```python
def is_safe(board, row, col):
    # Check for queens vertically up
    for i in range(row - 1, -1, -1):
        if board[i][col] == 'Q':
            return False

    # Check for queens on the diagonal left up
    for i in range(row - 1, -1, -1):
        j = col - (row - i)  # Calculate the column index for the
diagonal left
        if j >= 0 and board[i][j] == 'Q':
            return False

    # Check for queens on the diagonal right up
    for i in range(row - 1, -1, -1):
        j = col + (row - i)  # Calculate the column index for the
diagonal right
        if j < len(board) and board[i][j] == 'Q':
            return False

    return True

def n_queen(board, row):
    # Base case: if we've placed queens in all rows
    if row == len(board):
        print_board(board)  # Print the board configuration
        return 1  # Return 1 for one solution found

    count = 0  # Local counter for the number of solutions

    # Try placing queens in all columns for the current row
    for j in range(len(board)):
        if is_safe(board, row, j):
            board[row][j] = 'Q'  # Place queen
            count += n_queen(board, row + 1)  # Recurse to place next
queen
            board[row][j] = 'X'  # Backtrack and remove queen

    return count  # Return total solutions found

def print_board(board):
    print("-----------Chess Board-----------")
    for row in board:
        for cell in row:  # Using "cell" to represent each item on the
board
            print(cell, end=" ")  # Print each cell in the row
        print()  # Move to the next line after printing the row
    print()

if __name__ == "__main__":
```

```
    n = int(input("Enter the value of n for n-Queens problem: "))

    # Base cases for n=1, n=2, and n=3 where there are limited or no
solutions
    if n == 2 or n == 3:
        print(f"No solution exists for n = {n}")
    else:
        # Create an empty chess board of size n x n
        board = []
        for i in range(n):
            row = []  # Create an empty row
            for j in range(n):
                row.append('X')  # Fill the row with 'X'
            board.append(row)  # Add the row to the board

        # Solve the n-Queens problem and get the total number of
solutions
        total_solutions = n_queen(board, 0)
        print(f"Total ways to solve n Queens = {total_solutions}")
```

```
Enter the value of n for n-Queens problem:  4

-----------Chess Board-----------
X Q X X
X X X Q
Q X X X
X X Q X

-----------Chess Board-----------
X X Q X
Q X X X
X X X Q
X Q X X

Total ways to solve n Queens = 2
```