Problem A. Coffee Break

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

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 $a_1, a_2, \ldots, a_n$, 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 $a_1, a_2, \ldots, a_n$ $(1 \le a_i \le m)$, 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

standard input	standard output

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

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

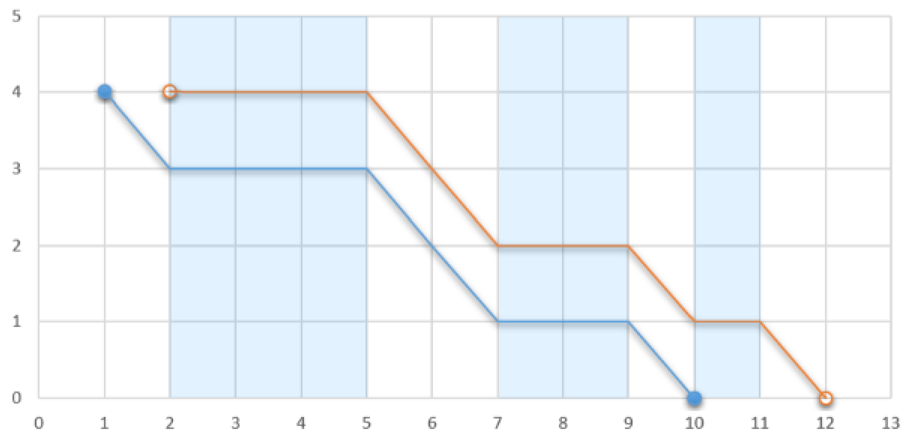
Problem B. Glider

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

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

standard input	standard output

3 4
2 5
7 9
10 11
	10

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

1 1000000000 1 1000000000	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 $1999999899 - (-100) = 1999999999$.

Problem C. Bacteria

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

Recently Monocarp has created his own mini-laboratory!

The laboratory contains n bacteria. Monocarp knows that he can merge any two bacteria having **equal** sizes, and the resulting bacterium will have the size equal to the sum of sizes of merged bacteria. For example, if two bacteria having sizes equal to 7 merge, one bacterium with size 14 is the result.

It becomes hard to watch for many bacteria, so Monocarp wants to merge all of them into one bacterium. It may not be possible to do this with the bacteria Monocarp has, so he can buy any number of bacteria of any possible integer sizes in a special store.

You have to determine the minimum number of bacteria Monocarp has to buy to merge them with the n bacteria his laboratory contains into exactly one bacterium.

Input

The first line contains one integer n ($1 \le n \le 2 \cdot 10^5$) — the number of bacteria Monocarp's laboratory contains.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$), where a_i is the size of the i-th bacterium in the laboratory.

Output

If it is impossible to merge the bacteria (possibly after buying some) into only one bacterium, print `-1`.

Otherwise print the minimum number of bacteria Monocarp has to buy to merge them with the n bacteria his laboratory contains into exactly one bacterium.

Examples

standard input
2 1 4
standard output
2

standard input
3 3 6 9
standard output
-1

standard input
7 1000000000 1000000000 1000000000 1000000000 1000000000 1000000000 1000000000
standard output
1

Note

In the first example Monocarp should buy one bacterium having size 1 and one bacterium having size 2. Then Monocarp will have 4 bacteria having sizes $[1, 4, 1, 2]$. Then two bacteria having sizes 1 can be merged into one having size 2. Then Monocarp will have 3 bacteria having sizes $[2, 4, 2]$. Then two bacteria having sizes 2 can be merged into one having size 4. Then Monocarp will have 2 bacteria having sizes $[4, 4]$, which can be merged into one having size 8.

In the second example no matter which bacteria Monocarp will buy, he cannot merge all his bacteria.

In the third example Monocarp needs to buy one bacterium having size 1000000000.

Problem D. Masquerade strikes back

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

Quite often the jury of Saratov SU use the problem "Masquerade" in different practice sessions before the contest. This problem is quite easy — all you need is to print the product of two integers which were read from the input stream.

As usual, the jury had prepared this problem once again. The jury had n testcases, the i-th testcase was a pair of positive integers a_i and b_i, both integers didn't exceed 10^7. All testcases were pairwise distinct.

Unfortunately, something went wrong. Due to hardware issues all testcases have disappeared. All that the jury were able to restore are the number of testcases n and the answers to these testcases, i. e. a sequence of n numbers $c_1, c_2, \ldots, c_n$, such that $a_i \cdot b_i = c_i$.

The jury ask you to help them. Can you provide any possible testset? Remember that all testcases were distinct and all numbers in each testcase were positive integers and didn't exceed 10^7.

Input

First line contains one insteger n ($1 \le n \le 2 \cdot 10^5$) — the number of lost testcases.

Second line contains n space-separated integers $c_1, c_2, \ldots, c_n$ ($1 \le c_i \le 10^7$) — the answers to the testcases.

Output

If there is no such testset, print `NO`.

Otherwise, print `YES` in first line. Then print n more lines, the i-th of them should contain two space separated positive integers a_i and b_i not exceeding 10^7. All pairs (a_i, b_i) must be distinct, and, for each $i \in [1, n]$, the condition $a_i \cdot b_i = c_i$ must be met.

Examples

standard input	standard output

`4`
`1 3 3 7`
	`YES`
	`1 1`
	`1 3`
	`3 1`
	`1 7`

`5`
`3 1 3 3 7`
	`NO`

`6`
`9 10 9 10 9 10`
	`YES`
	`1 9`
	`1 10`
	`3 3`
	`5 2`
	`9 1`
	`2 5`

Note

In the first example one of the possible testsets is ($a_1 = 1$, $b_1 = 1$), ($a_2 = 1$, $b_2 = 3$), ($a_3 = 3$, $b_3 = 1$), ($a_4 = 1$, $b_4 = 7$).

In the second example a testset consisting of distinct tests doesn't exist.

Problem E. Painting the Fence

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

There is a beautiful fence near Monocarp's house. The fence consists of n planks numbered from left to right. The i-th plank has color a_i.

Monocarp's father have decided to give his son m orders. Each order is a color c_j. After each order Monocarp finds leftmost and rightmost planks currently having color c_j and repaints all planks between them into color c_j.

For example, if, at first, fence looked like (from left to right) $[1, 2, 3, 1, 4, 1, 5, 6]$, then after fulfilling an order with color 1 fence will look like $[1, 1, 1, 1, 1, 1, 5, 6]$.

Assume that Monocarp fulfills all orders in the order they come, one by one.

Note that if current order is about color x and there is no more than one plank in the fence having color x, then Monocarp doesn't repaint anything, so he can skip this order and skip to the next one.

Find out the color of each plank after Monocarp has done all the given orders.

Input

The first line contains one integer n $(1 \le n \le 3 \cdot 10^5)$ — the number of planks in the fence.

The second line contains n space-separated integers $a_1, a_2, \ldots, a_n$ $(1 \le a_i \le 3 \cdot 10^5)$, where a_i is the initial color of the i-th plank.

The third line contains one integer m $(1 \le m \le 3 \cdot 10^5)$ — the number of orders.

The fourth line contains m space-separated integers $c_1, c_2, \ldots, c_m$ $(1 \le c_j \le 3 \cdot 10^5)$, where c_j is the color of the j-th order.

Output

Print n space-separated integers — the colors of planks in the fence after processing all m orders.

Examples

standard input	standard output

`4`
`1 2 1 2`
`2`
`2 1`
	`1 2 2 2`

`8`
`7 1 7 1 23 9 23 1`
`4`
`23 4 7 1`
	`7 7 7 1 1 1 1 1`

Note

In the first example initial appearance of the fence is $[1, 2, 1, 2]$. After the first order (color 2) fence will look like $[1, 2, 2, 2]$. After the second order (color 1) appearance of the fence will not change.

In the second example initial appearance of the fence is $[7, 1, 7, 1, 23, 9, 23, 1]$. After the first order (color 23) the fence will look like $[7, 1, 7, 1, 23, 23, 23, 1]$. After the second order (color 4) appearance of the fence will not change. After the third order (color 7) the fence will look like $[7, 7, 7, 1, 23, 23, 23, 1]$. After the fourth order (color 1) the fence will look like $[7, 7, 7, 1, 1, 1, 1, 1]$.

Problem F. Tickets

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Last n days Monocarp used public transport to get to the university. He received a ticket with number t_i during the i-th day.

Tickets' numbers are six digit non-negative integers with possible leading zeroes. For example, 123456, 000000, 099999, 999999 are correct ticket numbers, but 1234567, 12345, 9 are not. Every day tickets are numbered from zero. The first passenger gets ticket number 000000, the second one — ticket number 000001, the third one — number 000002 and so on every day. Assume that each day the number of passengers doesn't exceed 10^6.

Unluckiness of the ticket is equal to absolute difference between the sum of the first three digits and the sum of the last three. For example, unluckiness of the ticket number 345123 is equal to $|(3+4+5)-(1+2+3)| = 6$, or unluckiness of the ticket number 238526 is equal to $|(2+3+8)-(5+2+6)| = 0$.

One passenger is luckier than other if unluckiness of first passenger's ticket is strictly less than unluckiness of the second one's ticket.

For each of n days for given Monocarp's ticket's number t_i calculate the number of passengers who received their tickets before him during this day and are luckier than Monocarp.

Examine examples for the further understanding of the statement.

Input

The first line contains one integer n $(1 \le n \le 2 \cdot 10^5)$ — the number of days during which Monocarp used public transport.

Each of the next n lines contains one six digit integer t_i $(0 \le t_i < 10^6$, t_i can have leading zeroes) — the ticket Monocarp received during the corresponding day.

Output

Print n lines: one integer per line — the number of passengers who received their tickets before Monocarp during the corresponding day and are luckier than Monocarp.

Example

standard input	standard output
5 001000 000000 999000 453234 654331	1 0 998999 121496 470362

Note

During the first day the only one passenger who got ticket before Monocarp was luckier. This passenger got ticket number 000000.

During the second day Monocarp was the first one, so there was nobody before him.

During the third day all passengers except one who got tickets before Monocarp were more luckier than him. The one whose unluckiness was equal to Monocarp's unluckiness got ticket number 000999.

Problem G. Tree Reconstruction

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

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

standard input	standard output

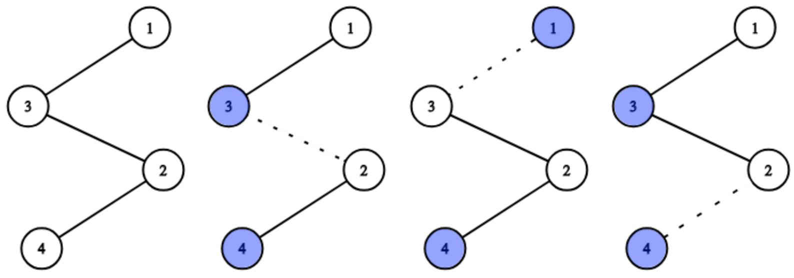
4
3 4
1 4
3 4
	YES
	1 3
	3 2
	2 4

3
1 3
1 3
	NO

3
1 2
2 3
	NO

Note

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



Problem H. Theater Square

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

The Theater Square can be represented as a rectangle having height n and length m, divided into square 1×1 cells. Let's denote the cell located at the intersection of i-th row and j-th column as (i, j). The rows are numbered from top to bottom, the columns — from left to right.

There is a rectangular fountain inside the Teather Square. The cell in its left upper corner is (x_1, y_1), the cell in its right lower corner is (x_2, y_2).

The Theater Square soon will be paved with tiles having height 1 and length 2. Every cell **(except cells inside the fountain)** should be paved, and no cell should be covered by more than one tile. All tiles will be laid out horizontally, so the cells covered by each tile are in the same row. To pave the whole Theater Square it might be necessary to break some tiles. After breaking a tile, two new tiles of size 1×1 are formed (which cannot be broken further). You may consider that the mayor, who ordered the paving of the Theater Square, has infinite number of tiles 1×2.

Since broken tiles are not beautiful, among all possible ways to pave the Theater Square the mayor wants to choose a way such that **the number of tiles to be broken into two lesser tiles** is minimum possible. Pay attention that **tiles should be laid horizontally**, no tile can cover cells in different rows.

Help the mayor! Tell him the minimum possible number of tiles to be broken.

Input

The first line contains two integers n and m ($1 \le n, m \le 2 \cdot 10^5$) — the height and the length of the Theater Square, respectively.

The second line contains four numbers x_1, y_1, x_2, y_2 ($1 \le x_1 \le x_2 \le n, 1 \le y_1 \le y_2 \le m$) — the coordinates of left upper corner and right lower corner of the fountain.

Output

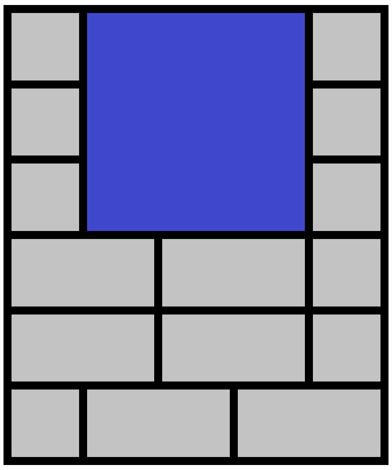
Print one number — minimum possible number of tiles mayor has to break in order to pave the whole Theater Square.

Examples

standard input	standard output
6 5 1 2 3 4	5
6 1 3 1 4 1	2
1 12 1 3 1 8	0

Note

One of the optimal ways to pave the Theater Square in the first example:



5 tiles are to be broken.

Problem I. Heist

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

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

standard input	standard output
4 10 13 12 8	2
5 7 5 6 4 8	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.

Problem J. Buying a TV Set

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

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

standard input	standard output
`17 15 5 3`	`3`
`14 16 7 22`	`0`
`4 2 6 4`	`1`
`1000000000000000000 1000000000000000000 999999866000004473 999999822000007597`	`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: $(3, 2)$.

Problem K. Medians and Partition

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Let median of some array be the number which would stand in the middle of this array if it was sorted beforehand. If the array has even length let median be smallest of of two middle elements. For example, median of the array $[10, 3, 2, 3, 2]$ is 3 (i.e. $[2, 2, \underline{3}, 3, 10]$). Median of the array $[1, 5, 8, 1]$ is 1 (i.e. $[1, \underline{1}, 5, 8]$).

Let array be m-*good* if its median is greater or equal than m.

Let the partition of array $[a_1, a_2, \ldots, a_n]$ be a set of subarrays $\{b_1, b_2, \ldots, b_k\}$ such that $b_1 = [a_1, a_2, \ldots, a_{i_1}]$, $b_2 = [a_{i_1+1}, a_{i_1+2}, \ldots, a_{i_2}]$, $\ldots$, $b_k = [a_{i_{k-1}+1}, a_{i_{k-1}+2}, \ldots, a_n]$. For example, array $[10, 3, 2, 3, 2]$ can be partitioned as follows: $\{[10, 3, 2, 3, 2]\}$ or $\{[10], [3], [2], [3], [2]\}$, or $\{[10], [3, 2, 3, 2]\}$, or $\{[10, 3], [2], [3, 2]\}$ and so on.

You are given array a of length n and integer m. Find the partition of a into maximum number of subarrays such that each subarray is m-*good*.

Input

The first line contains two integers n and m ($1 \le n \le 5\,000$, $1 \le m \le 5\,000$) — length of array a and constant m.

The second line contains n integers a_1, a_2, $\ldots$, a_n ($1 \le a_i \le 5\,000$)— array a.

Output

If there is no valid partition of array a into m-*good* subarrays, print 0. Otherwise print maximum number of subarrays in partition of array a such that each subarray is m-*good*.

Examples

standard input	standard output

`5 2`
`10 3 2 3 2`
	`5`

`5 3`
`10 3 2 3 2`
	`1`

`5 4`
`10 3 2 3 2`
	`0`

Note

In the first example array can be partitioned into 5 subarrays: $\{[10], [3], [2], [3], [2]\}$. Medians of each part greater of equal than 2.

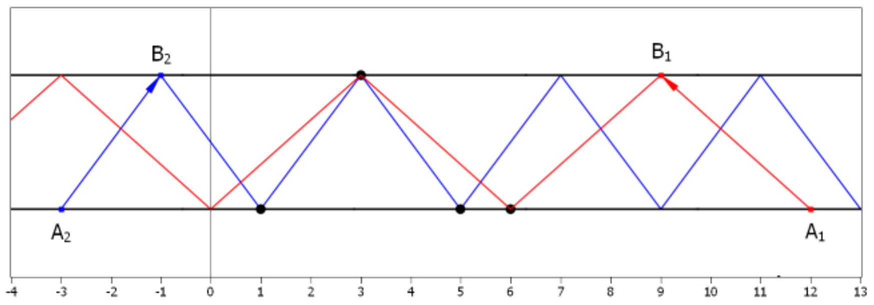
In the second example we can't partition array into several subarrays since medians of $[2]$, $[3, 2]$, $[2, 3, 2]$ and $[3, 2, 3, 2]$ are less than 3.

Problem L. Ray in the tube

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

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

standard input	standard output

3 1
1 5 6
1 3
3
	3

Note

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