Try to answer as much challenges as you can!

Submission: please add all solution on one RAR file include folder for each Challenge and submit it to **:**
http://www.evapharmacareers.com/Eva_Hackathon_Academy_2021.aspx

You can solve Challenge in any language.

Challenge 1 :

Given an input string (`s`) and a pattern (`p`), implement wildcard pattern matching with support for `'?'` and `'*'` where:

- `'?'` Matches any single character.
- `'*'` Matches any sequence of characters (including the empty sequence).

The matching should cover the **entire** input string (not partial).

Example 1:

```
Input: s = "aa", p = "a"

Output: false

Explanation: "a" does not match the entire string "aa".
```

Example 2:

```
Input: s = "aa", p = "*"

Output: true

Explanation: '*' matches any sequence.
```

Example 3:

```
Input: s = "cb", p = "?a"

Output: false

Explanation: '?' matches 'c', but the second letter is 'a', which does not match 'b'.
```

Example 4:

```
Input: s = "adceb", p = "*a*b"

Output: true

Explanation: The first '*' matches the empty sequence, while the second '*' matches the
substring "dce".
```

Example 5:

```
Input: s = "acdcb", p = "a*c?b"

Output: false
```

Constraints:

- `0 <= s.length, p.length <= 2000`
- `s` contains only lowercase English letters.
- `p` contains only lowercase English letters, `'?'` or `'*'`.

Challenge 2:

Given a string `s` representing a valid expression, implement a basic calculator to evaluate it, and return *the result of the evaluation*.

Note: You are **not** allowed to use any built-in function which evaluates strings as mathematical expressions, such as `eval()`.

Example 1:

```
Input: s = "1 + 1"

Output: 2
```

Example 2:

```
Input: s = " 2-1 + 2 "

Output: 3
```

Example 3:

```
Input: s = "(1+(4+5+2)-3)+(6+8)"

Output: 23
```

Constraints:

- `1 <= s.length <= 3 * 10`^{5}
- `s` consists of digits, `'+'`, `'-'`, `'('`, `')'`, and `' '`.
- `s` represents a valid expression.
- `'+'` is not used as a unary operation.
- `'-'` could be used as a unary operation but it has to be inside parentheses.
- There will be no two consecutive operators in the input.
- Every number and running calculation will fit in a signed 32-bit integer.

Challenge 3:

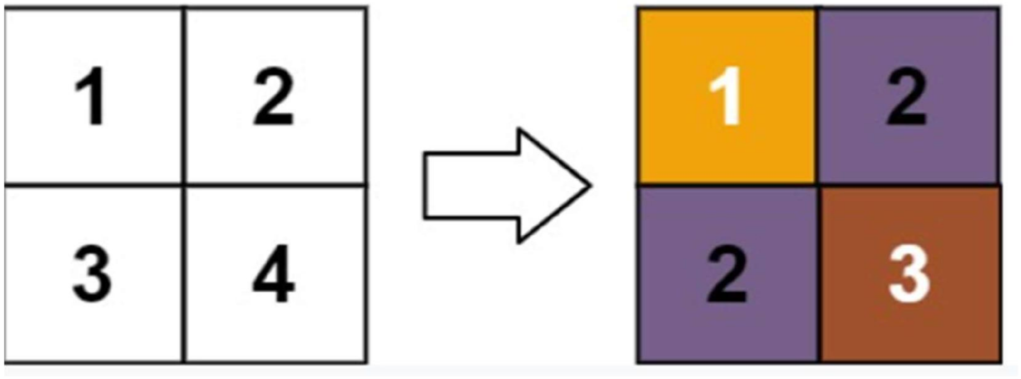
Given an m x n matrix, return *a new matrix* answer *where* answer[row][col] *is the rank of* matrix[row][col].

The rank is an integer that represents how large an element is compared to other elements. It is calculated using the following rules:

- The rank is an integer starting from 1.
- If two elements p and q are in the same row or column, then:
    - If p < q then rank(p) < rank(q)
    - If p == q then rank(p) == rank(q)
    - If p > q then rank(p) > rank(q)
- The rank should be as small as possible.

It is guaranteed that answer is unique under the given rules.

Example 1:



Input: matrix = [[1,2],[3,4]]

Output: [[1,2],[2,3]]

Explanation:

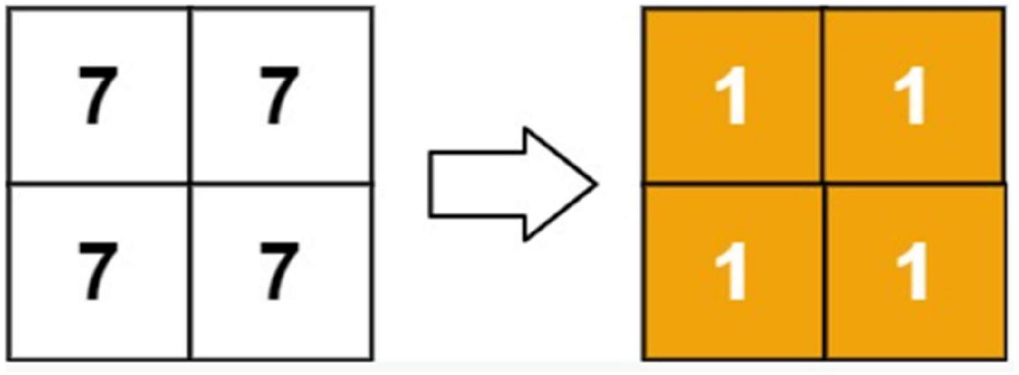
The rank of matrix[0][0] is 1 because it is the smallest integer in its row and column.

The rank of matrix[0][1] is 2 because matrix[0][1] > matrix[0][0] and matrix[0][0] is rank 1.

The rank of matrix[1][0] is 2 because matrix[1][0] > matrix[0][0] and matrix[0][0] is rank 1.

The rank of matrix[1][1] is 3 because matrix[1][1] > matrix[0][1], matrix[1][1] > matrix[1][0], and both matrix[0][1] and matrix[1][0] are rank 2.

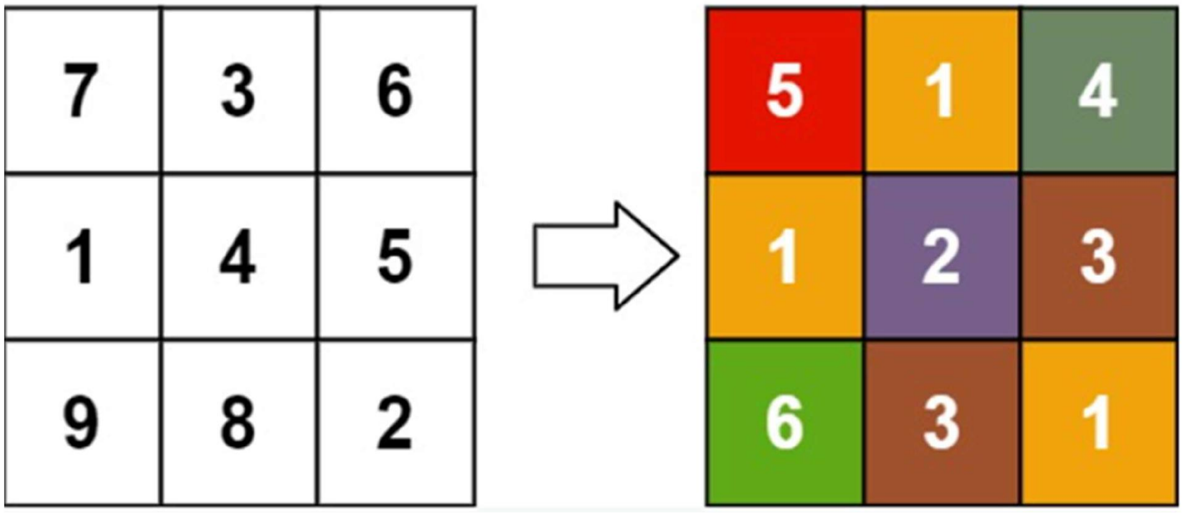
Example 2:



Input: matrix = [[7,7],[7,7]]

Output: [[1,1],[1,1]]

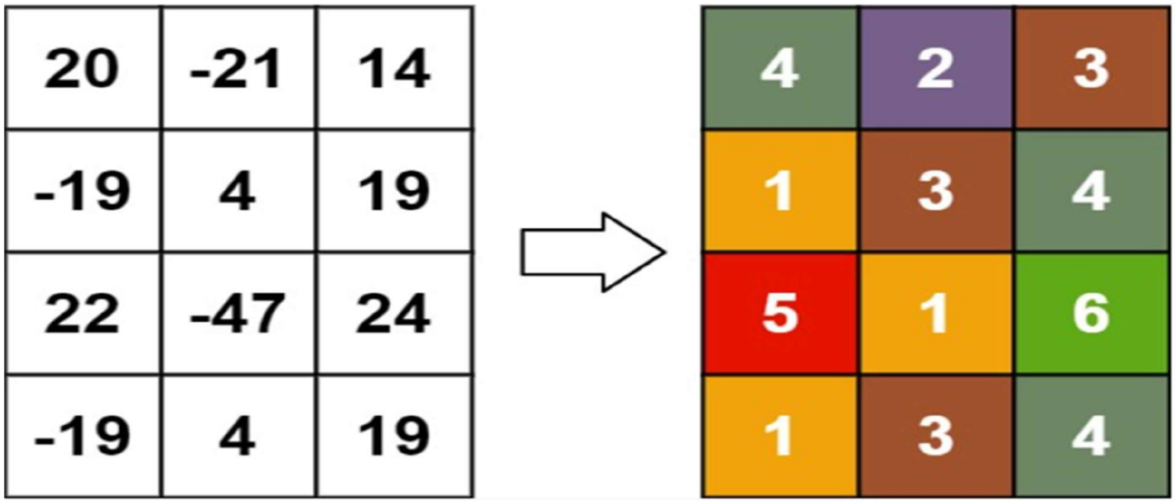
Example 3:



Input: matrix = [[7,3,6],[1,4,5],[9,8,2]]

Output: [[5,1,4],[1,2,3],[6,3,1]]

Example 4:



Input: matrix = [[20,-21,14],[-19,4,19],[22,-47,24],[-19,4,19]]

Output: [[4,2,3],[1,3,4],[5,1,6],[1,3,4]]

Constraints:

- m == matrix.length
- n == matrix[i].length
- 1 <= m, n <= 500
- -10^9 <= matrix[row][col] <= 10^9

Challenge 4 :

The demons had captured the princess and imprisoned her in **the bottom-right corner** of a `dungeon`. The `dungeon` consists of `m x n` rooms laid out in a 2D grid. Our valiant knight was initially positioned in **the top-left room** and must fight his way through `dungeon` to rescue the princess.

The knight has an initial health point represented by a positive integer. If at any point his health point drops to `0` or below, he dies immediately.

Some of the rooms are guarded by demons (represented by negative integers), so the knight loses health upon entering these rooms; other rooms are either empty (represented as 0) or contain magic orbs that increase the knight's health (represented by positive integers).

To reach the princess as quickly as possible, the knight decides to move only **rightward** or **downward** in each step.

Return *the knight's minimum initial health so that he can rescue the princess*.

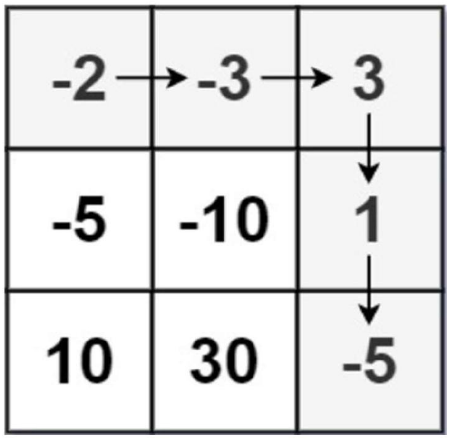
Note that any room can contain threats or power-ups, even the first room the knight enters and the bottom-right room where the princess is imprisoned.

Example 1:

```
Input: dungeon = [[0]]

Output: 1
```

Example 2:



```
Input: dungeon = [[-2,-3,3],[-5,-10,1],[10,30,-5]]

Output: 7
```

```
Explanation: The initial health of the knight must be at least 7 if he follows the
optimal path: RIGHT-> RIGHT -> DOWN -> DOWN.
```

Constraints:

- `m == dungeon.length`
- `n == dungeon[i].length`
- `1 <= m, n <= 200`
- `-1000 <= dungeon[i][j] <= 1000`

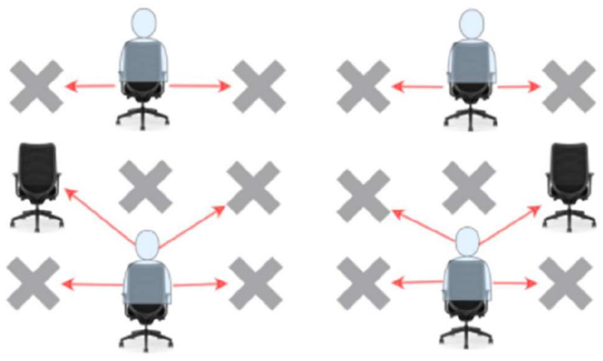
Challenge 5 :

Given a `m * n` matrix `seats` that represent seats distributions in a classroom. If a seat is broken, it is denoted by `'#'` character otherwise it is denoted by a `'.'` character.

Students can see the answers of those sitting next to the left, right, upper left and upper right, but he cannot see the answers of the student sitting directly in front or behind him. Return the **maximum** number of students that can take the exam together without any cheating being possible..

Students must be placed in seats in good condition.

Example 1:



```
Input: seats = [["#",".","#","#",".","#"],

                [".","#","#","#","#","."],

                ["#",".","#","#",".","#"]]

Output: 4

Explanation: Teacher can place 4 students in available seats so they don't cheat on the
exam.
```

Example 2:

```
Input: seats = [[".","#"],
                ["#","#"],
                ["#","."],
                ["#","#"],
                [".","#"]]
Output: 3
Explanation: Place all students in available seats.
```

Example 3:

```
Input: seats = [["#",".",".",".","#"],
                [".","#",".","#","."],
                [".",".","#",".","."],
                [".","#",".","#","."],
                ["#",".",".",".","#"]]
Output: 10
Explanation: Place students in available seats in column 1, 3 and 5.
```

Constraints:

- `seats` contains only characters `'.'` and `'#'.`
- `m == seats.length`
- `n == seats[i].length`
- `1 <= m <= 8`
- `1 <= n <= 8`

Deadline for attaching the answers Monday 30th of August
Good Luck

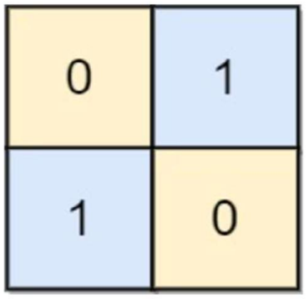
Challenge 6:

You are given an `n x n` binary grid `board`. In each move, you can swap any two rows with each other, or any two columns with each other.

Return *the minimum number of moves to transform the board into a* ***chessboard board***. If the task is impossible, return `-1`.

A **chessboard board** is a board where no `0`'s and no `1`'s are 4-directionally adjacent.

Example 1:

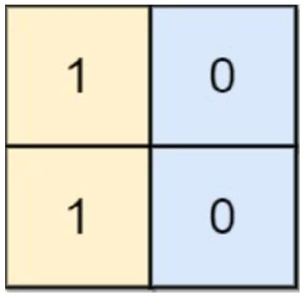


```
Input: board = [[0,1],[1,0]]

Output: 0

Explanation: Also note that the board with 0 in the top left corner, is also a valid
chessboard.
```

Example 2:

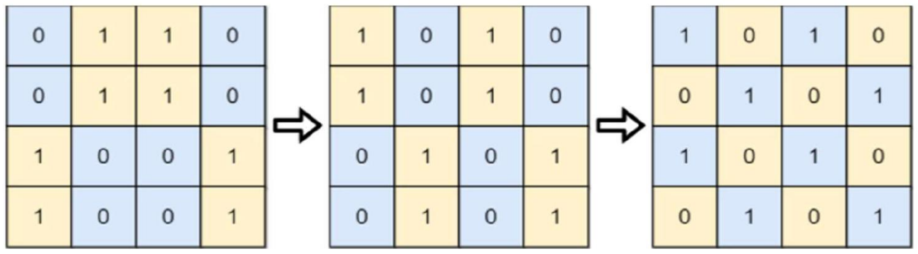


```
Input: board = [[1,0],[1,0]]

Output: -1

Explanation: No matter what sequence of moves you make, you cannot end with a valid
chessboard.
```

Example 3:



```
Input: board = [[0,1,1,0],[0,1,1,0],[1,0,0,1],[1,0,0,1]]

Output: 2

Explanation: One potential sequence of moves is shown.

The first move swaps the first and second column.

The second move swaps the second and third row.
```

Constraints:

- `n == board.length`
- `n == board[i].length`
- `2 <= n <= 30`
- `board[i][j]` is either `0` or `1`.

Deadline for attaching the answers Monday 30th of August
Good Luck