class NQueens:

def \_\_init\_\_(self, n):

self.n = n

self.board = [-1] \* n

self.solutions = []

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

for prev\_row in range(row):

prev\_col = self.board[prev\_row]

if col == prev\_col or abs(row - prev\_row) == abs(col - prev\_col):

return False

return True

def solve(self):

self.\_solve\_helper(0)

def \_solve\_helper(self, row):

if row == self.n:

solution = self.board.copy()

self.solutions.append(solution)

return

for col in range(self.n):

if self.is\_safe(row, col):

self.board[row] = col

self.\_solve\_helper(row + 1)

self.board[row] = -1

def display\_solutions(self):

print(f"Total solutions: {len(self.solutions)}")

for i, solution in enumerate(self.solutions):

print(f"Solution {i + 1}:")

for row in range(self.n):

row\_str = ""

for col in range(self.n):

if solution[row] == col:

row\_str += "Q "

else:

row\_str += ". "

print(row\_str.strip())

print()

# User-driven code

n = int(input("Enter the number of queens: "))

queens = NQueens(n)

queens.solve()

queens.display\_solutions()