EX-1.16

Title :

Compute the next state of the board in Conway’s Game of Life.

Aim:

To design and implement a Python program to compute the next state of the m x n grid board in the Game of Life based on given rules.

Procedure:

1. Read the input 2D board representing the current state with 0 (dead) and 1 (live).
2. Create a copy or use a method to simultaneously update the board without conflicting updates.
3. For each cell, count the number of live neighbors (up to 8 neighbors).
4. Apply the rules for each cell based on neighbors:
    - Live cell with fewer than 2 or more than 3 live neighbors → dies (0).
    - Live cell with 2 or 3 live neighbors → lives (1).
    - Dead cell with exactly 3 live neighbors → becomes live (1).
5. Update the board cells simultaneously after processing all cells.
6. Print or return the updated board.

Algorithm:

1. Start
2. Read board of size `m x n`.
3. Iterate each cell `(i, j)`:
    - Count live neighbors (consider boundaries).
    - Determine next state based on rules.
4. Store updates temporarily or encode next state using markers to apply simultaneously.
5. Update the original board to next state.
6. Return/print the updated board.
7. Stop

Input:

4 3

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

Program :

```
def gameOfLife(board):
    m, n = len(board), len(board[0])
    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 dx, dy in directions:
                x, y = i + dx, j + dy
                if 0 <= x < m and 0 <= y < n and board[x][y] in (1, 2):
                    live_neighbors += 1

            if board[i][j] == 1:
                if live_neighbors < 2 or live_neighbors > 3:
                    board[i][j] = 2  # live to dead
            else:
                if live_neighbors == 3:
                    board[i][j] = 3  # dead to live

    for i in range(m):
        for j in range(n):
```

```
            if board[i][j] == 2:
                board[i][j] = 0
            elif board[i][j] == 3:
                board[i][j] = 1

m, n = map(int, input("Enter m and n: ").split())
board = [list(map(int, input().split())) for _ in range(m)]

gameOfLife(board)

for row in board:
    print(' '.join(map(str, row)))
```

Performance Analysis:

Time Complexity: O(m*n)

Space Complexity: O(1)

program output:

```
File  Edit  Format  Run  Options  Window  Help
def game_of_life(board):
    m, n = len(board), len(board[0])

    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 dx, dy in directions:
                x, y = i + dx, j + dy
                if 0 <= x < m and 0 <= y < n and (board[x][y] == 1 or board[x][y] == 2):
                    live_neighbors += 1

            if board[i][j] == 1:
                if live_neighbors < 2 or live_neighbors > 3:
                    board[i][j] = 2
            else:
                if live_neighbors == 3:
                    board[i][j] = 3

    for i in range(m):
        for j in range(n):
            if board[i][j] == 2:
                board[i][j] = 0
            elif board[i][j] == 3:
                board[i][j] = 1

    return board

m = int(input("Enter number of rows (m): "))
n = int(input("Enter number of columns (n): "))
print(f"Enter board rows, each with {n} space-separated 0/1 values:")

board = []
for _ in range(m):
    row = list(map(int, input().split()))
    while len(row) != n:
        print(f"Please enter exactly {n} integers for this row.")
        row = list(map(int, input().split()))
    board.append(row)

result = game_of_life(board)
print("Next state of the board:")
for row in result:
    print(row)
```

```
IDLE Shell 3.13.5
File  Edit  Shell  Debug  Options  Window  Help
Python 3.13.5 (main, Jun 25 2025, 18:55:22) [GCC 14.2.0] on linux
Enter "help" below or click "Help" above for more information.
>>>
======= RESTART: /home/cyberkalai/Documents/DAA experiments files/exp1.16 ======
Enter number of rows (m): 4
Enter number of columns (n): 3
Enter board rows, each with 3 space-separated 0/1 values:
0 1 0
0 0 1
1 1 1
0 0 0
Next state of the board:
[0, 0, 0]
[1, 0, 1]
[0, 1, 1]
[0, 1, 0]
>>>
======= RESTART: /home/cyberkalai/Documents/DAA experiments files/exp1.16 ======
Enter number of rows (m): 2
Enter number of columns (n): 2
Enter board rows, each with 2 space-separated 0/1 values:
1 1
1 0
Next state of the board:
[1, 1]
[1, 1]
>>>
Ln: 27  Col: 0
```

Result :

Thus the given program Game of Life is executed and got output successfully.