Codeforces Round #484 (Div. 2)

A. Row

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

You're given a row with $$$n$$$ chairs. We call a seating of people "maximal" if the two following conditions hold:

1. There are no neighbors adjacent to anyone seated.
2. It's impossible to seat one more person without violating the first rule.

The seating is given as a string consisting of zeros and ones ($$$0$$$ means that the corresponding seat is empty, $$$1$$$ — occupied). The goal is to determine whether this seating is "maximal".

Note that the first and last seats are **not** adjacent (if $$$n \ne 2$$$).

Input
The first line contains a single integer $$$n$$$ ($$$1 \leq n \leq 1000$$$) — the number of chairs.

The next line contains a string of $$$n$$$ characters, each of them is either zero or one, describing the seating.

Output
Output "`Yes`" (without quotation marks) if the seating is "maximal". Otherwise print "`No`".

You are allowed to print letters in whatever case you'd like (uppercase or lowercase).

Examples

input
3 101
output
Yes

input
4 1011
output
No

input
5 10001
output
No

Note
In sample case one the given seating is maximal.

In sample case two the person at chair three has a neighbour to the right.

In sample case three it is possible to seat yet another person into chair three.

B. Bus of Characters

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

In the Bus of Characters there are $$$n$$$ rows of seat, each having $$$2$$$ seats. The width of both seats in the $$$i$$$-th row is $$$w_i$$$ centimeters. All integers $$$w_i$$$ are distinct.

Initially the bus is empty. On each of $$$2n$$$ stops one passenger enters the bus. There are two types of passengers:

- an introvert always chooses a row where both seats are empty. Among these rows he chooses the one with the smallest seats width and takes one of the seats in it;
- an extrovert always chooses a row where exactly one seat is occupied (by an introvert). Among these rows he chooses the one with the largest seats width and takes the vacant place in it.

You are given the seats width in each row and the order the passengers enter the bus. Determine which row each passenger will take.

Input
The first line contains a single integer $$$n$$$ ($$$1 \le n \le 200\,000$$$) — the number of rows in the bus.

The second line contains the sequence of integers $$$w_1, w_2, \dots, w_n$$$ ($$$1 \le w_i \le 10^{9}$$$), where $$$w_i$$$ is the width of each of the seats in the $$$i$$$-th row. It is guaranteed that all $$$w_i$$$ are **distinct**.

The third line contains a string of length $$$2n$$$, consisting of digits '0' and '1' — the description of the order the passengers enter the bus. If the $$$j$$$-th character is '0', then the passenger that enters the bus on the $$$j$$$-th stop is an introvert. If the $$$j$$$-th character is '1', the the passenger that enters the bus on the $$$j$$$-th stop is an extrovert. It is guaranteed that the number of extroverts **equals** the number of introverts (i. e. both numbers equal $$$n$$$), and for each extrovert there **always** is a suitable row.

Output
Print $$$2n$$$ integers — the rows the passengers will take. The order of passengers should be the same as in input.

Examples

input
2 3 1 0011
output
2 1 1 2

input
6 10 8 9 11 13 5 010010011101
output
6 6 2 3 3 1 4 4 1 2 5 5

Note
In the first example the first passenger (introvert) chooses the row $$$2$$$, because it has the seats with smallest width. The second passenger (introvert) chooses the row $$$1$$$, because it is the only empty row now. The third passenger (extrovert) chooses the row $$$1$$$, because it has exactly one occupied seat and the seat width is the largest among such rows. The fourth passenger (extrovert) chooses the row $$$2$$$, because it is the only row with an empty place.

C. Cut 'em all!

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

You're given a tree with $$$n$$$ vertices.

Your task is to determine the maximum possible number of edges that can be removed in such a way that all the remaining connected components will have even size.

Input
The first line contains an integer $$$n$$$ ($$$1 \le n \le 10^5$$$) denoting the size of the tree.

The next $$$n - 1$$$ lines contain two integers $$$u$$$, $$$v$$$ ($$$1 \le u, v \le n$$$) each, describing the vertices connected by the $$$i$$$-th edge.

It's guaranteed that the given edges form a tree.

Output
Output a single integer $$$k$$$ — the maximum number of edges that can be removed to leave all connected components with even size, or $$$-1$$$ if it is impossible to remove edges in order to satisfy this property.

Examples

input
4 2 4 4 1 3 1
output

```
1
```

input

```
3
1 2
1 3
```

output

```
-1
```

input

```
10
7 1
8 4
8 10
4 7
6 5
9 3
3 5
2 10
2 5
```

output

```
4
```

input

```
2
1 2
```

output

```
0
```

Note

In the first example you can remove the edge between vertices $$$1$$$ and $$$4$$$. The graph after that will have two connected components with two vertices in each.

In the second example you can't remove edges in such a way that all components have even number of vertices, so the answer is $$$-1$$$.

D. Shark

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

For long time scientists study the behavior of sharks. Sharks, as many other species, alternate short movements in a certain location and long movements between locations.

Max is a young biologist. For $$$n$$$ days he watched a specific shark, and now he knows the distance the shark traveled in each of the days. All the distances are distinct. Max wants to know now how many locations the shark visited. He assumed there is such an integer $$$k$$$ that if the shark in some day traveled the distance strictly less than $$$k$$$, then it didn't change the location; otherwise, if in one day the shark traveled the distance greater than or equal to $$$k$$$; then it was changing a location in that day. Note that it is possible that the shark changed a location for several consecutive days, in each of them the shark traveled the distance at least $$$k$$$.

The shark never returned to the same location after it has moved from it. Thus, in the sequence of $$$n$$$ days we can find consecutive nonempty segments when the shark traveled the distance less than $$$k$$$ in each of the days: each such segment corresponds to one location. Max wants to choose such $$$k$$$ that the lengths of all such segments are equal.

Find such integer $$$k$$$, that the number of locations is as large as possible. If there are several such $$$k$$$, print the smallest one.

Input

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

The second line contains $$$n$$$ distinct positive integers $$$a_1, a_2, \ldots, a_n$$$ ($$$1 \leq a_i \leq 10^9$$$) — the distance traveled in each of the day.

Output

Print a single integer $$$k$$$, such that

1. the shark was in each location the same number of days,
2. the number of locations is maximum possible satisfying the first condition,
3. $$$k$$$ is smallest possible satisfying the first and second conditions.

Examples

input

```
8
```

1 2 7 3 4 8 5 6
output
7

input
6 25 1 2 3 14 36
output
2

Note

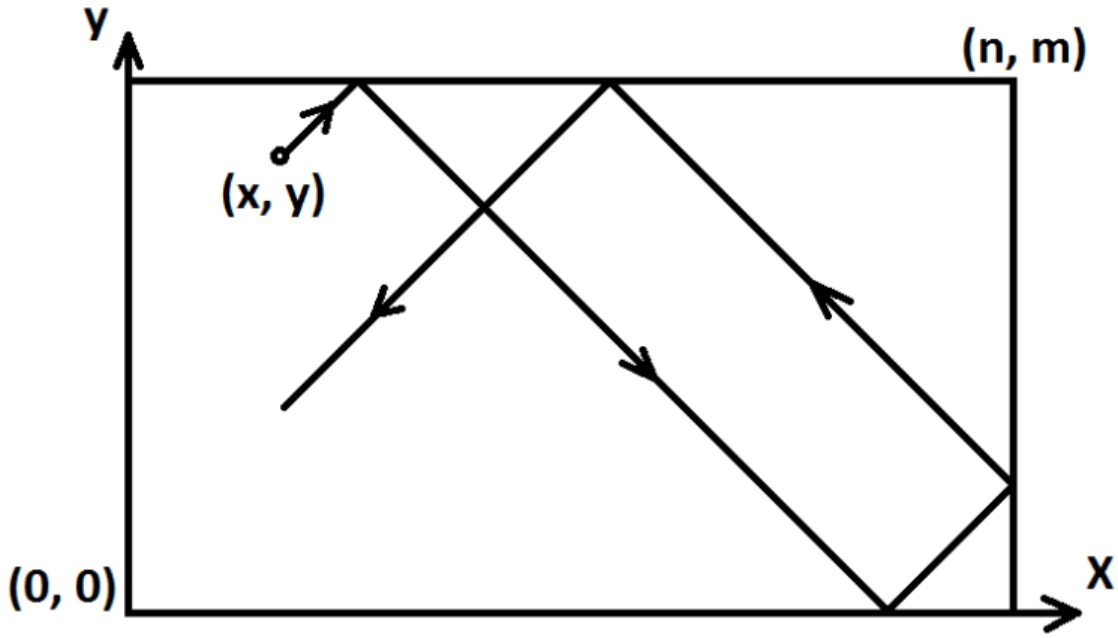
In the first example the shark travels inside a location on days $$$1$$$ and $$$2$$$ (first location), then on $$$4$$$-th and $$$5$$$-th days (second location), then on $$$7$$$-th and $$$8$$$-th days (third location). There are three locations in total.

In the second example the shark only moves inside a location on the $$$2$$$-nd day, so there is only one location.

E. Billiard

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

Consider a billiard table of rectangular size $$$n \times m$$$ with four pockets. Let's introduce a coordinate system with the origin at the lower left corner (see the picture).



There is one ball at the point $$$(x, y)$$$ currently. Max comes to the table and strikes the ball. The ball starts moving along a line that is parallel to one of the axes or that makes a $$$45^{\circ}$$$ angle with them. We will assume that:

1. the angles between the directions of the ball before and after a collision with a side are equal,
2. the ball moves indefinitely long, it only stops when it falls into a pocket,
3. the ball can be considered as a point, it falls into a pocket if and only if its coordinates coincide with one of the pockets,
4. initially the ball is not in a pocket.

Note that the ball can move along some side, in this case the ball will just fall into the pocket at the end of the side.

Your task is to determine whether the ball will fall into a pocket eventually, and if yes, which of the four pockets it will be.

Input

The only line contains $$$6$$$ integers $$$n$$$, $$$m$$$, $$$x$$$, $$$y$$$, $$$v_x$$$, $$$v_y$$$ ($$$1 \leq n, m \leq 10^9$$$, $$$0 \leq x \leq n$$$; $$$0 \leq y \leq m$$$; $$$-1 \leq v_x, v_y \leq 1$$$; $$$(v_x, v_y) \neq (0, 0)$$$) — the width of the table, the length of the table, the $$$x$$$-coordinate of the initial position of the ball, the $$$y$$$-coordinate of the initial position of the ball, the $$$x$$$-component of its initial speed and the $$$y$$$-component of its initial speed, respectively. It is guaranteed that the ball is not initially in a pocket.

Output

Print the coordinates of the pocket the ball will fall into, or $$$-1$$$ if the ball will move indefinitely.

Examples

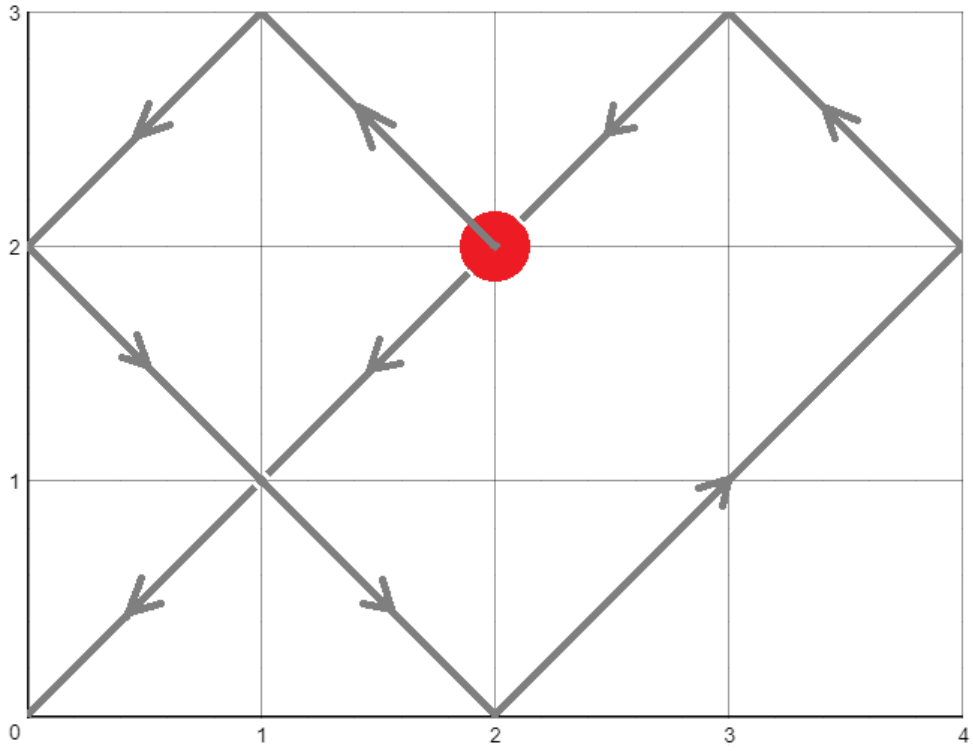
input
4 3 2 2 -1 1
output
0 0

input
4 4 2 0 1 1
output
-1

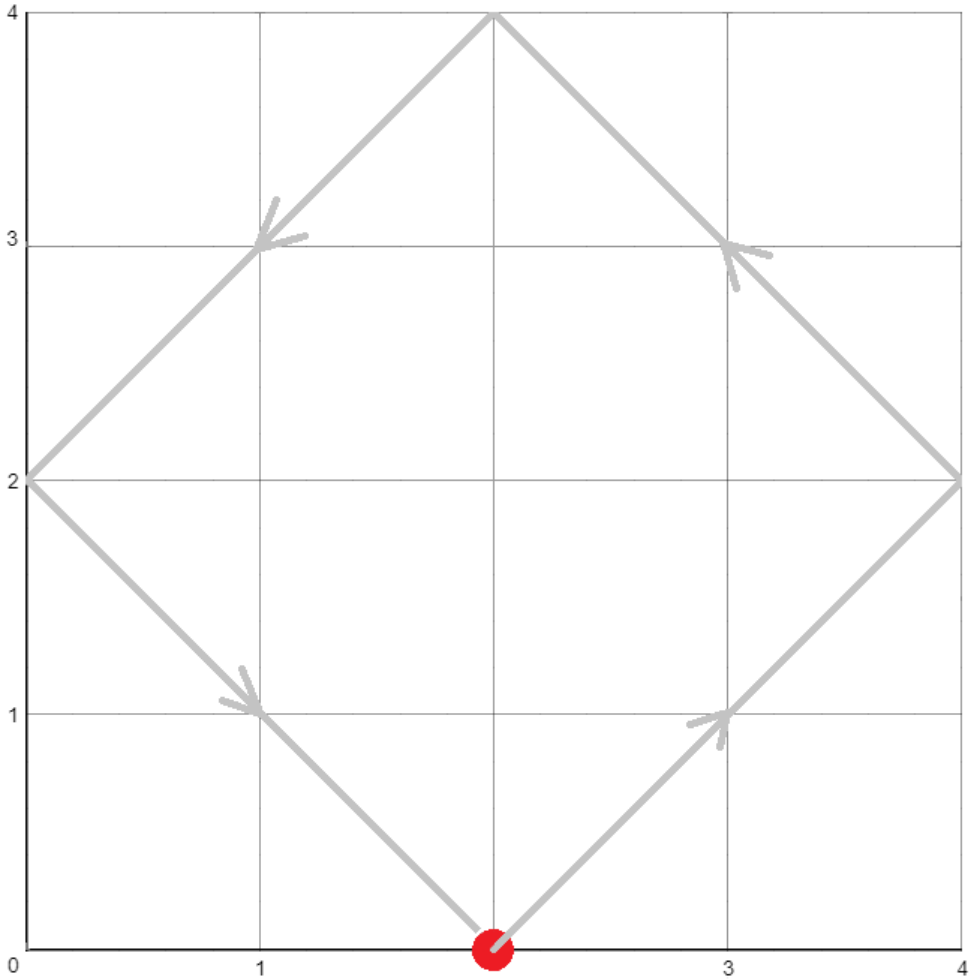
input
10 10 10 1 -1 0
output
-1

Note

The first sample:



The second sample:



In the third sample the ball will never change its $$$y$$$ coordinate, so the ball will never fall into a pocket.

F. The Meeting Place Cannot Be Changed

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

Petr is a detective in Braginsk. Somebody stole a huge amount of money from a bank and Petr is to catch him. Somebody told Petr that some luxurious car moves along the roads without stopping.

Petr knows that it is the robbers who drive the car. The roads in Braginsk are one-directional and each of them connects two intersections. Petr wants to select one intersection such that if the robbers continue to drive the roads indefinitely, they will sooner or later come to that intersection. The initial position of the robbers is unknown. Find such an intersection that fits the requirements.

Input

The first line of the input contains two integers $$$n$$$ and $$$m$$$ ($$$2 \leq n \le 10^5$$$, $$$2 \leq m \leq 5 \cdot 10^5$$$) — the number of intersections and the number of directed roads in Braginsk, respectively.

Each of the next $$$m$$$ lines contains two integers $$$u_i$$$ and $$$v_i$$$ ($$$1 \le u_i, v_i \le n$$$, $$$u_i \ne v_i$$$) — the start and finish of the $$$i$$$-th directed road. It is guaranteed that the robbers can move along the roads indefinitely.

Output

Print a single integer $$$k$$$ — the intersection Petr needs to choose. If there are multiple answers, print any. If there are no such intersections, print $$$-1$$$.

Examples

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

input
3 3 1 2 2 3 3 1

output
1

Note

In the first example the robbers can move, for example, along the following routes: $$$(1-2-3-1)$$$, $$$(3-4-5-3)$$$, $$$(1-2-3-4-5-3-1)$$$. We can show that if Petr chooses the $$$3$$$-rd intersection, he will eventually meet the robbers independently of their route.