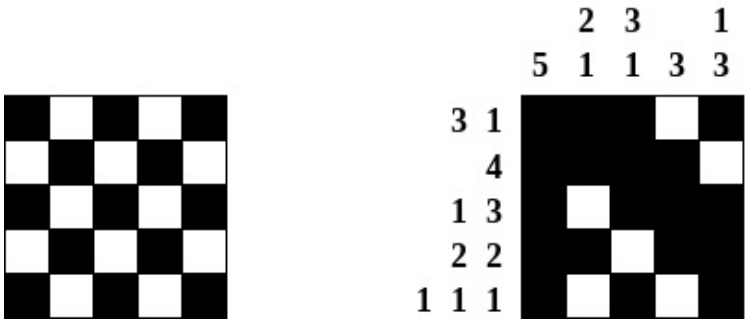
# 4. Chessboard Nonogram

There is a chessboard of size $N \times M$. The rows of the chessboard are numbered $0$ through $N-1$, and the columns are numbered $0$ through $M-1$. We refer to the cell in row $i$ ($0 \leq i \leq N-1$) and column $j$ ($0 \leq j \leq M-1$) as cell $(i, j)$. Initially cell $(i, j)$ is colored in black if $i + j$ is even, white otherwise.




Jinu additionally colored some of the white cells (possibly none) in the chessboard black. Then, Jinu made a nonogram puzzle from the chessboard. Specifically:

- For each row $i$ ($0 \leq i \leq N-1$), Jinu wrote the lengths of the runs of black cells on that row from left to right.
- For each column $j$ ($0 \leq j \leq M-1$), Jinu wrote the lengths of the runs of black cells on that column from top to bottom.

Write a program that recovers the chessboard from Jinu's puzzle.

## Implementation details

You should implement the following function. It will be called by the grader once for each test case.

```
int[][] SolveNonogram (int N, int M, int[][] Rclue, int[][] Cclue)
```

- $N$: number of rows of the chessboard.
- $M$: number of columns of the chessboard.
- $Rclue$: an array of integer arrays. For each $i$ ($0 \leq i \leq N-1$) and $k$ ($0 \leq k \leq |Rclue[i]| - 1$), $Rclue[i][k]$ is the length of the $(k+1)$-th run of black cells from the left in row $i$. Note that the size of $Rclue[i]$ can be different for each $i$.
- $Cclue$: an array of integer arrays. For each $j$ ($0 \leq j \leq M-1$) and $k$ ($0 \leq k \leq |Cclue[j]| - 1$), $Cclue[j][k]$ is the length of the $(k+1)$-th run of black cells

from the top in column $j$. Note that the size of $Cclue[j]$ can be different for each $j$.
- This function should return a two-dimensional $N$ by $M$ array of integers $P$. For each cell $(i, j)$, $P[i][j]$ should be 1 if $(i, j)$ is colored black, and 0 if $(i, j)$ is colored white.
- If there are multiple valid answers, this function should return any one of them.

## Constraints

- $1 \leq N \leq 100$
- $1 \leq M \leq 100$
- There exists at least one answer to the given puzzle.

## Subtasks

1. (25 points) $N = 1$
2. (75 points) No additional constraints.

## Example

Consider the following call:

```
SolveNonogram(5, 5,
              [[3, 1], [4], [1, 3], [2, 2], [1, 1, 1]],
              [[5], [2, 1], [3, 1], [3], [1, 3]])
```

One possible answer is:

```
[[1, 1, 1, 0, 1],
 [1, 1, 1, 1, 0],
 [1, 0, 1, 1, 1],
 [1, 1, 0, 1, 1],
 [1, 0, 1, 0, 1]]
```

## Sample grader

You can download the sample grader package on the same page you downloaded the problem statement. (scroll down if you don't see the attachment)

If you use IDEs like Visual Studio, Eclipse or Code::Blocks, then import `nonogram.cpp`, `nonogram.h` and `grader.cpp` into one project and you will be able to compile all these files at once.

If you want to compile by yourself, refer to the compilation commands in the statement

page.

You should submit only `nonogram.cpp`.

## Input format

- line 1: $N$ $M$
- line $2+i$ $(0 \leq i \leq N-1)$: $|Rclue[i]|$ $Rclue[i][0]$ $Rclue[i][1]$ $\cdots$ $Rclue[i][|RClue[i]|-1]$
- line $2+N+j$ $(0 \leq j \leq M-1)$: $|Cclue[j]|$ $Cclue[j][0]$ $Cclue[j][1]$ $\cdots$ $Cclue[j][|Cclue[j]|-1]$

## Output format

- line $1+i$ $(0 \leq i \leq N-1)$: $P[i][0]$ $P[i][1] \cdots P[i][M-1]$