```
def print_solution(board):
    for row in board:
        print(" ".join("Q" if c else "." for c in row))

def is_safe(board, row, col, N):
    # Check left row
    for i in range(col):
        if board[row][i]:
            return False

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

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

    return True

def solve_nq_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_nq_util(board, col + 1, N):
                return True
            board[i][col] = 0  # BACKTRACK

    return False

def solve_n_queens(N):
    board = [[0] * N for _ in range(N)]
    if not solve_nq_util(board, 0, N):
        print("Solution does not exist.")
        return
    print_solution(board)

# Run the program
n = int(input("Enter number of queens (N): "))
solve_n_queens(n)
```

```
Enter number of queens (N): 5
Q . . . .
. . . Q .
. Q . . .
. . . . Q
. . Q . .
```

Start coding or generate with AI.