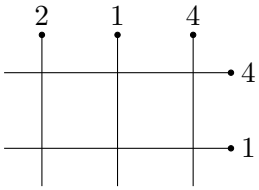
Political cost (cost)

In the city where you live there are N streets going East-to-West (from 0th Street to $(N-1)$th Street) and M avenues going North-to-South (from 0th Avenue to $(M-1)$th Avenue). Every street or avenue has a *political weight*, which is the importance of the most important citizen living in it. We represent political weights as two arrays $A[0 \ldots N-1]$ and $B[0 \ldots M-1]$ of integers from 1 to K. The following plot represents such a city with 2 streets and 3 avenues, with political weights of $A = [1, 4]$ and $B = [2, 1, 4]$ respectively.



The major wants to organize a parade through the city. If the parade goes through the intersection of xth Street with yth Avenue, the traffic of both roads will be disrupted, and the major will incur a *political cost* of $\max(A[x], B[y])$. If the parade crosses goes through multiple intersections, the political cost will be the *maximum* of the political cost of each intersection. Notice that costs are not summed: what matters is not how many people the parade bothers, but how important is the most important citizen the parade bothers.

The *political distance* between two intersections is the smallest *political cost* of a parade departing from the first intersection and arriving at the second intersection. Your job is to compute the sum of the political distances between all pairs of intersections in the city.

Implementation

You will have to submit a single `.cpp` source file.

> ☞ Among this task's attachments you will find a template `cost.cpp` with a sample implementation.

You have to implement the following function:

C++
```
int solve(int N, int M, int K, vector<int> A, vector<int> B);
```

- Integer N represents the number of East-to-West streets.
- Integer M represents the number of North-to-South avenues.
- The array A, indexed from 0 to $N-1$, contains the values $A_0, A_1, \ldots, A_{N-1}$, where A_i is the political weight of the i-th East-to-West street.
- The array B, indexed from 0 to $M-1$, contains the values $B_0, B_1, \ldots, B_{N-1}$, where B_i is the political weight of the i-th North-to-South street.
- The function should return the sum of the political distances between all possible pairs of intersections, **modulo** 1000003.

The grader will call the function `solve` and will print its return value to the output file.

Sample Grader

The task's directory contains a simplified version of the jury grader, which you can use to test your solution locally. The simplified grader reads the input data from `stdin`, calls the functions that you must implement, and finally writes the output to `stdout`.

The input is made up of 3 lines, containing:

- Line 1: the integers N, M and K.
- Line 2: the integers A_i, space-separated.
- Line 3: the integers B_i, space-separated.

The output is made up of a single line, containing the value returned by the function `solve`.

Constraints

- $1 \leq N \leq 3 \times 10^5$.
- $1 \leq M \leq 3 \times 10^5$.
- $1 \leq K \leq N + M$.
- $1 \leq A_i \leq K$ for $i = 0 \ldots N - 1$.
- $1 \leq B_i \leq K$ for $i = 0 \ldots M - 1$.

Scoring

Your program will be tested on a set of test cases grouped by subtask. To obtain the score associated to a subtask, you need to correctly solve all the test cases it contains.

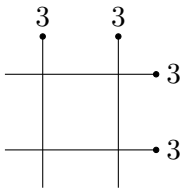
- **Subtask 1 [0 points]**: Sample test cases.
- **Subtask 2 [10 points]**: $N \leq 10^1, M \leq 10^1$.
- **Subtask 3 [10 points]**: $N \leq 10^2, M \leq 10^2$.
- **Subtask 4 [10 points]**: $N = 1, M \leq 10^4$.
- **Subtask 5 [10 points]**: $N = 1, M \leq 10^5$.
- **Subtask 6 [10 points]**: $N \leq 10^3, M \leq 10^3$.
- **Subtask 7 [10 points]**: $N \leq 10^4, M \leq 10^4$.
- **Subtask 8 [10 points]**: $N \leq 10^5, M \leq 10^5$ and the arrays A and B are non-decreasing, that is, if $i < j$, then $A_i \leq A_j$ and $B_i \leq B_j$.
- **Subtask 9 [10 points]**: $N \leq 10^5, M \leq 10^5, K \leq 10^1$.
- **Subtask 10[10 points]**: $N \leq 10^5, M \leq 10^5$.
- **Subtask 11[10 points]**: No additional constraints.

Examples

stdin	stdout
2 2 4 3 3 3 3	48
1 3 4 2 2 3 1	25
2 3 5 1 4 2 1 4	135

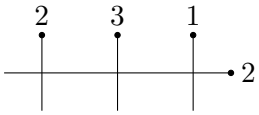
Explanation

In the **first sample case**, we have a city with 2 streets and 2 avenues, all of them of political weight 3:



There are 16 different pairs of intersections. Since the political distance between each pair of intersections is 3, the solution is $3 \cdot 16 = 48$.

In the **second sample case** there is 1 street and 3 avenues, with political weights $A = [2]$ and $B = [2, 3, 1]$ respectively:



There are 9 pairs of intesections. Three of these pairs start and end at the same intersection, and have political distances of 2, 3 and 2 respectively (the rightmost avenue has a political weight of 1, but the political weight of the only street is 2, so the political distance of any parade is at least 2). For each remaining pair of intersections, the parade joining them must cross the middle avenue, and hence must have political distance of 3. So the total sum is $2 + 3 + 2 + 6 \cdot 3 = 25$.

The **third sample case** corresponds to the example given in the statement of the problem. Here, there are 2 streets and 3 avenues. You can check, with some patience, that the sum of the political distances of the 36 pairs of intersections is 135.