def is\_safe(board, row, col):

for i in range(col):

if board[row][i]: return False

for i, j in zip(range(row, -1, -1), range(col, -1, -1)):

if board[i][j]: return False

for i, j in zip(range(row, len(board)), range(col, -1, -1)):

if board[i][j]: return False

return True

def solve\_n\_queens\_util(board, col):

if col >= len(board): return True

for i in range(len(board)):

if is\_safe(board, i, col):

board[i][col] = True

if solve\_n\_queens\_util(board, col + 1): return True

board[i][col] = False

return False

def solve\_n\_queens(n):

board = [[False] \* n for \_ in range(n)]

if not solve\_n\_queens\_util(board, 0):

return "No solution exists"

return board

# Example usage

solution = solve\_n\_queens(8)

for row in solution:

print(' '.join('Q' if col else '.' for col in row))

