KString

SEASON 2021/2022 – FORTH ROUND

A string is called **K**-symmetrical if it can be represented as **K** concatenated copies of another string. For example, the string `"abababab"` is simultaneously **1**-symmetrical (1 × `"abababab"`), **2**-symmetrical (2 × `"abab"`) and **4**-symmetrical (4 × `"ab"`), but not **3**-symmetrical or **6**-symmetrical. Obviously every string is **1**-symmetrical.

You're given a string **S**, consisting of lowercase latin letters and a natural number **K**. Your task is to rearrange the letters in the string **S** in such a way that the resulting string becomes **K**-symmetric or determine that it is impossible to do so.

Input

On the first line of the file `kstring.in` the string **S** and the number **K** are given.

Output

On one line in the file `kstring.out`, print the rearranged letters of **S** so that they form a **K**-symmetrical string, or **"-1"** if this is not possible. If there is more than one solution, print any of them.

Constraints

$1 \leq |S| \leq 10^5$

$1 \leq K \leq |S|$

Time limit: 0.2 sec.

Memory limit: 256 MB.

Sample tests

Input (kstring.in)	Output (kstring.out)
abacbc 2	abcabc

Input (kstring.in)	Output (kstring.out)
abbaba 3	bababa

Input (kstring.in)	Output (kstring.out)
abccaba 2	-1

Scenery

SEASON 2021/2022 – FORTH ROUND

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 $N \times M$ matrix. The rows and columns are indexed from 1. Barry knows the positions of each of the intended Q buildings. Building with number i will be constructed in cell (x_i, y_i) (more than one building can be constructed in the same cell). Each cell has a certain view beauty factor $a_{i,j}$. A building in cell (x, y) 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 (x, y_1) and (x, y_2) is $|y_1 - y_2|$ and the distance between two cells in the same column (x_1, y) and (x_2, y) is $|x_1 - x_2|$.

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 called `scenery.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 - N, M and Q, respectively the size of the neighbourhood and the number of buildings to be built. On each of the next N lines, M numbers are entered – the initial view beauty factor for each cell. On the following Q lines, two numbers are entered – the row and column for each building.

Output

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

Constraints

$1 \leq N, M \leq 1000$

$1 \leq Q \leq 10^5$

$-10^9 \leq a_{i,j} \leq 10^9$

Scenery

SEASON 2021/2022 – FORTH ROUND

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

Dominoes

SEASON 2021/2022 – FORTH ROUND

Sashka won't stop playing with her favourite dominoes. She has four types of dominoes (Fig. 1). They have two sides, left and right respectively. Both of the sides are colored in blue or red. For convenience, Sashka denoted them as follows:

- Domino №1: blue-blue
- Domino №2: blue-red
- Domino №3: red-blue
- Domino №4: red-red

You should note that blue-red is different from red-blue.

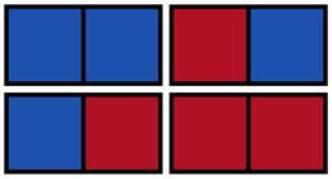

Fig. 1

Sashka takes out all the dominoes she has and orders them in a sequence. Sashka denotes a sequence of dominoes as *beautiful* if each adjacent side of the dominoes is with different color. More precisely, if the left side of the i-th domino is l_i and the right is r_i, then for every $1 \leq i \leq N-1$, $r_i \neq l_{i+1}$. A sample *beautiful* sequence is shown below (Fig. 2).

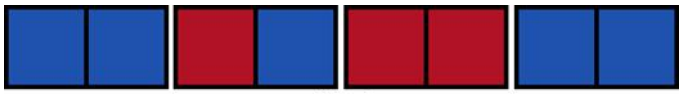

Fig. 2

In Bulgaria there are T domino sets, where the i-th set contains $d_{i,1}$ dominoes from the first type, $d_{i,2}$ from the second type, $d_{i,3}$ from the third type and $d_{i,4}$ from the forth type. At first, Sashka took the *intermediate* assessment of a domino set to be the count of different *beautiful* sequences, which can be made from it. Two domino sequences are different if two different dominoes are on the same positions in the sequences. She realized that the count of such sequences could be pretty big, so she changed the method of assessment. The new *intermediate* assessment of a domino set is taken to be the count of achievable *beautiful* sequences modulo 10^9+7. The bigger the remainder is, the better the *intermediate* assessment is for the set, but there is more. Sashka is allowed to recolor K_i dominoes in the i-th set to increase her assessment of the set. Recoloring of a domino in the i-th set goes like this: Sashka chooses two types of dominoes x and y ($x \neq y$, $1 \leq x, y \leq 4$, $d_{i,x} \neq 0$) and decreases $d_{i,x}$ by 1 and increases $d_{i,y}$ by 1 ($d_{i,x} := d_{i,x} - 1, d_{i,y} := d_{i,y} + 1$, where $:=$ is the sign for assignment). **So in the end, the *final* assessment of a domino set is the maximal *intermediate* assessment of all possible recolorings**. Sashka wants to find the *final* assessment of all dominoes set in Bulgaria, but it'll be hard for her to calculate it by hand. That's why she asks you, as her third cousin, to write a program `dominoes.cpp`, which will find it.

Input

On the first line of `dominoes.in` the number T is given – the number of domino sets in Bulgaria. The i-th of the next T rows contains 5 natural numbers, $d_{i,1}, d_{i,2}, d_{i,3}, d_{i,4}$, K_i respectively.

Dominoes

SEASON 2021/2022 – FORTH ROUND

Output

For each of the domino sets, you should print a number in `dominoes.out`, which is its *final* assessment.

Constraints

$1 \leq T \leq 5$

$1 \leq d_{i,1} + d_{i,2} + d_{i,3} + d_{i,4} \leq 120$

$0 \leq d_{i,1},\ d_{i,2}, d_{i,3}, d_{i,4} \leq 120$

$0 \leq K_i \leq 20$

For every $1 \leq i, j \leq T$, $d_{i,1} + d_{i,2} + d_{i,3} + d_{i,4} = d_{j,1} + d_{j,2} + d_{j,3} + d_{j,4}$

Time Limit: 1 sec.
Memory Limit: 256 MB.

Sample testcases

Input (dominoes.in)	Output (dominoes.out)
3 1 1 1 0 0 1 0 1 1 0 3 0 0 0 2	1 3 3
3 3 3 3 1 3 8 1 1 0 4 10 0 0 0 5	120 40 5
2 9 2 5 10 5 9 8 4 5 3	3171168 2944656
3 30 30 30 30 20 30 60 15 15 20 60 60 0 0 20	999079969 997861536 0

Dominoes

SEASON 2021/2022 – FORTH ROUND

Explanation of the sample testcases

For the first testcase:

- For the first domino set the only possible sequence is:

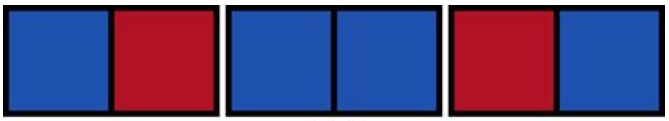

Fig. 3

- For the second domino set, the three possible sequences are the following:

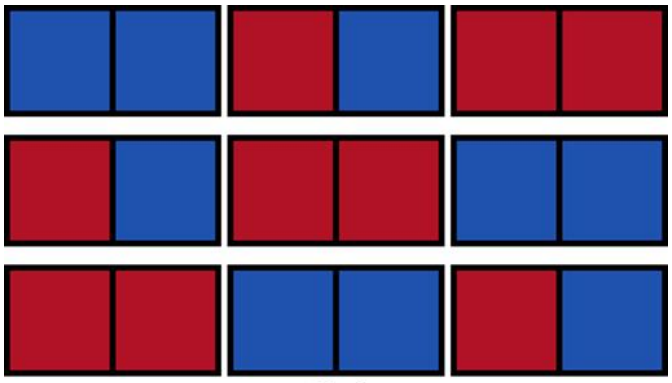

Fig. 4

- For the third domino set, $\{3,0,0,0\} \rightarrow \{1,0,1,1\}$

The rest of the sample testcases probably have many beautiful sequences, but the space won't be enough ☺.