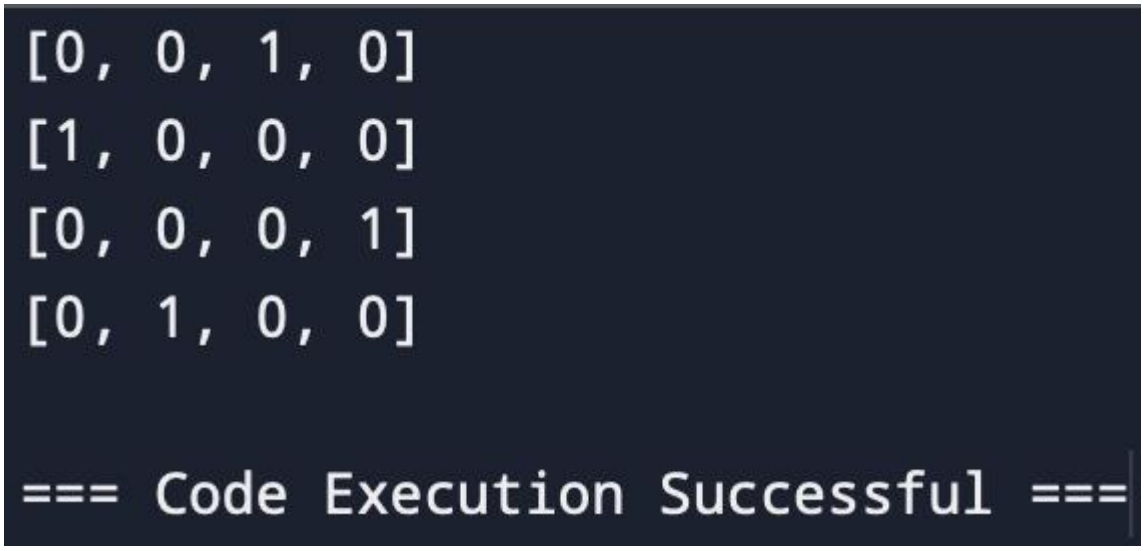
114. N-Queens Problem

PROGRAM:-

```
def is_safe(board, row, col, N):
    for i in range(col):
        if board[row][i] == 1:
            return False
    for i, j in zip(range(row, -1, -1), range(col, -1, -1)):
        if board[i][j] == 1:
            return False
    for i, j in zip(range(row, N, 1), range(col, -1, -1)):
        if board[i][j] == 1:
            return False
    return True

def solve_n_queens_util(board, col, N):
    if col >= N:
        return True
    for i in range(N):
        if is_safe(board, i, col, N):
            board[i][col] = 1
            if solve_n_queens_util(board, col + 1, N) == True:
                return True
            board[i][col] = 0
    return False

def solve_n_queens(N):
    board = [[0 for _ in range(N)] for _ in range(N)]
    if solve_n_queens_util(board, 0, N) == False:
        return False
    return board

def print_solution(board):
    for row in board:
        print(row)

# Example Usage
N = 4
solution = solve_n_queens(N)
if solution:
    print_solution(solution)
else:
    print("No solution exists for N = ", N)
```

OUTPUT:-



TIME COMPLEXITY:-O(n!)