Codeforces Round #509 (Div. 2)

A. Heist

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

There was an electronic store heist last night.

All keyboards which were in the store yesterday were numbered in ascending order from some integer number x. For example, if $x = 4$ and there were 3 keyboards in the store, then the devices had indices 4, 5 and 6, and if $x = 10$ and there were 7 of them then the keyboards had indices 10, 11, 12, 13, 14, 15 and 16.

After the heist, only n keyboards remain, and they have indices $a_1, a_2, \dots, a_n$. Calculate the minimum possible number of keyboards that have been stolen. The staff remember neither x nor the number of keyboards in the store before the heist.

Input

The first line contains single integer n $(1 \le n \le 1\,000)$ — the number of keyboards in the store that remained after the heist.

The second line contains n distinct integers $a_1, a_2, \dots, a_n$ $(1 \le a_i \le 10^9)$ — the indices of the remaining keyboards. The integers a_i are given in arbitrary order and are pairwise distinct.

Output

Print the minimum possible number of keyboards that have been stolen if the staff remember neither x nor the number of keyboards in the store before the heist.

Examples

input
```
4
10 13 12 8
```
output
```
2
```

input
```
5
7 5 6 4 8
```
output
```
0
```

Note

In the first example, if $x = 8$ then minimum number of stolen keyboards is equal to 2. The keyboards with indices 9 and 11 were stolen during the heist.

In the second example, if $x = 4$ then nothing was stolen during the heist.

B. Buying a TV Set

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

Monocarp has decided to buy a new TV set and hang it on the wall in his flat. The wall has enough free space so Monocarp can buy a TV set with screen width not greater than a and screen height not greater than b. Monocarp is also used to TV sets with a certain aspect ratio: formally, if the width of the screen is w, and the height of the screen is h, then the following condition should be met: $\frac{w}{h} = \frac{x}{y}$.

There are many different TV sets in the shop. Monocarp is sure that for any pair of **positive integers** w and h there is a TV set with screen width w and height h in the shop.

Monocarp isn't ready to choose the exact TV set he is going to buy. Firstly he wants to determine the optimal screen resolution. He has decided to try all possible variants of screen size. But he must count the number of pairs of **positive integers** w and h, beforehand, such that $(w \le a)$, $(h \le b)$ and $(\frac{w}{h} = \frac{x}{y})$.

In other words, Monocarp wants to determine the number of TV sets having aspect ratio $\frac{x}{y}$, screen width not exceeding a, and screen height not exceeding b. Two TV sets are considered different if they have different screen width or different screen height.

Input

The first line contains four integers a, b, x, y ($1 \le a, b, x, y \le 10^{18}$) — the constraints on the screen width and height, and on the aspect ratio.

Output

Print one integer — the number of different variants to choose TV screen width and screen height so that they meet the aforementioned constraints.

Examples

input
```
17 15 5 3
```
output
```
3
```

input
```
14 16 7 22
```
output
```
0
```

input
```
4 2 6 4
```
output
```
1
```

input
```
1000000000000000000 1000000000000000000 999999866000004473 999999822000007597
```
output
```
1000000063
```

Note

In the first example, there are 3 possible variants: $(5,3)$, $(10,6)$, $(15,9)$.

In the second example, there is no TV set meeting the constraints.

In the third example, there is only one variant: $(3,2)$.

C. Coffee Break

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

Recently Monocarp got a job. His working day lasts exactly m minutes. During work, Monocarp wants to drink coffee at certain moments: there are n minutes $a_1, a_2, \dots, a_n$, when he is able and willing to take a coffee break (for the sake of simplicity let's consider that each coffee break lasts exactly one minute).

However, Monocarp's boss doesn't like when Monocarp takes his coffee breaks too often. So for the given coffee break that is going to be on minute a_i, Monocarp must choose the day in which he will drink coffee during the said minute, so that every day at least d minutes pass between any two coffee breaks. Monocarp also wants to take these n coffee breaks in a minimum possible number of **working** days (he doesn't count days when he is not at work, and he doesn't take coffee breaks on such days). Take into account that more than d minutes pass between the end of any working day and the start of the following working day.

For each of the n given minutes determine the day, during which Monocarp should take a coffee break in this minute. You have to minimize the number of days spent.

Input

The first line contains three integers n, m, d $(1 \le n \le 2 \cdot 10^5, n \le m \le 10^9, 1 \le d \le m)$ — the number of coffee breaks Monocarp wants to have, the length of each working day, and the minimum number of minutes between any two consecutive coffee breaks.

The second line contains n distinct integers $a_1, a_2, \dots, a_n$ $(1 \le a_i \le m)$, where a_i is some minute when Monocarp wants to have a coffee break.

Output

In the first line, write the minimum number of days required to make a coffee break in each of the n given minutes.

In the second line, print n space separated integers. The i-th of integers should be the index of the day during which Monocarp should have a coffee break at minute a_i. Days are numbered from 1. If there are multiple optimal solutions, you may print any of them.

Examples

input

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

output

```
3
3 1 1 2
```

input

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

output

```
2
2 1 1 2 2 1 2 1 1 2
```

Note

In the first example, Monocarp can take two coffee breaks during the first day (during minutes 1 and 5, 3 minutes will pass between these breaks). One break during the second day (at minute 2), and one break during the third day (at minute 3).

In the second example, Monocarp can determine the day of the break as follows: if the minute when he wants to take a break is odd, then this break is on the first day, if it is even, then this break is on the second day.

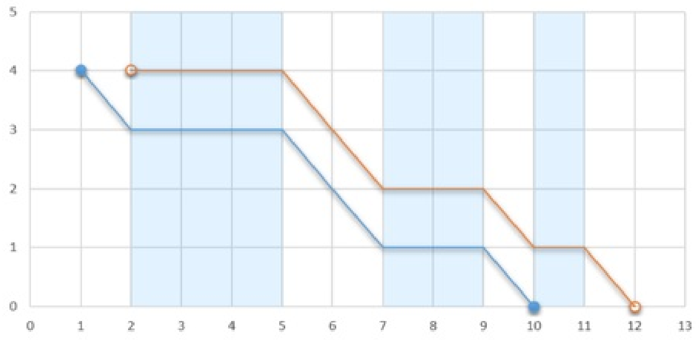
D. Glider

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

A plane is flying at a constant height of h meters above the ground surface. Let's consider that it is flying from the point $(-10^9, h)$ to the point $(10^9, h)$ parallel with Ox axis.

A glider is inside the plane, ready to start his flight at any moment (for the sake of simplicity let's consider that he may start only when the plane's coordinates are integers). After jumping from the plane, he will fly in the same direction as the plane, parallel to Ox axis, covering a unit of distance every second. Naturally, he will also descend; thus his second coordinate will decrease by one unit every second.

There are ascending air flows on certain segments, each such segment is characterized by two numbers x_1 and x_2 ($x_1 < x_2$) representing its endpoints. No two segments share any common points. When the glider is inside one of such segments, he doesn't descend, so his second coordinate stays the same each second. The glider still flies along Ox axis, covering one unit of distance every second.



If the glider jumps out at 1, he will stop at 10. Otherwise, if he jumps out at 2, he will stop at 12.

Determine the maximum distance along Ox axis from the point where the glider's flight starts to the point where his flight ends if the glider can choose any integer coordinate to jump from the plane and start his flight. After touching the ground the glider stops altogether, so he cannot glide through an ascending airflow segment if his second coordinate is 0.

Input

The first line contains two integers n and h $(1 \le n \le 2 \cdot 10^5, 1 \le h \le 10^9)$ — the number of ascending air flow segments and the altitude at which the plane is flying, respectively.

Each of the next n lines contains two integers x_{i1} and x_{i2} $(1 \le x_{i1} < x_{i2} \le 10^9)$ — the endpoints of the i-th ascending air flow segment. No two segments intersect, and they are given in ascending order.

Output

Print one integer — the maximum distance along Ox axis that the glider can fly from the point where he jumps off the plane to the point where he lands if he can start his flight at any integer coordinate.

Examples

input
```
3 4
2 5
7 9
10 11
```
output
```
10
```

input
```
5 10
5 7
11 12
16 20
25 26
30 33
```
output
```
18
```

input
```
1 1000000000
1 1000000000
```
output
```
1999999999
```

Note
In the first example if the glider can jump out at $(2, 4)$, then the landing point is $(12, 0)$, so the distance is $12 - 2 = 10$.

In the second example the glider can fly from $(16, 10)$ to $(34, 0)$, and the distance is $34 - 16 = 18$.

In the third example the glider can fly from $(-100, 1000000000)$ to $(1999999899, 0)$, so the distance is $1999999899 - (-100) = 1999999999$.

E. Tree Reconstruction

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

Monocarp has drawn a tree (an undirected connected acyclic graph) and then has given each vertex an index. All indices are distinct numbers from 1 to n. For every edge e of this tree, Monocarp has written two numbers: the maximum indices of the vertices of the two components formed if the edge e (and only this edge) is erased from the tree.

Monocarp has given you a list of $n - 1$ pairs of numbers. He wants you to provide an example of a tree that will produce the said list if this tree exists. If such tree does not exist, say so.

Input
The first line contains one integer n ($2 \le n \le 1\,000$) — the number of vertices in the tree.

Each of the next $n - 1$ lines contains two integers a_i and b_i each ($1 \le a_i < b_i \le n$) — the maximal indices of vertices in the components formed if the i-th edge is removed.

Output
If there is no such tree that can produce the given list of pairs, print "NO" (without quotes).

Otherwise print "YES" (without quotes) in the first line and the edges of the tree in the next $n - 1$ lines. Each of the last $n - 1$ lines should contain two integers x_i and y_i ($1 \le x_i, y_i \le n$) — vertices connected by an edge.

Note: The numeration of edges doesn't matter for this task. Your solution will be considered correct if your tree produces the same pairs as given in the input file (possibly reordered). That means that you can print the edges of the tree you reconstructed in any order.

Examples
input
```
4
3 4
1 4
3 4
```
output
```
YES
1 3
3 2
2 4
```

input

```
3
1 3
1 3
```

output

```
NO
```

input

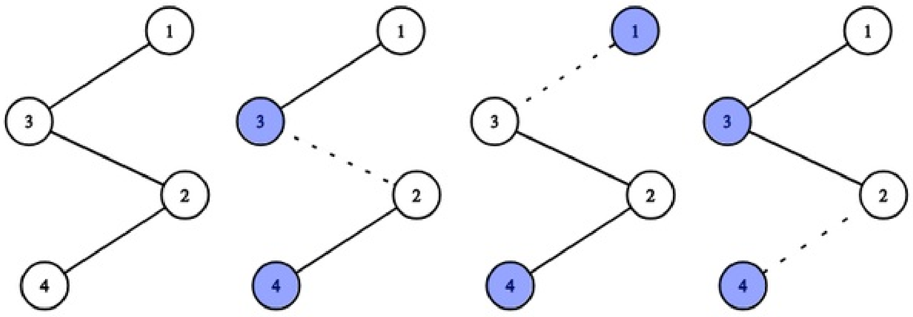
```
3
1 2
2 3
```

output

```
NO
```

Note

Possible tree from the first example. Dotted lines show edges you need to remove to get appropriate pairs.

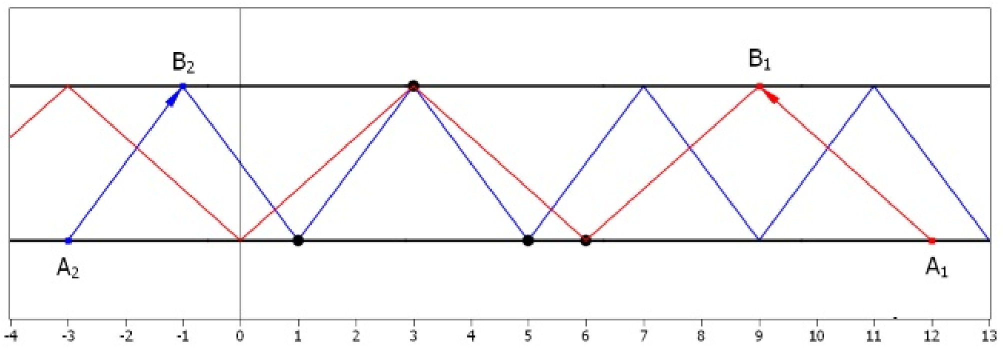


F. Ray in the tube

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

You are given a tube which is reflective inside represented as two non-coinciding, but parallel to Ox lines. Each line has some special integer points — positions of sensors on sides of the tube.

You are going to emit a laser ray in the tube. To do so, you have to choose **two** integer points A and B on the first and the second line respectively (coordinates can be negative): the point A is responsible for the position of the laser, and the point B — for the direction of the laser ray. The laser ray is a ray starting at A and directed at B which will reflect from the sides of the tube (it doesn't matter if there are any sensors at a reflection point or not). A sensor will only register the ray if the ray hits exactly at the position of the sensor.



Examples of laser rays. Note that image contains two examples. The 3 sensors (denoted by black bold points on the tube sides) will register the blue ray but only 2 will register the red.

Calculate the maximum number of sensors which can register your ray if you choose points A and B on the first and the second lines respectively.

Input

The first line contains two integers n and y_1 ($1 \le n \le 10^5$, $0 \le y_1 \le 10^9$) — number of sensors on the first line and its y coordinate.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 10^9$) — x coordinates of the sensors on the first line in the ascending order.

The third line contains two integers m and y_2 ($1 \le m \le 10^5$, $y_1 < y_2 \le 10^9$) — number of sensors on the second line and its y coordinate.

The fourth line contains m integers $b_1, b_2, \ldots, b_m$ ($0 \le b_i \le 10^9$) — x coordinates of the sensors on the second line in the ascending order.

Output

Print the only integer — the maximum number of sensors which can register the ray.

Example

input

```
3 1
1 5 6
1 3
3
```

output

```
3
```

Note

One of the solutions illustrated on the image by pair A_2 and B_2.