \le 3 \cdot 10^5$) — the number of additions.

The next m lines contain 3 integers each. The i-th of these lines contains integers l_i, r_i and d_i ($1 \le l \le r \le n$, $1 \le d_i \le 10^9$), that mean that Alyona puts d_i cubes on the tio of each of the towers from l_i to r_i.

Output

Print m lines. In i-th line print the maximum width of the hills after the i-th addition.

Example

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

Note

The first sample is as follows:

After addition of 2 cubes on the top of each towers from the first to the third, the number of cubes in the towers become equal to $[7, 7, 7, 5, 5]$. The hill with maximum width is $[7, 5]$, thus the maximum width is 2.

After addition of 1 cube on the second tower, the number of cubes in the towers become equal to $[7, 8, 7, 5, 5]$. The hill with maximum width is now $[7, 8, 7, 5]$, thus the maximum width is 4.

After addition of 1 cube on the fourth tower, the number of cubes in the towers become equal to $[7, 8, 7, 6, 5]$. The hill with maximum width is now $[7, 8, 7, 6, 5]$, thus the maximum width is 5.

D. Recover a functional graph

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

Functional graph is a directed graph in which all vertices have outdegree equal to 1. Loops are allowed.

Some vertices of a functional graph lay on a cycle. From the others we can come to a cycle by making a finite number of steps along the edges (we consider only finite functional graphs in this problem).

Let's compute two values for each vertex. $precycle_i$ is the amount of edges we should pass to get to a vertex which is a part of some cycle (zero, if i itself lies on a cycle), $cycle_i$ is the length of the cycle we get to.

You are given the information about these values for some functional graph. For each vertex you know the values $precycle_i$ and $cycle_i$, however, instead of some values there can be the question mark. It means that these values are unknown.

Build any functional graph that suits the description or determine that there is no such graph.

Input

The first line contains single integer n ($1 \le n \le 300$) — the number of vertices in the graph.

Each of the next n lines contain two integers — $precycle_i$ ($0 \le precycle_i \le n$ - 1) and $cycle_i$ ($1 \le cycle_i \le n$). There could be question marks instead of some of these values.

Output

In case there is no solution, print `-1`.

Otherwise, print n integers. i-th of them is the number of vertex to which the edge form the i-th vertex go.

The vertices should be in the same order as they go in input data.

If there are multiple solutions, print any of them.

Examples

input
3 0 3 0 3 ? ?

output
2 3 1

input
5 3 2 ? ? ? ? ? ? ? ?

output
5 3 2 2 4

input
8 ? 3 ? ? 0 2 0 2 0 3 0 3 0 3 3 3

output
5 1 4 3 6 7 5 2

input
1 ? ?

output
1

input

```
6
0 3
0 3
0 3
0 3
0 3
0 3
```

output

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

input

```
2
1 1
1 1
```

output

```
-1
```

E. Gosha is hunting

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

Gosha is hunting. His goal is to catch as many Pokemons as possible. Gosha has a Poke Balls and b Ultra Balls. There are n Pokemons. They are numbered 1 through n. Gosha knows that if he throws a Poke Ball at the i-th Pokemon he catches it with probability p_i. If he throws an Ultra Ball at the i-th Pokemon he catches it with probability u_i. He can throw at most one Ball of each type at any Pokemon.

The hunting proceeds as follows: at first, Gosha chooses no more than a Pokemons at which he will throw Poke Balls and no more than b Pokemons at which he will throw Ultra Balls. After that, he throws the chosen Balls at the chosen Pokemons. If he throws both Ultra Ball and Poke Ball at some Pokemon, he is caught if and only if he is caught by any of these Balls. The outcome of a throw doesn't depend on the other throws.

Gosha would like to know what is the expected number of the Pokemons he catches if he acts in an optimal way. In other words, he would like to know the maximum possible expected number of Pokemons can catch.

Input

The first line contains three integers n, a and b ($2 \le n \le 2000$, $0 \le a, b \le n$) — the number of Pokemons, the number of Poke Balls and the number of Ultra Balls.

The second line contains n real values $p_1, p_2, ..., p_n$ ($0 \le p_i \le 1$), where p_i is the probability of catching the i-th Pokemon if Gosha throws a Poke Ball to it.

The third line contains n real values $u_1, u_2, ..., u_n$ ($0 \le u_i \le 1$), where u_i is the probability of catching the i-th Pokemon if Gosha throws an Ultra Ball to it.

All the probabilities are given with exactly three digits after the decimal separator.

Output

Print the maximum possible expected number of Pokemons Gosha can catch. The answer is considered correct if it's absolute or relative error doesn't exceed 10^{-4}.

Examples

input
3 2 2 1.000 0.000 0.500 0.000 1.000 0.500
output
2.75

input
4 1 3 0.100 0.500 0.500 0.600 0.100 0.500 0.900 0.400
output
2.16

input
3 2 0 0.412 0.198 0.599 0.612 0.987 0.443
output
1.011