Harry loves walking in Varna. To be precise - in the "Chaika" district. While Harry was planning his next trip, he found out that a construction company "Galab" has started the construction of new buildings in the neighborhood. This didn't appeal to him at all because it would spoil the landscape. Fortunately, Barry, a friend of Harry, works for the company and has the plans for the construction.

The neighborhood can be envisioned as an matrix. The rows and columns are indexed from 1. Barry knows the positions of each of the intended buildings. Building with number will be constructed in cell (more than one building can be constructed in the same cell). Each cell has a certain view beauty factor. A building in cell decreases the beauty of all cells in its row and column by the distance to the cell. The distance between two cells in the same row and is and the distance between two cells in the same column and is.

Harry wants to know what the view beauty factor will be for each cell after all the buildings are constructed. At the moment, he is busy making a list of places he wants to visit (again) in Varna, so there is no time for these simple calculations. He needs your help. Write a program calledscenery.cpp to calculate the view beauty factor for each cell of the neighborhood.

**Input**

From the first line of the scenery.in file, three positive integers are entered - , and , respectively the size of the neighbourhood and the number of buildings to be built. On each of the next lines, numbers are entered – the initial view beauty factor for each cell. On the following lines, two numbers are entered – the row and column for each building.

**Output**

In the scenery.out file, you need to print lines with numbers that represent the view beauty factor for each of the cells in the neighborhood after all the buildings are built.

**Constraints**

**Time limit: 2sec.**

**Memory limit: 256 MB.**

**Sample test:**

|  |  |
| --- | --- |
| **Input(scenery.in)** | **Output(scenery.out)** |
| 4 5 5  3 9 -4 15 2  11 -5 -2 8 1  7 2 14 -9 17  3 12 -8 4 -7  2 4  3 3  1 2  3 2  4 3 | 2 7 -10 12 -1  8 -9 -6 8 0  4 -1 12 -13 12  1 7 -9 1 -9 |