Tiling Game

Barchin and Charos are playing a game on a grid of $2N \times 2M$ unit square cells. The rows are numbered 0 to $2N-1$ from top to bottom, and the columns are numbered 0 to $2M-1$ from left to right. For $0 \le i < 2N$ and $0 \le j < 2M$, we denote the cell in row i and column j by (i, j).

Barchin gives Charos $N \cdot M$ **blocks**, one by one. Each block is a 2×2 square consisting of four 1×1 **tiles**. Barchin has colored every tile of the block either black or white while guaranteeing that **at least one tile is white**.

Charos must place each block on the grid **immediately after receiving it**, without knowing what blocks she will receive later. Blocks cannot be rotated. Each block must be placed entirely inside the grid, covering exactly four grid cells. Moreover, the top-left tile of each block must cover a cell whose row and column coordinates are both even. Each cell of the grid can be covered by at most one block.

Barchin wins the game if, after any block is placed, there exists a 2×2 square of cells that is covered by four black tiles. Formally, if cells (a, b), $(a+1, b)$, $(a, b+1)$, $(a+1, b+1)$ are all covered by black tiles for some $0 \le a < 2N-1$ and $0 \le b < 2M-1$, then Barchin wins. The indices a and b do not need to be even.

Charos wins if she places all $N \cdot M$ blocks without Barchin ever winning. Note that placing $N \cdot M$ blocks will completely cover the grid.

Your task is to implement a strategy for Charos to win the game. It can be proven that, under the given constraints, Charos can always place the blocks so as to guarantee a win, regardless of the colorings of the blocks she receives later.

Implementation Details

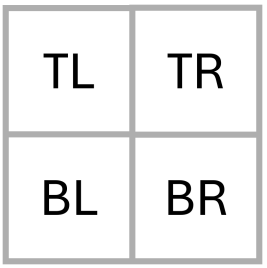
You should implement two procedures:

```
void init(int N, int M)
```

- N: half the number of rows in the grid.
- M: half the number of columns in the grid.
- The procedure is called exactly once per test case, at the beginning of the execution of your program.

```
std::pair<int, int> receive_block(int TL, int TR, int BL, int BR)
```

- TL, TR, BL, BR: the colors of the top-left, top-right, bottom-left, and bottom-right tiles of the current block, respectively, as shown in the figure below. Each value is either 0 (white) or 1 (black).
- This procedure is called exactly $N \cdot M$ times per test case, after the initial call to `init`.



This procedure should return a pair of integers (i, j), where i is the row coordinate and j is the column coordinate of the cell where the top-left tile of this block should be placed. Both i and j must be even, and the 2×2 region covered by the block must not overlap any previously placed block.

If `receive_block` ever returns a pair that does not satisfy these requirements, or if after placing the block a 2×2 square of cells becomes completely covered by black tiles, the grader will immediately terminate your program and the verdict for the test case will be `Output isn't correct`.

The behaviour of the grader is **not adaptive**. This means that the sequence of blocks Barchin gives to Charos is fixed before `init` is called.

Constraints

For each block, let S be the number of black tiles among its four tiles. That is, $S = TL + TR + BL + BR$.

- $1 \leq N, M \leq 100$
- $0 \leq S \leq 3$ for every block.

Subtasks

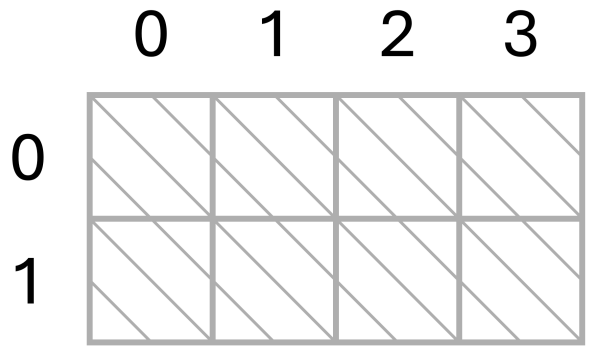
Subtask	Score	Additional Constraints
1	6	$S = 1$ for every block, and $N = 2$.
2	16	$S = 3$ for every block. $N = M$, N is even, and each of the four possible block colorings appears exactly $\frac{N^2}{4}$ times.
3	10	$S = 1$ for every block.
4	29	$S \leq 2$ for every block.
5	39	No additional constraints.

Example

Consider a game with $N = 1$ and $M = 2$, so the grid has 2 rows and 4 columns. The grader first calls:

```
init(1, 2)
```

Initially, all cells are empty. The grid looks like this:

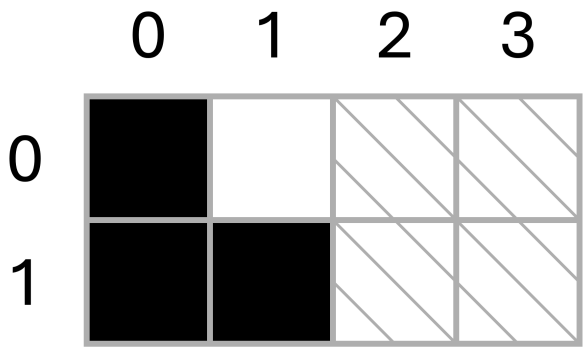


There are $N \cdot M = 2$ blocks to place. Suppose Barchin gives a block with three black tiles and one white tile at the top-right corner. The grader calls:

```
receive_block(1, 0, 1, 1)
```

Charos decides to place this block at the left side of the grid by returning $(0, 0)$.

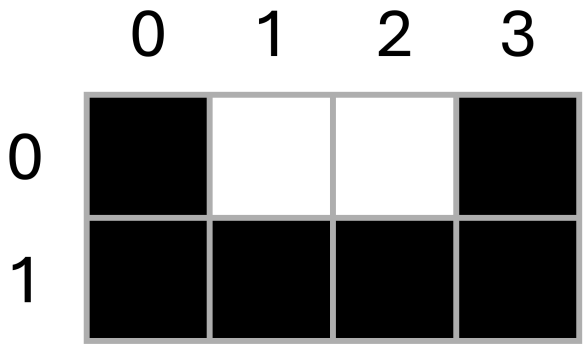
The grid now looks like this:



Barchin then gives another block with three black tiles and one white tile at the top-left corner:

```
receive_block(0, 1, 1, 1)
```

The only remaining cell with even row and even column that can serve as the top-left corner of a 2×2 block is $(0, 2)$, so Charos returns $(0, 2)$. The grid ends up like this:



No 2×2 square is completely covered by black tiles, so Charos has successfully placed all blocks without Barchin ever winning. Charos wins the game.

Sample Grader

Input format:

```
N M
TL[0] TR[0] BL[0] BR[0]
TL[1] TR[1] BL[1] BR[1]
...
TL[NM-1] TR[NM-1] BL[NM-1] BR[NM-1]
```

Output format:

```
R[0] C[0]
R[1] C[1]
...
R[NM-1] C[NM-1]
```

Here, $R[k]$ and $C[k]$ are the pair of integers returned by the k-th call to `receive_block`.