Columbia with Elizabeth

Time limit: 1 sec

There is an imaginary city called Columbia. The city can be described as a grid of **R** rows and **C** columns. Each cell in the grid is a block of the city. Each block has its own dweller and they don't want other people to pass by easily. Hence, they set up entrance fee. To go into any block, you have to pay the entrance fee which may vary among blocks. From any block, you can go into an adjacent block. Two blocks are adjacent when they share a side.

You are a member of the block at the top left corner of Columbia. Since you have a lot of friends in every block, you want to know the minimum cost of traveling from your block to all other blocks. You don't have to pay the fee for your starting block.

You have another friend, Elizabeth, who is always be with you. Elizabeth will travel with you along the blocks. She has a special ability that can open a “Tear”. A tear is a special portal that allows you to move into another block without paying the fee. Elizabeth can use this abilities only 2 times for one trip. Additionally, a Tear can be use to move to block that is not more than 2 blocks away, i.e., it can move to blocks that are adjacent to the blocks that are adjacent to the current block.

Input

- The first line contains two integers that describe the size of the city **R** and **C.**
- The next **R** lines describe the entrance fee. Each line gives the cost of each row, starting from the top row to the bottom row using the following format.
    - For each row, there are **C** integers that indicate the entrance fee of each block in that row, starting from the leftmost block to the rightmost block. The fee is non-negative integer not exceeding 1,000.

Output

Your program should output a grid of R row and C columns which give the minimal cost of traveling to each block in the city.

Limits

- For 50% of the test data, **R*C** will not exceeding 5,000
- For 100% of the test data, **R*C** will not exceeding 200,000

Example

Input	Output
4 4 1 1 1 1 9 8 7 1 1 1 1 1 1 2 4 9	0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 2 // to go to the lower right cell, first, we move right twice to cell (0, 2) paying fee of 2 units and then we use tear twice to move to (1,3) and to (3, 3) paying nothing.
5 5 1 2 1 2 9 9 9 9 9 9 1 1 1 9 4 9 9 9 9 8 2 2 2 8 8	0 0 0 0 0 0 0 0 0 2 0 0 0 1 2 0 0 1 2 5 0 1 2 5 9 // to go to the last cell, first, we move right 3 times to cell (0, 3) paying a total fee of 5 units and then we use tear to move to (1,4), paying nothing. After that, we move down to cell (2, 4) paying additional 4. Finally, we use tear to move to (4,4), paying nothing.