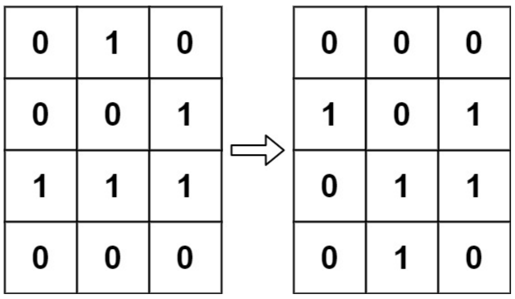
141."The Game of Life, also known simply as Life, is a cellular automaton devised by the British mathematician John Horton Conway in 1970." The board is made up of an m x n grid of cells, where each cell has an initial state: live (represented by a 1) or dead (represented by a 0). Each cell interacts with its eight neighbors (horizontal, vertical, diagonal) using the following four rules

Any live cell with fewer than two live neighbors dies as if caused by under-population.

1. Any live cell with two or three live neighbors lives on to the next generation.
2. Any live cell with more than three live neighbors dies, as if by over-population.
3. Any dead cell with exactly three live neighbors becomes a live cell, as if by reproduction.

The next state is created by applying the above rules simultaneously to every cell in the current state, where births and deaths occur simultaneously. Given the current state of the m x n grid board, return *the next state*.

Example 1:

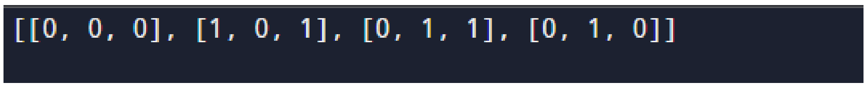


Input: board = [[0,1,0],[0,0,1],[1,1,1],[0,0,0]]
Output: [[0,0,0],[1,0,1],[0,1,1],[0,1,0]]

AIM: To solve the game of lime

PROGRAM:

```
  def gameOfLife(board):

if not board:

   return []



m = len(board)

n = len(board[0])

next_board = [[board[row][col] for col in range(n)] for row in range(m)]



directions = [(-1, -1), (-1, 0), (-1, 1), (0, -1), (0, 1), (1, -1), (1, 0), (1, 1)]

for i in range(m):
        for j in range(n):
            live_neighbors = 0
            for di, dj in directions:
                ni, nj = i + di, j + dj
                if 0 <= ni < m and 0 <= nj < n and board[ni][nj] == 1:
                    live_neighbors += 1
            if board[i][j] == 1:
                if live_neighbors < 2 or live_neighbors > 3:
                    next_board[i][j] = 0
            else:
                if live_neighbors == 3:
                    next_board[i][j] = 1

    return next_board
board = [[0,1,0],[0,0,1],[1,1,1],[0,0,0]]
print(gameOfLife(board))
```

OUTPUT: 

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

TIME COMPLEXITY: O(m*n)