Codeforces Round #523 (Div. 2)

A. Coins

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

You have unlimited number of coins with values $1, 2, \ldots, n$. You want to select some set of coins having the total value of S.

It is allowed to have multiple coins with the same value in the set. What is the minimum number of coins required to get sum S?

Input
The only line of the input contains two integers n and S ($1 \le n \le 100\,000$, $1 \le S \le 10^9$)

Output
Print exactly one integer — the minimum number of coins required to obtain sum S.

Examples

input
5 11
output
3

input
6 16
output
3

Note
In the first example, some of the possible ways to get sum 11 with 3 coins are:

- $(3, 4, 4)$
- $(2, 4, 5)$
- $(1, 5, 5)$
- $(3, 3, 5)$

It is impossible to get sum 11 with less than 3 coins.

In the second example, some of the possible ways to get sum 16 with 3 coins are:

- $(5, 5, 6)$
- $(4, 6, 6)$

It is impossible to get sum 16 with less than 3 coins.

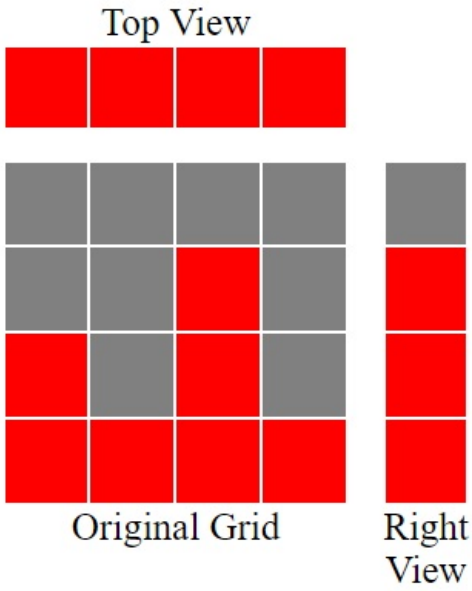
B. Views Matter

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

You came to the exhibition and one exhibit has drawn your attention. It consists of n stacks of blocks, where the i-th stack consists of a_i blocks resting on the surface.

The height of the exhibit is equal to m. Consequently, the number of blocks in each stack is less than or equal to m.

There is a camera on the ceiling that sees the top view of the blocks and a camera on the right wall that sees the side view of the blocks.



Find the maximum number of blocks you can remove such that the views for both the cameras would not change.

Note, that while originally all blocks are stacked on the floor, it is not required for them to stay connected to the floor after some blocks are removed. There is **no gravity** in the whole exhibition, so no block would fall down, even if the block underneath is removed. It is not allowed to move blocks by hand either.

Input

The first line contains two integers n and m ($1 \le n \le 100\,000$, $1 \le m \le 10^9$) — the number of stacks and the height of the exhibit.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le m$) — the number of blocks in each stack from left to right.

Output

Print exactly one integer — the maximum number of blocks that can be removed.

Examples

input
5 6 3 3 3 3 3
output
10

input
3 5 1 2 4
output
3

input
5 5 2 3 1 4 4
output
9

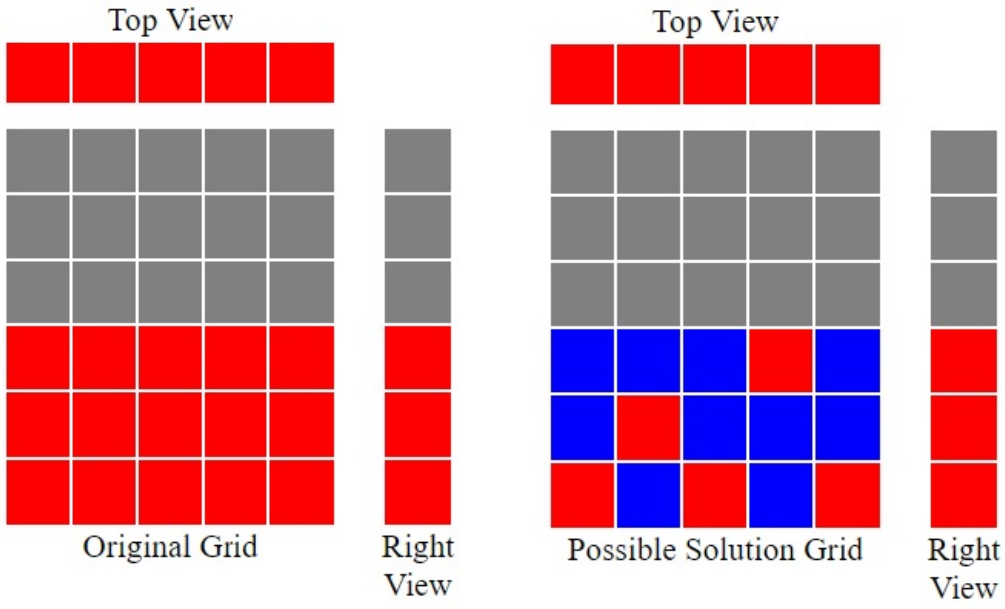
input
1 1000 548
output
0

input
3 3 3 1 1
output
1

Note

The following pictures illustrate the first example and its possible solution.

Blue cells indicate removed blocks. There are 10 blue cells, so the answer is 10.



C. Multiplicity

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given an integer array $a_1, a_2, \ldots, a_n$.

The array b is called to be a *subsequence* of a if it is possible to remove some elements from a to get b.

Array $b_1, b_2, \ldots, b_k$ is called to be *good* if it is not empty and for every i ($1 \le i \le k$) b_i is divisible by i.

Find the number of good subsequences in a modulo $10^9 + 7$.

Two subsequences are considered different if index sets of numbers included in them are different. That is, the values of the elements do not matter in the comparison of subsequences. In particular, the array a has exactly $2^n - 1$ different subsequences (excluding an empty subsequence).

Input
The first line contains an integer n ($1 \le n \le 100\,000$) — the length of the array a.

The next line contains integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^6$).

Output
Print exactly one integer — the number of good subsequences taken modulo $10^9 + 7$.

Examples

input
2 1 2
output
3

input
5 2 2 1 22 14
output
13

Note
In the first example, all three non-empty possible subsequences are good: $\{1\}$, $\{1, 2\}$, $\{2\}$

In the second example, the possible good subsequences are: $\{2\}$, $\{2, 2\}$, $\{2, 22\}$, $\{2, 14\}$, $\{2\}$, $\{2, 22\}$, $\{2, 14\}$, $\{1\}$, $\{1, 22\}$, $\{1, 14\}$, $\{22\}$, $\{22, 14\}$, $\{14\}$.

Note, that some subsequences are listed more than once, since they occur in the original array multiple times.

D. TV Shows

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

There are n TV shows you want to watch. Suppose the whole time is split into equal parts called "minutes". The i-th of the shows is going from l_i-th to r_i-th minute, both ends inclusive.

You need a TV to watch a TV show and you can't watch two TV shows which air at the same time on the same TV, so it is possible you will need multiple TVs in some minutes. For example, if segments $[l_i, r_i]$ and $[l_j, r_j]$ intersect, then shows i and j can't be watched simultaneously on one TV.

Once you start watching a show on some TV it is not possible to "move" it to another TV (since it would be too distracting), or to watch another show on the same TV until this show ends.

There is a TV Rental shop near you. It rents a TV for x rupees, and charges y ($y < x$) rupees for every extra minute you keep the TV. So in order to rent a TV for minutes $[a; b]$ you will need to pay $x + y \cdot (b - a)$.

You can assume, that taking and returning of the TV doesn't take any time and doesn't distract from watching other TV shows. Find the minimum possible cost to view all shows. Since this value could be too large, print it modulo $10^9 + 7$.

Input

The first line contains integers n, x and y ($1 \le n \le 10^5$, $1 \le y < x \le 10^9$) — the number of TV shows, the cost to rent a TV for the first minute and the cost to rent a TV for every subsequent minute.

Each of the next n lines contains two integers l_i and r_i ($1 \le l_i \le r_i \le 10^9$) denoting the start and the end minute of the i-th TV show.

Output

Print exactly one integer — the minimum cost to view all the shows taken modulo $10^9 + 7$.

Examples

input
5 4 3 1 2 4 10 2 4 10 11 5 9
output
60

input
6 3 2 8 20 6 22 4 15 20 28 17 25 20 27
output
142

input
2 1000000000 2 1 2 2 3
output
999999997

Note

In the first example, the optimal strategy would be to rent 3 TVs to watch:

- Show $[1, 2]$ on the first TV,
- Show $[4, 10]$ on the second TV,
- Shows $[2, 4]$, $[5, 9]$, $[10, 11]$ on the third TV.

This way the cost for the first TV is $4 + 3 \cdot (2 - 1) = 7$, for the second is $4 + 3 \cdot (10 - 4) = 22$ and for the third is $4 + 3 \cdot (11 - 2) = 31$, which gives 60 int total.

In the second example, it is optimal watch each show on a new TV.

In third example, it is optimal to watch both shows on a new TV. Note that the answer is to be printed modulo $10^9 + 7$.

E. Politics

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

There are n cities in the country.

Two candidates are fighting for the post of the President. The elections are set in the future, and both candidates have already planned how they are going to connect the cities with roads. Both plans will connect all cities using $n - 1$ roads only. That is, each plan can be viewed as a tree. Both of the candidates had also specified their choice of the capital among n cities (x for the first candidate and y for the second candidate), which may or may not be same.

Each city has a potential of building a port (one city can have at most one port). Building a port in i-th city brings a_i amount of money. However, each candidate has his specific demands. The demands are of the form:

- k x, which means that the candidate wants to build exactly x ports in the subtree of the k-th city of his tree (the tree is rooted at the capital of his choice).

Find out the maximum revenue that can be gained while fulfilling all demands of both candidates, or print `-1` if it is not possible to do.

It is additionally guaranteed, that each candidate has specified the port demands for the capital of his choice.

Input

The first line contains integers n, x and y ($1 \le n \le 500$, $1 \le x, y \le n$) — the number of cities, the capital of the first candidate and the capital of the second candidate respectively.

Next line contains integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 100\,000$) — the revenue gained if the port is constructed in the corresponding city.

Each of the next $n - 1$ lines contains integers u_i and v_i ($1 \le u_i, v_i \le n$, $u_i \ne v_i$), denoting edges between cities in the tree of the first candidate.

Each of the next $n - 1$ lines contains integers u'_i and v'_i ($1 \le u'_i, v'_i \le n$, $u'_i \ne v'_i$), denoting edges between cities in the tree of the second candidate.

Next line contains an integer q_1 ($1 \le q_1 \le n$), denoting the number of demands of the first candidate.

Each of the next q_1 lines contains two integers k and x ($1 \le k \le n$, $1 \le x \le n$) — the city number and the number of ports in its subtree.

Next line contains an integer q_2 ($1 \le q_2 \le n$), denoting the number of demands of the second candidate.

Each of the next q_2 lines contain two integers k and x ($1 \le k \le n$, $1 \le x \le n$) — the city number and the number of ports in its subtree.

It is guaranteed, that given edges correspond to valid trees, each candidate has given demand about each city at most once and that each candidate has specified the port demands for the capital of his choice. That is, the city x is always given in demands of the first candidate and city y is always given in the demands of the second candidate.

Output

Print exactly one integer — the maximum possible revenue that can be gained, while satisfying demands of both candidates, or `-1` if it is not possible to satisfy all of the demands.

Examples

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

input
5 1 1 3 99 99 100 2 1 2 1 3

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

output

```
198
```

input

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

output

```
-1
```

Note

In the first example, it is optimal to build ports in cities 2, 3 and 4, which fulfills all demands of both candidates and gives revenue equal to $2 + 3 + 4 = 9$.

In the second example, it is optimal to build ports in cities 2 and 3, which fulfills all demands of both candidates and gives revenue equal to $99 + 99 = 198$.

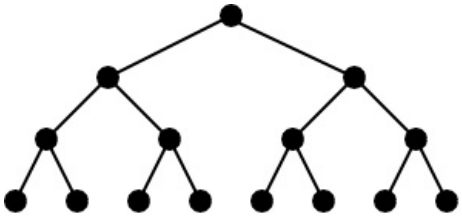
In the third example, it is not possible to build ports in such way, that all demands of both candidates are specified, hence the answer is `-1`.

F. Lost Root

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

The graph is called *tree* if it is connected and has no cycles. Suppose the tree is rooted at some vertex. Then tree is called to be perfect k-ary tree if each vertex is either a leaf (has no children) or has exactly k children. Also, in perfect k-ary tree all leafs must have same depth.

For example, the picture below illustrates perfect binary tree with 15 vertices:



There is a perfect k-ary tree with n nodes. The nodes are labeled with distinct integers from 1 to n, however you don't know how nodes are labelled. Still, you want to find the label of the root of the tree.

You are allowed to make at most $60 \cdot n$ queries of the following type:

- "`? a b c`", the query returns "Yes" if node with label b lies on the path from a to c and "No" otherwise.

Both a and c are considered to be lying on the path from a to c.

When you are ready to report the root of the tree, print

- "`! s`", where s is the label of the root of the tree.

It is possible to report the root only once and this query is not counted towards limit of $60 \cdot n$ queries.

Interaction

The first line of the standard input stream contains two integers n and k ($3 \le n \le 1500$, $2 \le k < n$) — the number of nodes in the

tree and the value of k.

It is guaranteed that n is such that the tree forms a perfect k-ary tree.

You can ask at most $60 \cdot n$ queries. To ask a query, print a line of form "? a b c", where $1 \le a, b, c \le n$. After that you should read a single line containing "Yes" or "No" depending on the answer of the query.

The tree is fixed for each test and it doesn't depend on your queries.

When you are ready to print the answer, print a line of the form "! s", where s is the label of the root vertex and then terminate your program.

After printing each query do not forget to print end of line and flush the output. Otherwise you may get `Idleness limit` exceeded. To do this, use:

- `fflush(stdout)` or `cout.flush()` in C++;
- `System.out.flush()` in Java;
- `flush(output)` in Pascal;
- `stdout.flush()` in Python;
- See documentation for other languages.

In case your program will make more than $60 \cdot n$ queries, but in other aspects would follow the interaction protocol and terminate coorectly, it will get verdict «`Wrong Answer`».

Hacks

To hack the solution use the following test format:

The first line should contain integers n and k ($3 \le n \le 1500$, $2 \le k \le 1500$) — the number of vertices and the k parameter of the tree.

Of course, the value of n must correspond to the size of the valid k-ary tree of some depth.

The second line should contain $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$) — the labels of the tree in the natural order, all labels must be distinct.

Let's call the following ordering of the tree vertices to be natural: first the root of the tree goes, then go all vertices on depth of one edge from root, ordered from left to right, then go all vertices on depth of two edges from root, ordered from left to right, and so on until the maximum depth.

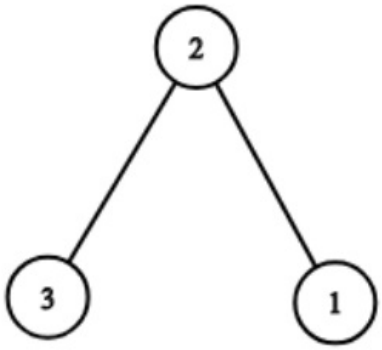
This way, the a_1 is the answer for the hack.

Example

input
3 2 No Yes
output
? 1 3 2 ? 1 2 3 ! 2

Note

The tree in the example is as follows:



The input and output for example illustrate possible interaction on that test (empty lines are inserted only for clarity).

The hack corresponding to the example would look like:

3 2
2 3 1