**Experiment 2:**

**Aim:** To write a python program to implement a 8 queen program.

**Algorithm:**

1. Represent the chessboard as a list where the index is the row and the value is the column position of the queen.
2. Ensure no column or diagonal conflicts when placing a queen.
3. Use recursive backtracking to place queens row by row.
4. Backtrack to the previous row if no valid position is found.
5. Save and display solutions once all queens are successfully placed.

**Program:**

def print\_board(board):

"""Prints the chessboard."""

for row in board:

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

print()

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

"""Checks if a queen can be placed at board[row][col]."""

# Check this row on the left side

for i in range(col):

if board[row][i]:

return False

# Check upper diagonal on the 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 the left side

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

if board[i][j]:

return False

return True

def solve\_n\_queens(board, col, n, solutions):

"""Recursive utility to solve the N-Queens problem."""

if col >= n:

solutions.append([row[:] for row in board])

return True

res = False

for i in range(n):

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

board[i][col] = True

res = solve\_n\_queens(board, col + 1, n, solutions) or res

board[i][col] = False # Backtrack

return res

def n\_queens():

"""Main function to solve the N-Queens problem."""

n = int(input("Enter the size of the board (e.g., 8 for 8-Queens): "))

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

solutions = []

solve\_n\_queens(board, 0, n, solutions)

if solutions:

print(f"Found {len(solutions)} solution(s) for {n}-Queens:")

for solution in solutions:

print\_board(solution)

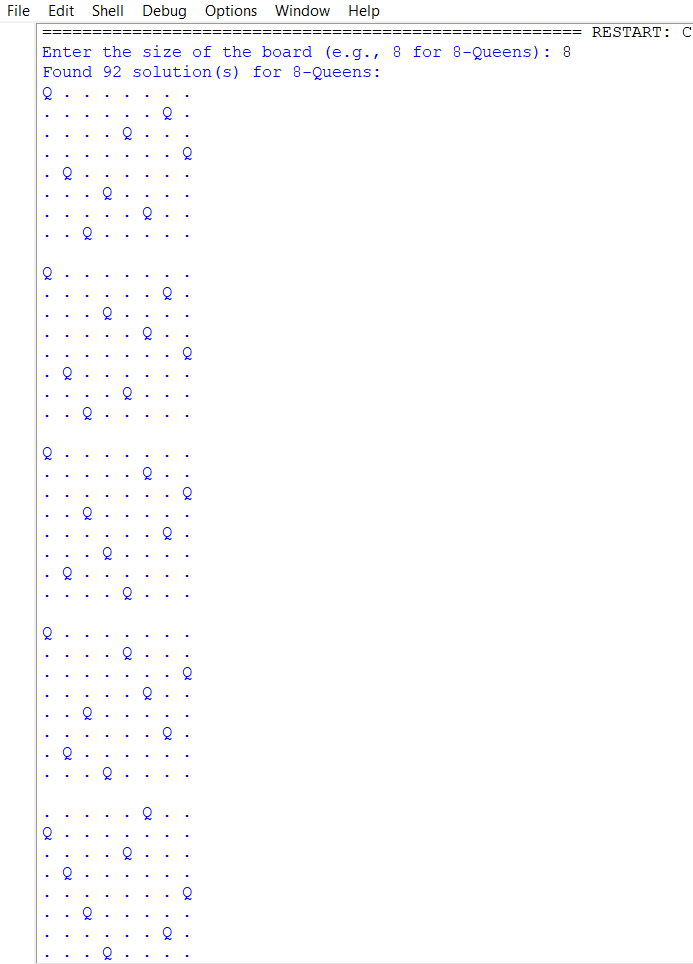
else:

print(f"No solution exists for {n}-Queens.")

if \_\_name\_\_ == "\_\_main\_\_":

n\_queens()

**Output:**



**Result:** Thus, the program was successfully completed using python programming.