Practical No:4

Code:

```
def is_safe(board, row, col, n):
  # Check if there is a queen in the same column
  for i in range(row):
    if board[i][col] == 1:
      return False

  # Check upper left diagonal
  for i, j in zip(range(row, -1, -1), range(col, -1, -1)):
    if board[i][j] == 1:
      return False

  # Check upper right diagonal
  for i, j in zip(range(row, -1, -1), range(col, n)):
    if board[i][j] == 1:
      return False

  return True

def solve_queens(board, row, n):
  if row == n:
    return True

  for col in range(n):
    if is_safe(board, row, col, n):
      board[row][col] = 1
      if solve_queens(board, row + 1, n):
        return True
      board[row][col] = 0
  return False
```

```
def n_queens_solution(n):
    board = [[0] * n for _ in range(n)]
    if solve_queens(board, 0, n):
        print("Solution found:")
        for row in board:
            print(row)
    else:
        print("No solution exists")
n_queens_solution(8)
```

Output:

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