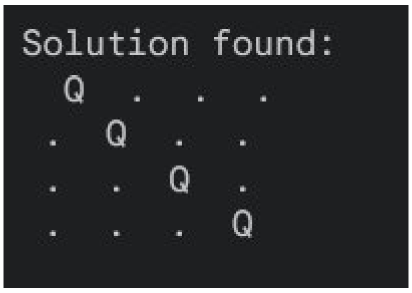
Sumedh ahire
FYMCA-B 03
BATCH 1
ASSIGNEMT 3

CODE:
```
def is_safe(row, col, board):
  """
  Checks if placing a queen at (row, col) is safe (no conflicts with existing queens).
  """
  # Check for queens in the same column above
  for i in range(row):
    if board[i][col] == 1:
      return False

  # Check for queens diagonally above
  i, j = row - 1, col - 1
  while i >= 0 and j >= 0:
    if board[i][j] == 1:
      return False
    i -= 1
    j -= 1

  # Check for queens diagonally below
  i, j = row - 1, col + 1
  while i >= 0 and j < len(board[0]):
    if board[i][j] == 1:
      return False
    i -= 1
    j += 1

  return True

def solve_n_queens(board, col):
  """
  Solves the n-queens problem using backtracking.
  """
  if col >= len(board[0]):
    return True  # All queens placed successfully

  for row in range(len(board)):
    if is_safe(row, col, board):
      board[row][col] = 1  # Place queen
      if solve_n_queens(board, col + 1):
        return True  # Backtrack if successful placement
      board[row][col] = 0  # Backtrack if placement leads to no solution

  return False  # No solution found for this configuration

def print_solution(board):
  """
  Prints the solution board with queens.
  """
  for row in board:
    for col in row:
      print(" Q " if col else " . ", end="")
    print()

# Create an empty board (4x4)
board = [[0 for _ in range(4)] for _ in range(4)]

if solve_n_queens(board, 0):
  print("Solution found:")
  print_solution(board)
else:
  print("No solution found")
```

OUTPUT:



```
Solution found:
  Q  .  .  .
 .  Q  .  .
 .  .  Q  .
 .  .  .  Q
```