NOI 2025 Preliminary Contest

22 February 2025

Tasks	Time Limit	Memory Limit
Train Or Bus	1 s	1024 MB
Ducks And Buttons	1 s	1024 MB
Snacks	2 s	1024 MB
Itinerary	1 s	1024 MB
Lasers 2	4 s	1024 MB

Have Fun!

Task 1: `Train Or Bus`

You are a tourist who wishes to explore some cities. There are $n + 1$ cities, numbered from 1 to $n + 1$ in sequence. There are some buses and trains running between these cities.

To travel between cities i and $i + 1$, you have two transportation options:

- Take a **train**, which takes $a[i]$ units of time.
- Take a **bus**, which takes $b[i]$ units of time.

Determine the minimum total time required to travel from city 1 to city $n + 1$.

Input format

Your program must read from standard input.

The first line of input contains one integer n.

The following n lines of input each contain one integer. The i-th of these lines contains $a[i]$.

The following n lines of input each contain one integer. The i-th of these lines contains $b[i]$.

Output format

Your program must print to standard output.

Output a single integer, the shortest time taken to travel from city 1 to city $n + 1$.

The output should contain only a single integer. Do not print any additional text such as `Enter a number` or `The answer is`.

Subtasks

For all testcases, the input will satisfy the following bounds:

- $1 \leq n \leq 10$
- $1 \leq a[i] \leq 10$ for all $1 \leq i \leq n$
- $1 \leq b[i] \leq 10$ for all $1 \leq i \leq n$

Your program will be tested on input instances that satisfy the following restrictions:

Subtask	**Marks**	**Additional Constraints**
0	0	Sample test cases
1	100	No additional constraints

Sample Test Case 1

Input	Output

`3`
`7`
`7`
`5`
`9`
`8`
`1`
	`15`

Sample Test Case 1 Explanation

You start at city 1. You then:

- Take the train from city 1 to city 2 (7 units of time taken).
- Take the train from city 2 to city 3 (7 units of time taken).
- Take the bus from city 3 to city 4 (1 unit of time taken).

The total time taken is 15.

Task 2: `Ducks And Buttons`

Warning for C++ users: The large size of integers involved in this problem may require the use of the `long long` data type instead of the `int` data type.

Shor the Duck is playing a game with his friends! The game is played on a 1-dimensional grid consisting of n cells in, arranged in a row and numbered from 1 to n from left to right.

Each cell has a button. The button on cell i will be permanently pressed if at any point in time there are at least $a[i]$ ducks on that cell. Even if the ducks leave, the button remains pressed. To win the game, all n buttons must be pressed.

Initially, there are d ducks on cell 1. In one move, a single duck can move one cell left or right.

Determine the minimum total number of moves required to win the game. **It is guaranteed that it is possible to win the game with some number of moves.**

Input format

Your program must read from standard input.

The first line of input contains two space-separated integers n and d.

The second line of input contains n space-separated integers $a[1], a[2], \ldots, a[n]$.

Output format

Your program must print to standard output.

Output a single integer, the minimum total number of moves required to win the game.

Subtasks

For all test cases, the input will satisfy the following bounds:

- $1 \le n \le 200\,000$
- $1 \le d \le 10^9$
- $0 \le a[i] \le d$ for all $1 \le i \le n$
- It is possible to win the game with some number of moves.

Your program will be tested on input instances that satisfy the following restrictions:

Subtask	Marks	Additional Constraints
0	0	Sample test cases
1	8	$n = 2$
2	5	$a[i] = 0$
3	11	$a[i] \le 1$
4	6	All values of $a[i]$ are equal
5	19	$n, d \le 1000$
6	12	$a[i]$ is non-decreasing
7	16	$a[i]$ is non-increasing
8	23	No additional constraints

Sample Test Case 1

This test case is valid for subtasks 1, 5, 7, and 8.

Input	Output

`2 199`
`175 42`
	`42`

Sample Test Case 2

This test case is valid for subtasks 3, 5, and 8.

Input	Output

`5 3`
`1 1 0 1 0`
	`3`

Sample Test Case 3

This test case is valid for subtasks 4, 5, 6, 7, and 8.

Input	Output

`5 7`
`2 2 2 2 2`
	`8`

Sample Test Case 4

This test case is valid for subtasks 5, 6, and 8.

Input	Output

`7 5`
`1 3 3 4 5 5 5`
	`30`

Sample Test Case 5

This test case is valid for subtasks 5, 7, and 8.

Input	Output

`8 9`
`7 6 6 6 3 3 3 1`
	`28`

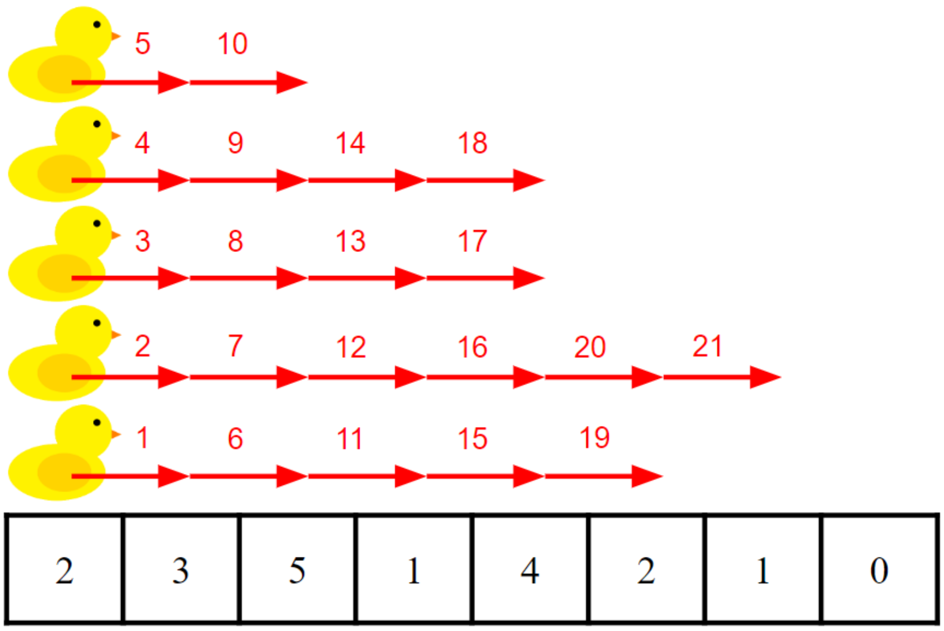
Sample Test Case 6

This test case is valid for subtasks 5 and 8.

Input	Output

`8 5`
`2 3 5 1 4 2 1 0`
	`21`

Sample Test Case 6 Explanation



One possible sequence of moves that minimises the total number of moves is shown above. Each red arrow is a move, and the number above it indicates the order of the moves, with move 1 coming first.

- Button 1 is first pressed before any moves occur.
- Button 2 is first pressed after move 3.
- Button 3 is first pressed after move 10.
- Button 4 is first pressed after move 11.
- Button 5 is first pressed after move 18.

- Button 6 is first pressed after move 20.
- Button 7 is first pressed after move 21.
- Button 8 is first pressed before any moves occur (since $a[8] = 0$).

Since all buttons are pressed by the end of all 21 moves, 21 moves is sufficient to win the game. It can be proven that 21 moves is the minimum total number of moves needed.

Sample Test Case 7

This test case is valid for subtasks 4, 6, and 8.

Input	Output

`4 1000000000`
`1 1 1 999999999`
	`2999999997`

Task 3: `Snacks`

Shor the Duck has prepared n plates of snacks to enjoy while watching movies! The i-th plate initially contains a snack with a deliciousness value of $a[i]$.

You need to process q queries. In the j-th query, Shor will do **both** of the following, in order:

1. Eat every snack whose deliciousness is between $l[j]$ and $r[j]$ (inclusive).
2. Then, replace each eaten snack with a new snack of deliciousness $x[j]$.

Before processing any queries, and after each query, Shor wants you to determine the sum of deliciousness of snacks across all plates.

Formally, you are given an array a of length n and must process q queries. Before processing any queries, print the sum of all elements in a. In the j-th query, update every element $a[i]$ such that $l[j] \leq a[i] \leq r[j]$ by setting $a[i] = x[j]$, then print the updated sum of all elements in a.

Input format

Your program must read from standard input.

The first line of input contains two space-separated integers n and q.

The second line of input contains n space-separated integers $a[1], a[2], \ldots, a[n]$.

The following q lines of input each contain three space-separated integers. The j-th of these lines contains $l[j], r[j]$, and $x[j]$, describing the j-th query.

Output format

Your program must print to standard output.

The output should contain $q + 1$ lines.

The first line of output should contain a single integer, the sum of all elements in a before all queries.

The following q lines of input should each contain one integer. The i-th of these lines should contain the sum of elements in a after the i-th query.

Subtasks

For all test cases, the input will satisfy the following bounds:

- $1 \leq n \leq 200\,000$
- $0 \leq q \leq 200\,000$
- $0 \leq a[i] \leq 10^9$ for all $1 \leq i \leq n$
- $0 \leq x[j] \leq 10^9$ for all $1 \leq j \leq q$
- $0 \leq l[j] \leq r[j] \leq 10^9$ for all $1 \leq j \leq q$

Your program will be tested on input instances that satisfy the following restrictions:

Subtask	Marks	Additional Constraints
0	0	Sample test cases
1	5	$q = 0$
2	12	$n, q \leq 2000$
3	21	$l[j] = r[j] \leq 200\,000$ and $a[i], x[j] \leq 200\,000$
4	13	$l[j] = r[j]$
5	16	$x[j] = 0$
6	33	No additional constraints

Sample Test Case 1

This test case is valid for subtasks 2, 3, 4, and 6.

Input	Output

5 3
1 6 2 4 6
6 6 3
2 2 3
3 3 5
	19
	13
	14
	20

Sample Test Case 2

This test case is valid for subtasks 2, 5, and 6.

Input	Output
`6 4`	`2049`
`929 121 5 3 919 72`	`1848`
`1 133 0`	`1848`
`70 79 0`	`0`
`900 999 0`	`0`
`1 1000 0`	

Sample Test Case 3

This test case is valid for subtasks 2 and 6.

Input	Output
`6 5`	`1259`
`7 72 727 123 321 9`	`1263`
`7 9 10`	`3352`
`10 72 727`	`3259`
`111 222 30`	`525`
`123 727 99`	`525`
`111 222 333`	

Sample Test Case 3 Explanation

Before all queries, the array a is $[7, 72, 727, 123, 321, 9]$, with a sum of 1259.

After the first query, the array a becomes $[10, 72, 727, 123, 321, 10]$, with a sum of 1263.

After the second query, the array a becomes $[727, 727, 727, 123, 321, 727]$, with a sum of 3352.

After the third query, the array a becomes $[727, 727, 727, 30, 321, 727]$, with a sum of 3259.

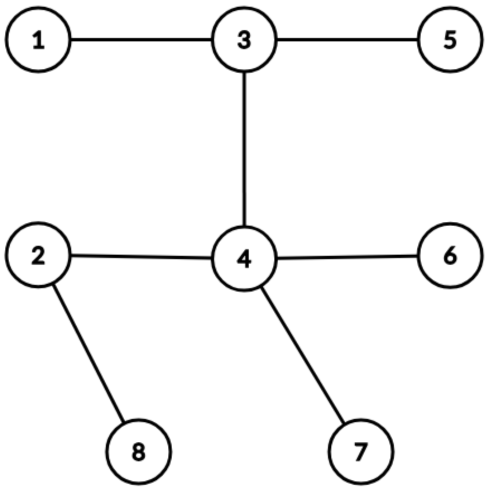
After the fourth query, the array a becomes $[99, 99, 99, 30, 99, 99]$, with a sum of 525.

After the fifth query, the array a becomes $[99, 99, 99, 30, 99, 99]$ with a sum of 525.

Task 4: `Itinerary`

The Scientific Committee is planning to visit n cities. The n cities are connected by exactly $n-1$ roads such that it is possible to move between all pairs of cities using the roads. Road i is built between cities $u[i]$ and $v[i]$.

Each city has its own airport. To begin the trip, the committee will fly from Singapore to one of the cities. To make the most out of their trip in the most efficient way possible, the committee wants to visit each city **at least once** by using each road **exactly twice** (once in each direction), before flying back home from the last city they find themselves in. A trip satisfying this condition is called a **tour**.



For example, let the diagram above represent a map of $n = 8$ cities. One possible tour starting from city 1 is $1 \to 3 \to 5 \to 3 \to 4 \to 2 \to 8 \to 2 \to 4 \to 6 \to 4 \to 7 \to 4 \to 3 \to 1$. Observe that this tour visits all cities a total of $2n - 1$ (equal to 15) times and ends at the same city it started from (city 1). It can be shown that these two properties are true for all tours in all possible maps of cities.

The committee also wants to visit m events which will happen in order from event 1 to event m. Event j will be held in city $s[j]$. A city can hold zero, one or multiple events, but no two consecutive events are held in the same city, i.e., $s[j] \neq s[j+1]$.

A tour that allows the committee to visit all events must visit the cities $s[1], s[2], \ldots, s[m]$ in order, not necessarily consecutively. Such a tour is called an **itinerary**. Formally, let $t[1], t[2], \ldots, t[2n-1]$ be the sequence of cities visited in some tour. A tour is an itinerary if and only if s is a subsequence of t. That is, s can be obtained by deleting zero or more elements from t and maintaining the order of the remaining elements. Using the same example as above, suppose that $m = 4$ and $s = [3, 5, 2, 7]$, then the tour $1 \to \underline{\mathbf{3}} \to \underline{\mathbf{5}} \to 3 \to 4 \to \underline{\mathbf{2}} \to 8 \to 2 \to 4 \to 6 \to 4 \to \underline{\mathbf{7}} \to 4 \to 3 \to 1$ above is an itinerary because the cities $3, 5, 2, 7$ are visited in order during the tour (underlined and marked in bold).

The committee is still deciding which city to start from, but they agree that a city is a good choice to start from if there exists an itinerary that starts from it. For all cities, help the committee determine if there exists at least one itinerary starting from that city.

Input format

Your program must read from standard input.

The first line of input contains two space-separated integers n and m, describing the number of cities and the number of events respectively.

The following $n - 1$ lines of input each contain two space-separated integers. The i-th of these lines contains $u[i]$ and $v[i]$, describing the endpoints of the i-th road.

The last line of input contains m space-separated integers $s[1], s[2], \ldots, s[m]$, describing the cities that are holding events.

Output format

Your program must print to standard output.

The output should contain n lines. If there exists an itinerary starting from city i, then the i-th line should contain a single integer 1. Otherwise, the i-th line should contain a single integer 0.

Subtasks

For all test cases, the input will satisfy the following bounds:

- $2 \leq n \leq 200\,000$
- $1 \leq m \leq 2n - 1$
- $1 \leq u[i], v[i] \leq n$ for all $1 \leq i \leq n - 1$
- $1 \leq s[j] \leq n$ for all $1 \leq j \leq m$
- $s[j] \neq s[j + 1]$ for all $1 \leq j \leq m - 1$
- It is possible to move between all pairs of cities using the roads.

Your program will be tested on input instances that satisfy the following restrictions:

Subtask	Marks	Additional Constraints
0	0	Sample test cases
1	7	$n \leq 1000, m = 2n - 1$
2	10	$u[i] = 1, v[i] = i + 1$
3	6	$n \leq 1000, u[i] = i, v[i] = i + 1$
4	7	$u[i] = i, v[i] = i + 1$
5	14	$n \leq 1000, m \leq 10$
6	5	$n \leq 1000$
7	19	$m \leq 10$
8	11	$s[1] = s[m]$
9	21	No additional constraints

Sample Test Case 1

This test case is valid for subtasks 5, 6, 7, and 9.

Input	Output
`8 4`	`1`
`1 3`	`0`
`2 4`	`1`
`3 4`	`1`
`4 6`	`0`
`5 3`	`1`
`2 8`	`1`
`7 4`	`0`
`3 5 2 7`	

Sample Test Case 1 Explanation

This test case is described as an example in the problem statement.

There are itineraries starting from cities 1, 3, 4, 6, and 7. An itinerary starting from city 1 is given in the problem statement.

On the other hand, it can be shown that there are no itineraries starting from cities 2, 5, and 8.

Sample Test Case 2

This test case is valid for subtasks 5, 6, 7, and 9.

Input	Output
`8 4`	`0`
`1 3`	`0`
`2 4`	`0`
`3 4`	`0`
`4 6`	`0`
`5 3`	`0`
`2 8`	`0`
`7 4`	`0`
`3 2 5 7`	

Sample Test Case 2 Explanation

This test case is the same as the example in the problem statement, except $s[2]$ and $s[3]$ are swapped. No itineraries exist at all.

Sample Test Case 3

This test case is valid for subtasks 1, 2, 5, 6, 7, 8, and 9.

Input	Output
`4 7`	`0`
`1 2`	`0`
`1 3`	`0`
`1 4`	`0`
`2 1 2 1 2 1 2`	

Sample Test Case 4

This test case is valid for subtasks 3, 4, 5, 6, 7, and 9.

Input	Output
`5 2`	`1`
`1 2`	`1`
`2 3`	`1`
`3 4`	`1`
`4 5`	`1`
`2 4`	

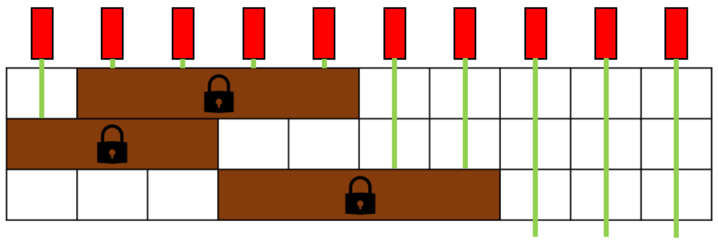
Task 5: `Lasers 2`

Pavement has purchased a laser toy from Gohcart (who originally bought it from Rar the Cat). The toy features a grid of h rows and w columns. The rows of the grid are numbered 1 to h from top to bottom, and the columns of the grid are numbered 1 to w from left to right.

Each row contains **exactly one** locked sliding wall. Initially, the wall in the i-th row spans columns $l[i]$ to $r[i]$ (1-indexed), and can be unlocked for $c[i]$ dollars. Once unlocked, it can be slid horizontally to any position along the i-th row, as long as it is aligned to the edges of the grid. No part of the wall can leave the left or right edges of the toy.

Each column contains a downward-facing laser positioned at the top of the column. If any sliding wall is contained in the i-th column, it will block the path of the laser in the i-th column.

A possible toy with $h = 3$ and $w = 10$ is depicted in the diagram below:



Given a total budget of k dollars, Pavement aims to maximise the number of lasers that are not blocked after optimally unlocking and sliding the walls. Determine the maximum number of unblocked lasers he can achieve.

Input format

Your program must read from standard input.

The first line of input contains three space-separated integers h, w and k, describing the number of rows, the number of columns and the budget, respectively.

The next h lines of input each contain three space-separated integers. The i-th of these lines contains $l[i]$, $r[i]$, and $c[i]$, describing the sliding wall in the i-th row.

Output format

Your program must print to standard output.

Output a single integer, the maximum number of unblocked lasers achievable.

Subtasks

For all testcases, the input will satisfy the following bounds:

- $1 \leq h, w \leq 2000$
- $0 \leq k \leq 10^9$
- $1 \leq l[i] \leq r[i] \leq w$ for all $1 \leq i \leq h$
- $0 \leq c[i] \leq 10^9$ for all $1 \leq i \leq h$

Your program will be tested on input instances that satisfy the following restrictions:

Subtask	Marks	Additional constraints
0	0	Sample test cases
1	6	$k = 0,\ c[i] = 10^9$
2	9	$l[i] = r[i]$
3	10	$h, w \leq 18$
4	7	$h, w \leq 100,\ k \leq 2000$
5	15	$h, w \leq 100$
6	23	$h, w \leq 500$
7	8	$r[1] - l[1] = r[2] - l[2] = \ldots = r[h] - l[h]$
8	22	No additional constraints

Sample Test Case 1

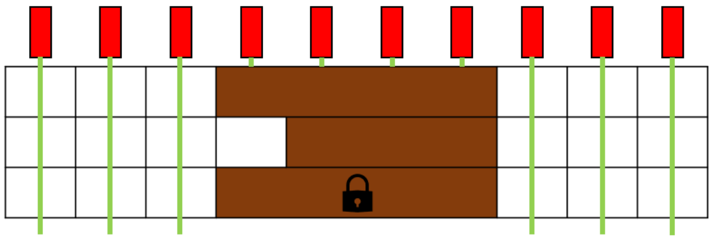
This test case is valid for subtasks 3, 4, 5, 6, and 8.

Input	Output

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

Sample Test Case 1 Explanation

The laser toy in the above figure corresponds to this test case. Pavement can unlock the first and second sliding walls for a total cost of $9 + 1 = 10$ dollars. He can then slide the first sliding wall such that it spans columns 4 to 7, and slide the second sliding wall to span columns 5 to 7.



This leaves 6 lasers (in columns 1, 2, 3, 8, 9, and 10) unblocked, which is the maximum possible.

Sample Test Case 2

This test case is valid for subtasks 2 to 8.

Input	Output

`10 10 50`
`8 8 0`
`3 3 0`
`6 6 2`
`7 7 9`
`1 1 50`
`5 5 21`
`6 6 4`
`10 10 4`
`10 10 3`
`10 10 3`
	`9`

Sample Test Case 3

This test case is valid for subtasks 1, 3, 4, 5, 6, and 8.

Input	Output

`4 17 0`
`2 4 1000000000`
`6 9 1000000000`
`8 13 1000000000`
`15 16 1000000000`
	`4`