DAA:- N queen

CODE:-

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

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

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

    return True

def solve_nqueens_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_nqueens_util(board, col + 1, n):
                return True

            # If placing the queen in board[i][col] doesn't lead to a solution
            # then remove the queen from board[i][col]
            board[i][col] = 0

    return False

def solve_nqueens(n):
    # Create an empty n x n chessboard
    board = [[0 for _ in range(n)] for _ in range(n)]

    if not solve_nqueens_util(board, 0, n):
        print("Solution does not exist")
        return False

    print_board(board)
    return True

def print_board(board):
    for row in board:
        print(" ".join(map(str, row)))

# Example usage for N=8
solve_nqueens(8)
```