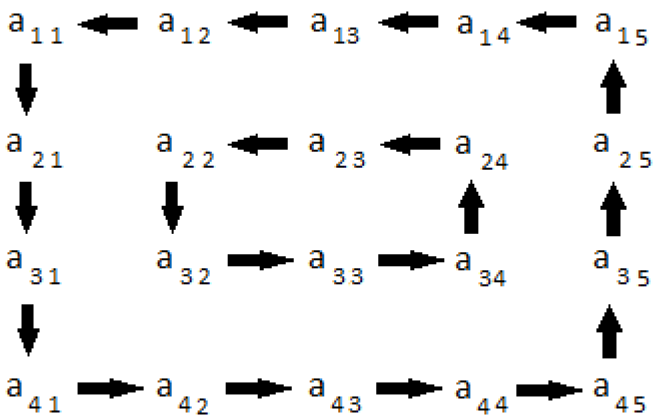
Name:

Problem #1

You are given a 2D matrix of dimension $m \times n$ and a positive integer r. You have to rotate the matrix r times and print the resultant matrix. Rotation should be in anti-clockwise direction. Rotation of a 4×5 matrix is represented by the following figure. Note that in one rotation, you have to shift elements by one step only.



Matrix Rotation

It is guaranteed that the minimum of *m* and *n* will be even.

As an example rotate the Start matrix by 2:

```
Start         First            Second
 1 2 3 4        2   3   4   5     3   4   5   6
12 1 2 5  ->  1   2   3   6 ->  2   3   4   7
11 4 3 6      12   1   4   7      1   2   1   8
10 9 8 7      11  10   9   8     12  11  10   9
```

Function Description

Complete the *matrixRotation* function. It should print the resultant 2D integer array and return nothing.

matrixRotation has the following parameter(s):

- *matrix*: a 2D array of integers
- *r*: an integer that represents the rotation factor

Input Format

The first line contains three space separated integers, m, n and r, the number of rows and columns in ***matrix***, and the required rotation.
The next lines ***m*** contain ***n*** space-separated integers representing the elements of a row of ***matrix***

.

Constraints

$$2 \leq \mathrm{m}, \mathrm{n} \leq 300$$
$$1 \leq r \leq 10^9$$
$$min(m, n)\%2 = 0$$
$$1 \leq a_{ij} \leq 10^8 \text{ where } i \in [1 \ldots m] \text{ and } j \in [1 \ldots n]$$

Output Format

Print each row of the rotated matrix as space-separated integers on separate lines.

Sample Input

Sample Input #01

```
4 4 2
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
```

Sample Output #01

```
3 4 8 12
2 11 10 16
1 7 6 15
5 9 13 14
```

Explanation #01

The matrix is rotated through two rotations.

```
 1  2  3  4      2  3  4  8      3  4  8 12
 5  6  7  8      1  7 11 12      2 11 10 16
 9 10 11 12  ->  5  6 10 16  ->  1  7  6 15
13 14 15 16      9 13 14 15      5  9 13 14
```

Sample Input #02

```
5 4 7
1 2 3 4
7 8 9 10
13 14 15 16
19 20 21 22
25 26 27 28
```

Sample Output #02

```
28 27 26 25
22 9 15 19
16 8 21 13
10 14 20 7
4 3 2 1
```

Explanation 02

The various states through 7 rotations:

```
1  2  3  4      2  3  4 10    3  4 10 16    4 10 16 22
7  8  9 10      1  9 15 16    2 15 21 22    3 21 20 28
13 14 15 16 ->  7  8 21 22 -> 1  9 20 28 -> 2 15 14 27 ->
19 20 21 22    13 14 20 28    7  8 14 27    1  9  8 26
25 26 27 28    19 25 26 27    13 19 25 26   7 13 19 25

10 16 22 28    16 22 28 27    22 28 27 26    28 27 26 25
 4 20 14 27    10 14  8 26    16  8  9 25    22  9 15 19
 3 21  8 26 ->  4 20  9 25 -> 10 14 15 19 -> 16  8 21 13
 2 15  9 25     3 21 15 19     4 20 21 13    10 14 20  7
 1  7 13 19     2  1  7 13     3  2  1  7     4  3  2  1
```

Sample Input #03

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

Sample Output #03

```
1 1
1 1
```

Explanation #03

All of the elements are the same, so any rotation will repeat the same matrix.

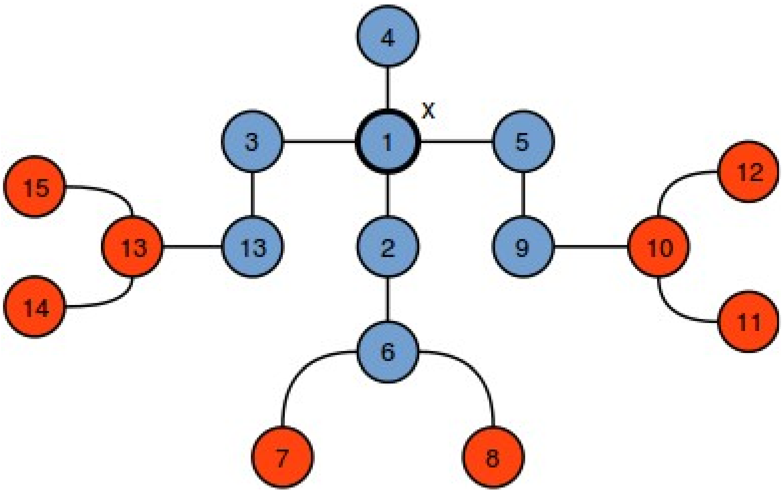
Problem #2

Jenny loves experimenting with trees. Her favorite tree has nodes ***n*** connected by ***n-1*** edges, and each edge is **1** unit in length. She wants to cut a *subtree* (i.e., a connected part of the original tree) of radius

from this tree by performing the following two steps:

1. Choose a node, **x** , from the tree.
2. . Cut a subtree consisting of *all* nodes which are *not further* than ***r*** units from node ***x***

For example, the blue nodes in the diagram below depict a subtree centered at ***x* = *1*** that has radius ***r* =*2***:



Given ***n, r*** and the definition of Jenny's tree, find and print the number of *different* subtrees she can cut out. Two subtrees are considered to be different if they are not isomorphic.

Input Format

The first line contains two space-separated integers denoting the respective values of ***n*** and ***r***.
Each of the next ***n* – *1*** subsequent lines contains two space-separated integers, ***x*** and ***y***, describing a bidirectional edge in Jenny's tree having length **1**.

Constraints

- $1 \leq n \leq 3000$
- $0 \leq r \leq 3000$
- $1 \leq x, y \leq n$

Subtasks

For ***50%*** of the max score:

$1 \leq n \leq 500$

$0 \leq r \leq 500$

Output Format

Print the total number of different possible subtrees.

Sample Input 0

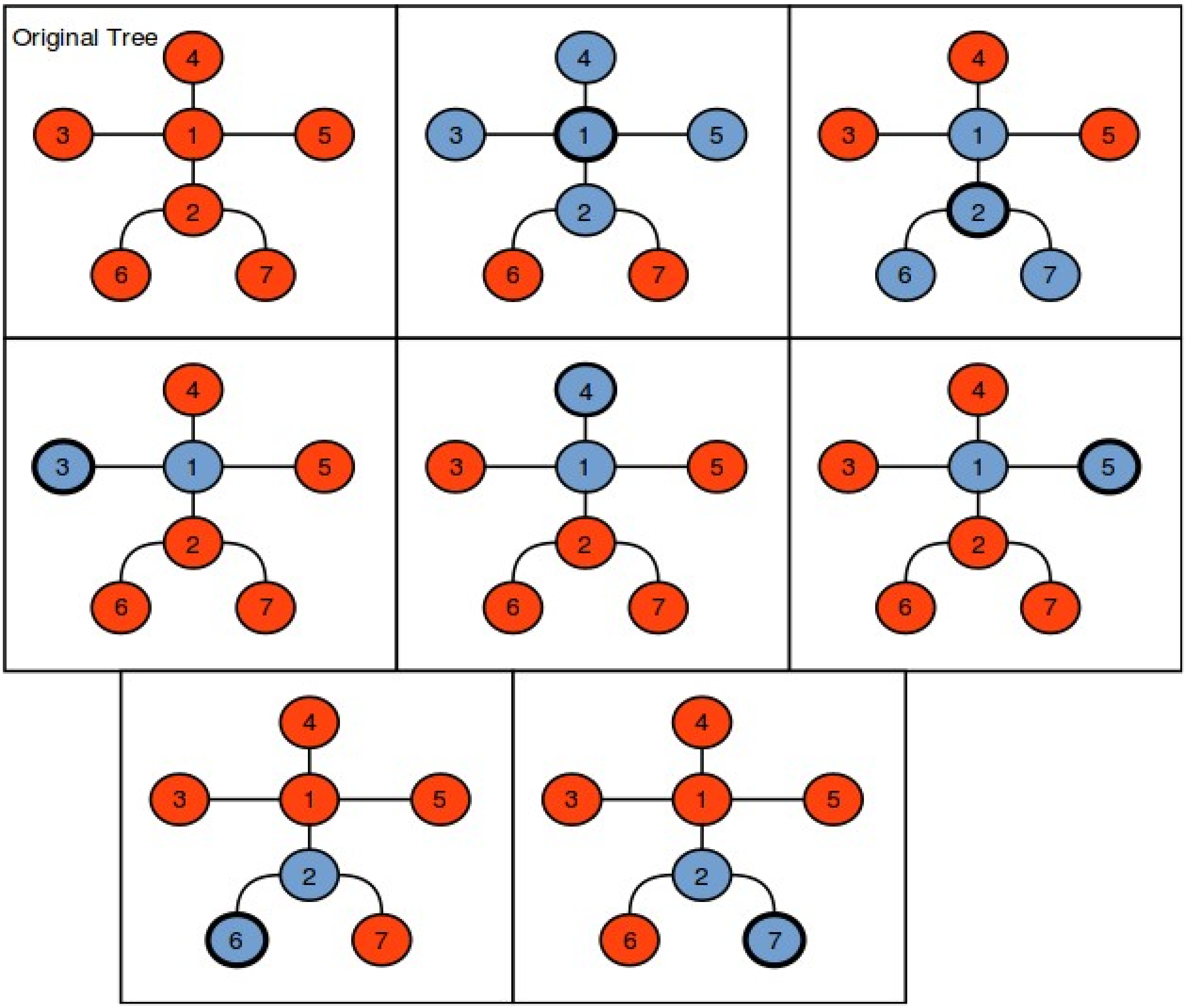
```
7 1
1 2
1 3
1 4
1 5
2 6
2 7
```

Sample Output 0

```
3
```

Explanation 0

In the diagram below, blue nodes denote the possible subtrees:



The last ***5*** subtrees are considered to be the same (i.e., they all consist of two nodes connected by one edge), so we print ***3*** as our answer.

Sample Input 1

```
7 3
1 2
2 3
3 4
```

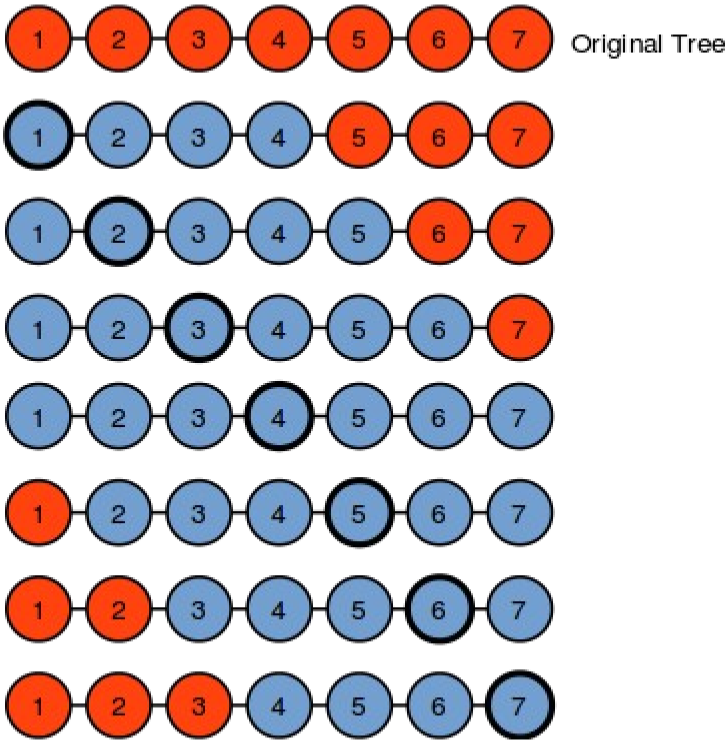
4 5
5 6
6 7

Sample Output 1

4

Explanation 1

In the diagram below, blue nodes denote the possible subtrees:



Here, we have four possible different subtrees.

Problem #3

For the purposes of this challenge, we define a binary tree to be a binary search tree with the following ordering requirements:

- The ***data*** value of every node in a node's left subtree is *less than* the data value of that node.
- The ***data*** value of every node in a node's right subtree is *greater than* the data value of that node.

Given the root node of a binary tree, can you determine if it's also a binary search tree?

Complete the function in your editor below, which has ***1*** parameter: a pointer to the root of a binary tree. It must return a *boolean* denoting whether or not the binary tree is a binary search tree. You may have to write one or more helper functions to complete this challenge.

Input Format

You are not responsible for reading any input from stdin. Hidden code stubs will assemble a binary tree and pass its root node to your function as an argument.

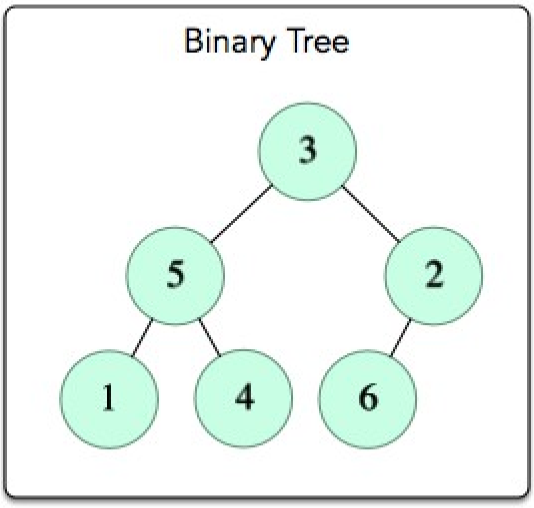
Constraints

$0 \leq data \leq 10^4$

Output Format

You are not responsible for printing any output to stdout. Your function must return *true* if the tree is a binary search tree; otherwise, it must return *false*. Hidden code stubs will print this result as a *Yes* or *No* answer on a new line.

Sample Input



Sample Output

```
No
```

Problem #4

Starting with a 1-indexed array of zeros and a list of operations, for each operation add a value to each of the array element between two given indices, inclusive. Once all operations have been performed, return the maximum value in your array.

For example, the length of your array of zeros $n = 10$. Your list of queries is as follows:

```
a b k
1 5 3
4 8 7
6 9 1
```

Add the values of k between the indices a and b inclusive:

```
index->  1 2 3  4  5 6 7 8 9 10
        [0,0,0, 0, 0,0,0,0,0, 0]
        [3,3,3, 3, 3,0,0,0,0, 0]
        [3,3,3,10,10,7,7,7,0, 0]
        [3,3,3,10,10,8,8,8,1, 0]
```

The largest value is ***10*** after all operations are performed.

Function Description

Complete the function *arrayManipulation* in the editor below. It must return an integer, the maximum value in the resulting array.

arrayManipulation has the following parameters:

- *n* - the number of elements in your array
- *queries* - a two dimensional array of queries where each *queries[i]* contains three integers, *a*, *b*, and *k*.

Input Format

The first line contains two space-separated integers n and m, the size of the array and the number of operations.
Each of the next lines contains three space-separated integers a, b and k, the left index, right index and summand.

Constraints

$3 \leq n \leq 10^7$
$1 \leq m \leq 2 * 10^5$
$1 \leq a \leq b \leq n$
$0 \leq k \leq 10^9$

Output Format

Return the integer maximum value in the finished array.

Sample Input

```
5 3
1 2 100
2 5 100
```

```
3 4 100
```

Sample Output

```
200
```

Explanation

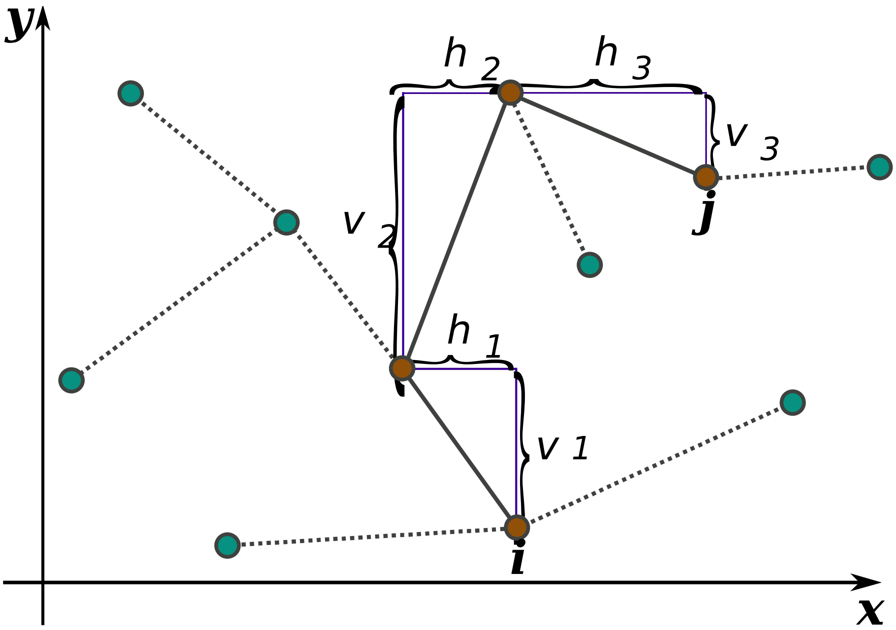
After the first update list will be `100 100 0 0 0`.
After the second update list will be `100 200 100 100 100`.
After the third update list will be `100 200 200 200 100`.
The required answer will be ***200***.

Problem #5

Burger Town is a city that consists of ***N*** special junctions and ***N-1*** pathways. There is exactly one shortest path between each pair of junctions. Junction ***i*** is located at ***(xi, yi)*** and the distance between two junctions ***i, j*** is defined by the Taxicab geometry.
Tim has recently afforded a taxicab to work as a taxicab driver. His vehicle was very cheap, but has a very big flaw. It can only drive ***H*** units horizontally and ***V*** units vertically before refueling.

If a customer wants to be brought from a junction ***i*** to another junction ***j***, then this car is only capable of driving the route, if the sum of horizontal distances and the sum of vertical distances on this path are less than or equal to ***H*** and ***V*** respectively.

Also, there is a unique path between any two junctions.



Now he has thoughts about returning the vehicle back to the seller. But he first wants to know, if it's even worth it. That's why he wants to know the number of unordered pairs ***(i, j)*** such that it is not possible to drive a customer from junction ***i*** to junction ***j***.

Input Format

On the first line you will be given ***N***, ***H*** and ***V*** separated by a single space.
Each of the next ***N*** lines contains two space separated integers ***xi, yi*** denoting the location of junction ***i***. Each of the next ***N-1*** lines contains two space separated integers describing a path existing between ***ui, vi*** i.e., there is a path between ***ui*** and ***vi***.

Output Format

Output the number of unordered pairs *(i,j)* such that it is not possible to drive from *i* to *j*.

Constraints

$2 \leq N \leq 10^5$

$0 \leq H, V \leq 10^{14}$

$0 \leq x_i, y_i \leq 10^9$

Sample Input

```
3 2 1
0 0
1 1
2 0
1 2
2 3
```

Sample Output

```
1
```

Explanation

The car is only capable of driving ***H*= 2** units horizontally and ***V*=1** unit vertically. The horizontal distance between junction 1 and 3(via 2) is equal to 2(0 → 1 → 2), which fits under the horizontal limit of the car. The vertical distance between 1 and 3 is also equal to 2(0→ 1→ 0), but this is not possible for this car since ***2 > V***.

Problem #6

Given an array, we define its *value* to be the value obtained by following these instructions:

- Write down all pairs of numbers from this array.
- Compute the product of each pair.
- Find the sum of all the products.

For example, for a given array, for a given array ***[7, 2, -1, 2]***,

Pairs	**(7, 2), (7, -1), (7, 2), (2, -1), (2, 2), (-1, 2)**
Products of the pairs	14, -7, 14, -2, 4, -2
Sum of the products	14 + (-7) + 14 + (-2) + 4 + (-2) = 21

Note that **(7, 2)** is listed twice, one for each occurrence of **2**.

Given an array of integers, find the largest *value* of any of its nonempty subarrays.

Note: A subarray is a contiguous subsequence of the array.

Complete the function `largestValue` which takes an array and returns an integer denoting the largest *value* of any of the array's nonempty subarrays.

Input Format

The first line contains a single integer ***n***, denoting the number of integers in array ***A***.
The second line contains ***n*** space-separated integers ***Ai*** denoting the elements of array ***A***.

Constraints

$$3 \leq n \leq 5 \cdot 10^5$$
$$-10^3 \leq A_i \leq 10^3$$

Subtasks

- ***n <= 5000*** for 20% of the points.
- ***n<= 2* 10^5*** for 70% of the points.

Output Format

Print a single line containing a single integer denoting the largest *value* of any of the array's nonempty subarrays.

Sample Input 0

```
6
-3 7 -2 3 5 -2
```

Sample Output 0

```
41
```

Explanation 0

In this case, we have **A = [-3,7,-2,3,5,-2]**. The largest-valued subarray turns out to be **[7,-2,3,5]** with value **(7*-2) + (7*3) + (7*5) + (-2*3) + (-2*5) + (3*5) = 41**.

Sample Input 1

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

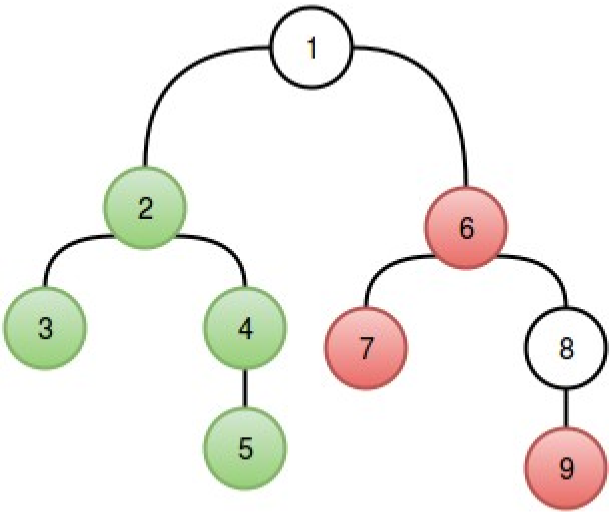
Sample Output 1

```
200
```

Problem #7

Treeland is a country with ***n*** cities and ***n-1*** roads. There is exactly *one* path between any two cities. The ruler of Treeland wants to implement a self-driving bus system and asks tree-loving Alex to plan the bus routes. Alex decides that each route must contain a subset of *connected* cities; a subset of cities is *connected* if the following two conditions are true:

1. There is a path between every pair of cities which belongs to the subset.
2. Every city in the path must belong to the subset.



In the figure above, **{2, 3, 4, 5}** is a *connected* subset, but **{6, 7, 9}** is not (for the second condition to be true, **8** would need to be part of the subset).

Each self-driving bus will operate within a *connected segment* of Treeland. A connected segment $[L, R]$ where $1 \leq L \leq R \leq n$ is defined by the connected subset of cities $S = \{x / x \in Z \wedge L \leq x \leq R\}$.

In the figure above, **[2, 5]** is a connected segment that represents the subset **{2, 3, 4, 5}**. Note that a single city can be a segment too.

Help Alex to find number of connected segments in Treeland.

Input Format

The first line contains a single positive integer, ***n***. The ***n - 1*** subsequent lines each contain two positive space-separated integers, ***ai*** and ***bi***, describe an edge connecting two nodes in tree ***T***.

Constraints

- $1 \leq n \leq 2 x 10^5$
- $1 \leq a_i, b_i \leq n$

Subtasks

- For ***25%*** score: $1 \leq n \leq 2 x 10^3$
- For ***50%*** score: $1 \leq n \leq 10^4$

Output Format

Print a single integer: the number of segments ***[L, R]***, which are connected in tree ***T***.

Sample Input

```
3
1 3
3 2
```

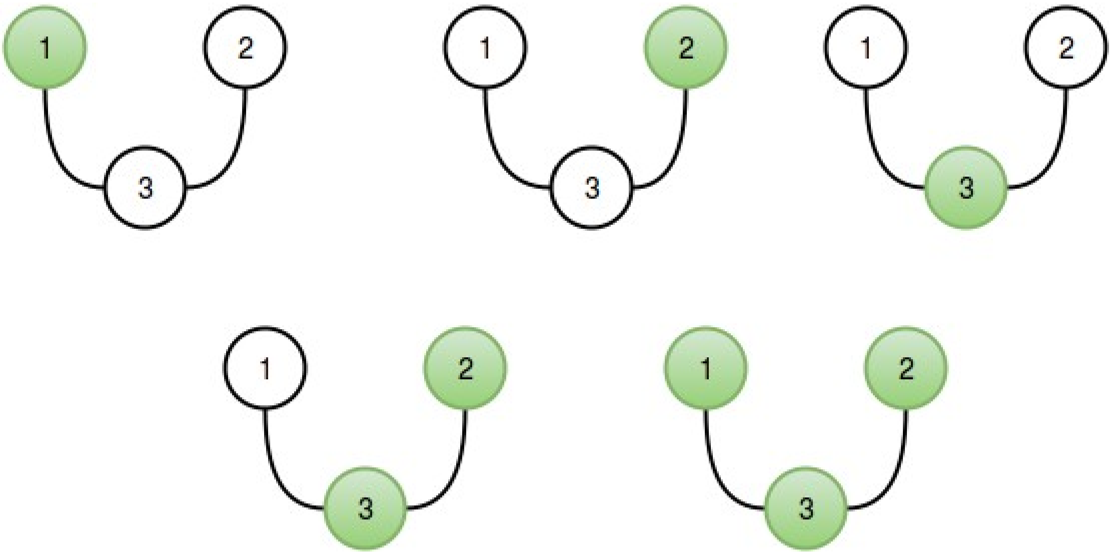
Sample Output

```
5
```

Explanation

The connected *segments* for our test case are: **[1, 1], [2, 2], [3, 3], [2, 3]** and **[1, 3]**. These *segments* can be represented by the respective subsets: **{1}**, **{2}**, **{3}**, **{2, 3}**, and **{1, 2, 3}**

.



Note: **[1,2]** is not a connected segment. It represents the subset **{1,2}** and the path between **1** and **2** goes through **3** which is not a member of the subset.

Problem #8

Given two numbers ***N*** and ***M***. ***N*** indicates the number of elements in the array A[](1 - *indexed*) and ***M*** indicates number of queries. You need to perform two types of queries on the array A[].
You are given **M** queries. Queries can be of two types, type **1** and type **2**.

- Type 1 queries are represented as `1 i j` : Modify the given array by removing elements from ***i*** to ***j*** and adding them to the front.
- Type 2 queries are represented as `2 i j` : Modify the given array by removing elements from ***i*** to ***j*** and adding them to the back.

Your task is to simply print $|A[1]-A[N]|$ of the resulting array after the execution of ***M*** queries followed by the resulting array.

Note While adding at back or front the order of elements is preserved.

Input Format

First line consists of two space-separated integers, ***N*** and ***M***.
Second line contains ***N*** integers, which represent the elements of the array.
M queries follow. Each line contains a query of either *type 1* or *type 2* in the form *i j*

Constraints

$1 \le N, M \le 10^5$
$1 \le A[i] \le 10^9$
$1 \le i \le j \le N$

Output Format

Print the absolute value i.e. *abs*(A[1] – A[*N*]) in the first line.
Print elements of the resulting array in the second line. Each element should be seperated by a single space.

Sample Input

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

Sample Output

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

Explanation

Given array is **{1, 2, 3, 4, 5, 6, 7, 8}**.
After execution of query **1 2 4**, the array becomes **{2, 3, 4, 1, 5, 6, 7, 8}**.
After execution of query **2 3 5**, the array becomes **{2, 3, 6, 7, 8, 4, 1, 5}**.
After execution of query **1 4 7**, the array becomes **{7, 8, 4, 1, 2, 3, 6, 5}**.

After execution of query **2 1 4**, the array becomes **{2, 3, 6, 5, 7, 8, 4, 1}**.
Now $|A[1]-A[N]|$ is $|(2-1)|$ i.e. **1** and the array is **23657841**