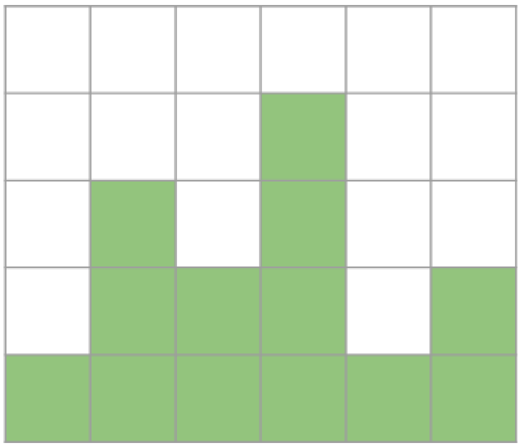
Task 1: `Pilot`

Rar the Cat has finally fulfilled his childhood dream of becoming a pilot, and wants to take his friend, Dinosaur, on a few scenic flights. Rar lives on a linear world, which can be described as a series of N integers, with the ith integer H_i indicating the height of the ith mountain from the leftmost edge of his world.

For example, the world described with $N = 6$, $H = \{1, 3, 2, 4, 1, 2\}$ will look like this:



Rar has a total of Q planes that he wishes to show off, with the ith plane having a maximum cruising altitude of Y_i metres. Each flight starts from the sth mountain and ends on the eth mountain. We may assume that $s \leq e$, i.e. **Rar will always fly toward the right**. As each of his planes have a maximum cruising altitude, he is unable to fly across, take off from, or land on a mountain where its height is greater than its cruising altitude, i.e. Rar is only able to fly over the ith mountain using the jth plane if $H_i \leq Y_j$.

For the ith plane, please help Rar determine the total number of different flights he can possibly take, i.e. the total number of ways Rar can choose s and e such that $s \leq e$, and there are no mountains between s and e inclusive of height greater than Y_i.

Input

Your program must read from standard input.

The first line of input will contain two integers, N and Q.

The second line of input will contain N integers, $H_1, \ldots, H_N$.

The third line of input will contain Q integers, $Y_1, \ldots, Y_Q$.

Output

Your program must print to standard output.

The output should contain Q lines with one integer each, with the number on the ith line indicating the total number of different flights Rar can take with his ith plane.

Subtasks

The maximum execution time on each instance is 1.0s. For all testcases, the input will satisfy the following bounds:

- $1 \leq N, Q, H_i, Y_i \leq 10^6$

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

Subtask	Marks	Additional Constraints
1	3	$N = 2$, $Q = 1$
2	10	$1 \leq N, Q \leq 30$
3	12	$1 \leq N, Q \leq 200$
4	15	$1 \leq N, Q \leq 10^3$
5	5	$1 \leq N \leq 10^5$, $Q = 1$, $Y_i = 10^6$
6	9	$1 \leq N, Q \leq 10^5$, $H_i = i$
7	14	$1 \leq N, Q \leq 10^5$, H is strictly increasing.
8	10	$1 \leq N \leq 10^5$, $Q = 1$
9	11	$1 \leq N, Q \leq 10^5$
10	11	-

Sample Testcase 1

This testcase is valid for subtasks 2, 3, 4, 9 and 10.

Input	Output

6 3
1 3 2 4 1 2
2 3 4
	5
	9
	21

Sample Testcase 1 Explanation

For the first plane, 5 flights are possible: (1, 1), (3, 3), (5, 5), (5, 6), (6, 6). For the second plane, 9 flights are possible: (1, 1), (1, 2), (1, 3), (2, 2), (2, 3), (3, 3), (5, 5), (5, 6), (6, 6). For the last plane, all 21 flights are possible.

Sample Testcase 2

This testcase is valid for subtasks 2, 3, 4, 9 and 10.

Input	Output
`6 3`	`0`
`2 2 5 2 2 2`	`9`
`1 2 10`	`21`

Sample Testcase 3

This testcase is valid for all subtasks.

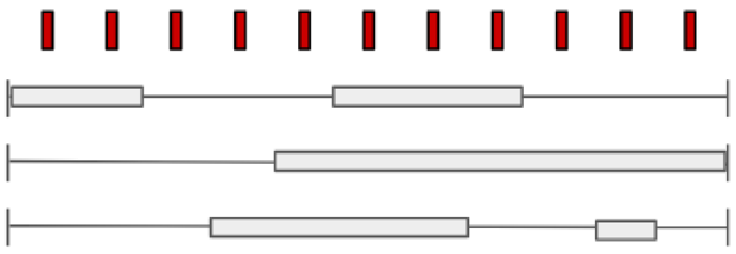
Input	Output
`2 1`	`3`
`1 2`	
`1000000`	

Task 2: `Lasers`

Mr. Panda knows cats love laser toys, and decides to invest in a laser toy for Rar the Cat. The laser toy that Mr. Panda has bought consists of L evenly spaced lasers at the top of the toy, pointing downward. The 1st laser is located 0.5 units away from the leftmost edge and the Lth laser is located 0.5 units away from the rightmost edge of the toy. Every adjacent lasers are of distance 1 unit.

There are R rows of sliding walls, with each row containing a set of non-overlapping walls. Precisely, each row contains some number of walls whose total length is at most L. These walls can be slid to any position on the same row, as long as their relative positions along the row remain the same and they do not overlap. A wall of width x units (where x is a positive integer) will block exactly x consecutive lasers.

A possible toy with $L = 11$ and $R = 3$ is depicted in the diagram below:



Rar the Cat, being the curious cat as he is, wishes to know: Out of the L lasers in his toy, how many lasers will always be blocked by at least one wall in all possible configurations of the toy?

Input

Your program must read from standard input.

The first line of the input will contain two integers, L and R.

The next R lines of input will describe one row each. It will start with a single integer X, the number of sliding walls in the row. X integers will follow, indicating the widths of the X walls in that row, with the first integer indicating the width of the leftmost wall. Note that the sum of widths of the walls on each row cannot exceed L units.

Output

Your program should print to standard output.

Output a single integer on a single line, the number of lasers that will be blocked by at least one wall in all possible configurations of the toy.

Subtasks

The maximum execution time on each instance is 1.0s. For all testcases, the input will satisfy the following bounds:

- $1 \leq R \leq 5 * 10^5$
- $1 \leq L \leq 10^9$
- $1 \leq \sum X \leq 5 * 10^5$
- $1 \leq \sum width \leq L$ for each row

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

Subtask	**Marks**	**Additional Constraints**
1	10	$R = 1, X = 1$
2	14	$X = 1$
3	20	$R = 2, 1 \leq L \leq 10^6$
4	21	$1 \leq L \leq 10^3, 1 \leq \sum X \leq 10^3$
5	22	$1 \leq L \leq 10^6$
6	13	-

Sample Testcase 1

This testcase is valid for subtasks 4, 5 and 6.

Input	Output

`11 3`
`2 2 3`
`1 7`
`2 4 1`
	`3`

Sample Testcase 1 Explanation

The 5th, 6th and 7th lasers from the left will always be blocked by the wall of width 7 in row 2.

Sample Testcase 2

This testcase is valid for subtasks 4, 5 and 6.

Input	Output

`10 3`
`3 1 5 1`
`4 2 2 3 1`
`3 1 6 2`
	`6`

Sample Testcase 2 Explanation

The 3rd, 4th, 5th, 6th, 7th and 9th lasers will always be blocked by at least one wall.

Sample Testcase 3

This testcase is valid for subtasks 1, 2, 4, 5 and 6.

Input	Output

`10 1`
`1 4`
	`0`

Sample Testcase 3 Explanation

Every laser can pass through all rows in at least one configuration.

Task 3: `Feast`

Gug is preparing a feast for his friends. The feast consists of N plates of food arranged in a single row, with the ith plate from the left giving A_i points of satisfaction if eaten. As some plates of food might be rotten, it is possible that A_i is negative.

There are a total of K people involved in the feast, and each person will be assigned a consecutive segment of plates to consume. This segment can possibly be empty. The segments of two people cannot overlap, as food cannot be eaten twice. Gug wishes to assign the plates to his friends such that the sum of satisfaction points of all the plates of food consumed is maximised.

Input

Your program must read from standard input.

The input starts with a line with two integers N and K.

The next line will contain N integers $A_1, \ldots, A_N$.

Output

Your program must print to standard output.

The output should contain a single integer on a single line, the sum of satisfaction points in an optimal assignment.

Subtasks

The maximum execution time on each instance is 1.0s. For all testcases, the input will satisfy the following bounds:

- $1 \leq K \leq N \leq 3 * 10^5$
- $0 \leq |A_i| \leq 10^9$

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

Subtask	Marks	Additional Constraints
1	4	$A_i \geq 0$
2	8	There will be at most one position where $A_i < 0$.
3	18	$K = 1$
4	10	$1 \leq K \leq N \leq 80$
5	11	$1 \leq K \leq N \leq 300$
6	20	$1 \leq K \leq N \leq 2000$
7	29	-

Sample Testcase 1

This testcase is valid for subtasks 3, 4, 5, 6 and 7.

Input	Output

`6 1`
`1 -2 3 -1 5 -6`
	`7`

Sample Testcase 1 Explanation

It is optimal to assign the only person to the segment `[3, -1, 5]`.

Sample Testcase 2

This testcase is valid for subtasks 2, 4, 5, 6 and 7.

Input	Output

`6 2`
`1 2 3 -10 5 6`
	`17`

Sample Testcase 2 Explanation

Picking the consecutive segments `[1, 2, 3]` and `[5, 6]` is the most optimal solution.

Sample Testcase 3

This testcase is valid for subtasks 4, 5, 6 and 7.

Input	Output

`6 4`
`-1 -2 -1 0 -5 -1`
	`0`

Sample Testcase 3 Explanation

As all the satisfaction points are non-positive, it is optimal to choose empty segments for all four people.

Task 4: `Rigged Roads`

Silvermill is on a tight budget lately, and Peanut, its mayor, is intending to demolish some roads to save on maintenance costs. Silvermill can be described as a city with N road intersections and E roads running between them, with the ith road connecting intersections A_i and B_i. Each road intersection is labelled from $1, \dots, N$ and each road is labelled from $1, \dots, E$. It is guaranteed it is possible to travel between any pair of road intersections directly or indirectly, and no two roads share the same endpoints.

To facilitate this effort, Peanut has hired you to help assess the maintenance cost of the roads. The task from Peanut is as follows: You need to report a list $W = (W_1, W_2, \dots, W_E)$ such that W is a permutation of $1, \dots, E$ and W_i is the cost to keep the ith road.

Peanut will then pick a subset of roads to keep such that:

- All the road intersections remain connected.
- The sum of costs of the kept roads are minimised.

In other words, Peanut will keep the **minimum spanning tree**, based on the weights given by you. Note that the minimum spanning tree is unique since the costs are distinct.

Unknown to Peanut, you have a hidden agenda. You wish to keep a subset R of roads that form a spanning tree. Notice that you can convince Peanut to pick R by carefully choosing W. Your goal is to find the lexicographically smallest[1] permutation W that satisfies the above condition.

In summary, given a subset R of roads that forms a spanning tree, find the lexicographically smallest weight assignment W such that the minimum spanning tree of the city is R.

Input

Your program must read from standard input.

The input starts with a line with two integers N and E.

E lines will follow. The ith line contains two integers, A_i and B_i, describing a single road.

The last line of input will contain $N - 1$ integers, the labels of roads in R, the set of roads you wish to keep.

[1]$(W_1, \dots, W_E)$ is smaller than $(W'_1, \dots, W'_E)$ if there exists $1 \le p \le E$ such that $W_p < W'_p$ and $W_i = W'_i$ for $i = 1, \dots, p-1$.

Output

Your program must print to standard output.

The output should contain E integers on a single line, the lexicographically minimal permutation W that would result in R being selected as the minimum spanning tree.

Subtasks

The maximum execution time on each instance is 2.0s. For all testcases, the input will satisfy the following bounds:

- $1 \le N, E \le 3 * 10^5$
- $1 \le A_i \ne B_i \le N$
- $1 \le R_i \le N$
- It is possible to travel between any two road intersections using only roads in R.

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

Subtask	Marks	Additional Constraints
1	8	$1 \le N, E \le 9$
2	19	$1 \le N, E \le 10^3$
3	9	$A_{R_i} = 1$, $B_{R_i} = i + 1$, i.e. R is a star.
4	10	$A_{R_i} = i$, $B_{R_i} = i + 1$, i.e. R is a line.
5	10	$E = N$, $A_i = i$, $B_i = i$ mod N + 1
6	12	$E = N$
7	32	-

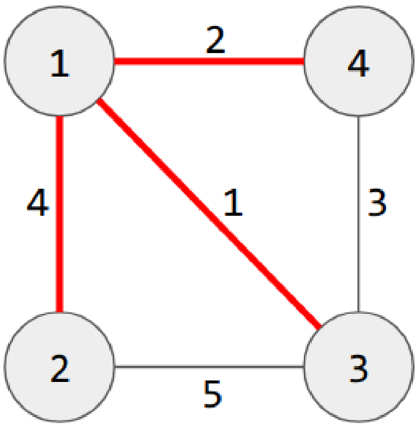
Sample Testcase 1

This testcase is valid for subtasks 1, 2, 3, 4 and 7.

Input	Output

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

Sample Testcase 1 Explanation



The graph above shows the roads and the road intersections (see input). The numbers on the edges are the weights assigned by you (see output). The highlighted roads are selected by Peanut, which is equivalent to R in the input. As such, the output is correct.

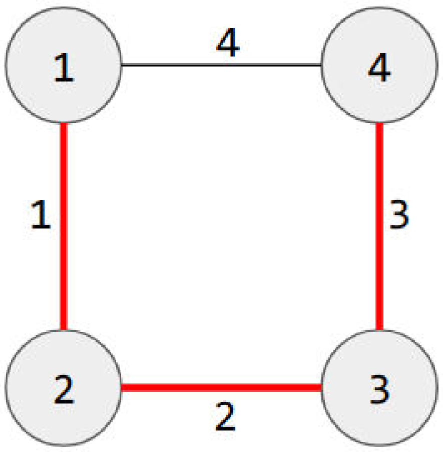
Sample Testcase 2

This testcase is only valid for subtasks 1, 2, 3, 5, 6 and 7.

Input	Output

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

Sample Testcase 2 Explanation



Task 5: `Shuffle`

Note that only C++ and Java may be used for this task.

Lim Li loves anime, and has recently bought a complete season of Umaru from Amazon. The N episodes of Umaru comes in a set of N CDs, with one episode in each CD. The CDs are labelled $1, \ldots, N$, and the episodes of Umaru are also numbered $1, \ldots, N$. She was prepared to begin watching, when she realised that due to a manufacturing fault, the episode numbers don't match up with the labels on the CDs!

She immediately contacts Amazon, and is informed that although the CD labels do not match up with the episode numbers, the episodes are merely shuffled and **no single episode is missing from the set**. Wishing to avoid spoilers, Lim Li is unwilling to play the CDs herself to figure out which episode lies in each CD.

Lim Li thus decides to engage the help of her friend, Rar the Cat. The procedure she will follow is as follows:

- Lim Li will group her CDs into B boxes, containing K CDs each. ($N = B \times K$)
- The boxes are shipped to Rar. In transit, the CDs in each box may move around within the box and the boxes themselves may arrive in a different order. As such, the order of CDs in each box and the order of the boxes themselves may be shuffled in transit.
- Rar will receive the boxes, and play the CDs on his own CD player.
- Rar will then write down the set of episodes he finds in each box, and send this information back to Lim Li in B separate pieces of paper. Each piece of paper states K integers, the episode numbers of the CDs inside the box.
- The entire set of N CDs are returned to Lim Li.

This procedure can be done a maximum of Q times, before Rar gets annoyed and stops helping her. Help Lim Li figure out the episode numbers in each CD with minimal help from Rar.

Implementation Details

This is an interactive task. Do not read from standard input or write to standard output.

You are to implement the following function:

- C++: `vector<int> solve(int N, int B, int K, int Q, int ST)`
- Java: `public int[] solve(int N, int B, int K, int Q, int ST)`

The `solve` function will be called exactly once.

You are allowed to call the following grader functions to complete the task:

- C++: `vector<vector<int>> shuffle(vector<vector<int>> boxes)`
- Java: `public static int[][] shuffle(int[][] boxes)`

The function `shuffle` is called with a 2-dimensional array `boxes`, describing the grouping of the N CDs into B boxes by Lim Li. `boxes` is to be formatted such that it contains B 1-dimensional arrays, each of which contains K integers representing the labels of the CDs in the box. If your program calls this function more than Q times or with invalid parameters, the program will terminate immediately and you will be given a *Wrong Answer* verdict.

The array passed to the function `shuffle` will **not** be modified. In other words, you can expect the 2D array `boxes` to remain the same after calling the function `shuffle`.

The function `shuffle` will return a 2-dimensional array, describing the information as given by Rar. The 2-dimensional array consists of B 1-dimensional arrays, each contains K integers describing the episode numbers of the CDs in the box.

Sample Interaction

Suppose $N = 6, B = 3, K = 2, Q = 100$, and the testcase belongs to Subtask 2. The episodes are `[3, 1, 4, 5, 2, 6]`. Your function will be called with the following parameters:

```
solve(6, 3, 2, 100, 2)
```

A possible interaction could be as follows:

- `shuffle([[1, 2], [3, 4], [5, 6]]) = [[6, 2], [5, 4], [3, 1]]`

  The 2D array [[1, 2], [3, 4], [5, 6]] represents the CD labels in the three boxes sent by Lim Li. Since the boxes are mis-ordered and the CDs in each box are shuffled, the boxes received by Rar may correspond to the 2D array [[6, 5], [4, 3], [1, 2]]. Rar finds the episodes for these CD labels and reports [[6, 2], [5, 4], [3, 1]].

- `shuffle([[2, 6], [3, 1], [5, 4]]) = [[6, 1], [2, 5], [4, 3]]`

  Since the boxes are shuffled, the boxes received by Rar may correspond to the 2D array [[6, 2], [5, 4], [3, 1]]. Rar finds the episodes for these CD labels and reports [[6, 1], [2, 5], [4, 3]].

- `shuffle([[6, 5], [4, 2], [3, 1]]) = [[5, 1], [3, 4], [2, 6]]`

  Since the boxes are shuffled, the boxes received by Rar may correspond to the 2D array [[4, 2], [1, 3], [5, 6]]. Rar finds the episodes for these CD labels and reports [[5, 1], [3, 4], [2, 6]].

At this point, your program decides that it has enough information and concludes that the episodes are `[3, 4, 1, 5, 2, 6]`. As such, your program will return `[3, 4, 1, 5, 2, 6]`. As the episodes are correct, and the program has used less than 100 queries, it would be deemed as correct for this testcase.

Subtasks

The maximum execution time on each instance is 1.0s. For all testcases, the input will satisfy the following bounds:

- $B, K \geq 2$
- $N \geq 6$
- $N = B \times K$

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

Subtask	Marks	Additional Constraints
1	2	$N = 6, B = 2, K = 3, Q = 100$
2	3	$N = 6, B = 3, K = 2, Q = 100$
3	12	$N \leq 1000, Q = 12$. The order of the B boxes will remain the same.
4	16	$N \leq 1000, K = 2, Q = 4$
5	15	$N \leq 1000, B = 2, Q = 12$
6	52	$N \leq 1000, Q = 2000$, $B, K > 2$. Please read ”Scoring” for further details.

Scoring

Subtask 6 is a special scoring subtask. Your score depends on the maximum number of queries q you make in any testcase.

- If $q > 2000$, you will score 0 points.
- If $500 < q \leq 2000$, you will score 8 points.
- If $50 < q \leq 500$, you will score 17 points.

- If $9 < q \leq 50$, you will score $22 + 30 * (\frac{50-q}{41})^2$ points.
- If $q \leq 9$, you will score 52 points.

Testing

You may download the grader file, the header file (for C++) and a solution template under *Attachments*. The sample testcases are also provided for your reference. Please use the compile and run scripts provided (the .sh files) for testing.

Grader Input Format for Testing

- one line with five integers, N, B, K, Q and ST, where ST is the subtask number.
- one line with N integers, the episodes, $E_1, E_2, ..., E_N$