Problem H. Greedy Algorithm

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

I'm playing Super Mario Galaxy 3 (thanks for the copy, Nintendo). Most of the levels are small planets in a shape of sphere, but bonus levels are something different. They are in a shape of **torus**. You can imagine it as rectangle with both pairs of opposite sides glued together. As a tribute to 8-bit predecessors, the surface of the planet is a small rectangular grid. Each cell in the grid has its own height.

Playing the bonus level consists of two parts. In the first part I can terraform the level by applying zero or more operations. In one operation I choose a row or a column in the grid, and increase the heights of all the cells in that row or column by one. I can perform this operation any number of times with same or different rows and columns.

After the terraforming a coin appears at each common side of two cells of equal height, and I can collect them. I'm good at platforming, so collecting all the coins is not a problem. Designing *algorithm* for terraforming the level so that the maximum possible number of coins appear — that's the problem. The problem for you, actually.

Input

The first line contains two integers n and m ($2 \le n, m \le 50$) — the dimensions of the rectangular grid.

The next n lines describe the initial heights of all cells. The i-th of them contains m integers h_{i1}, h_{i2}, $\ldots$, h_{im}, where h_{ij} denotes the height of the cell with coordinates (i, j).

All the heights are between 0 and 500, inclusive. It is **not required** that this holds after the terraforming.

Output

Print a single integer — the maximum number of coins can appear if I terraform the level optimally.

Examples

standard input	standard output
2 3 1 2 3 4 5 99	8
3 3 3 2 4 2 2 3 5 4 6	14
5 4 3 6 10 8 0 6 8 8 2 4 5 6 1 5 9 6 3 6 11 12	16

Note

The level from the first example after an optimal terraforming: