ADITHIYA . V

241501010

Experiment 1

IMPLEMENT EIGHT QUEENS PROBLEM

Aim:

To develop a Python program that solves the 8-Queens problem using the Backtracking algorithm. The program should ensure that no two queens attack each other and display valid chessboard configurations.

Case Scenario:

A chessboard consists of 8×8 squares, and your task is to place 8 queens on the board such that no two queens attack each other. Queens can attack in horizontal, vertical, and diagonal directions.

Task Requirements:

1. Problem Representation:
o Represent the 8-queens problem as a constraint satisfaction problem (CSP) or a search problem.
2. Algorithm Implementation:
o Implement a solution using either Backtracking or Genetic Algorithm. 3. Output Requirements:
o Display a valid 8×8 chessboard with queens (Q) placed correctly.
o Show multiple valid solutions if possible.
4. Performance Analysis:
o Compare execution time for different board sizes (e.g., 4×4, 8×8, 10×10).

Procedure:

1. Start
2. Initialize an N×N chessboard with all empty positions (.).
3. Define a function is_safe(board, row, col, N):

🎬 Check if placing a queen at (row, col) violates any constraints.

4. Define a recursive function solve_n_queens(board, row, N):

🎬 If row == N, print the board (solution found).
🎬 Try placing a queen in each column (0 to N-1).
🎬 If is_safe() == True, place the queen and recurse for the next row.

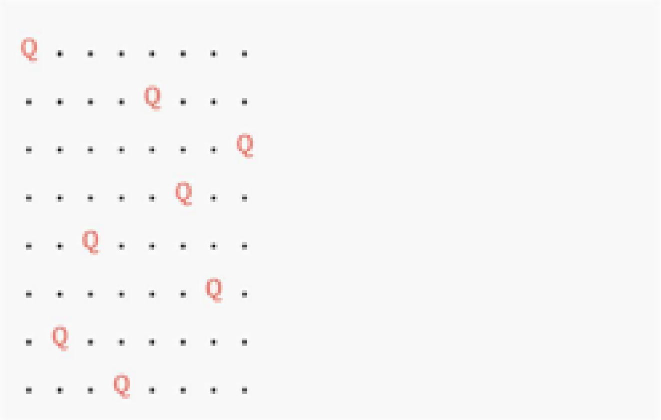
🎬 If placing a queen leads to failure, backtrack (remove the queen).

5. Call solve_n_queens() for the first row (row = 0).
6. If a solution is found, print the board; else, print "No solution exists."
7. End

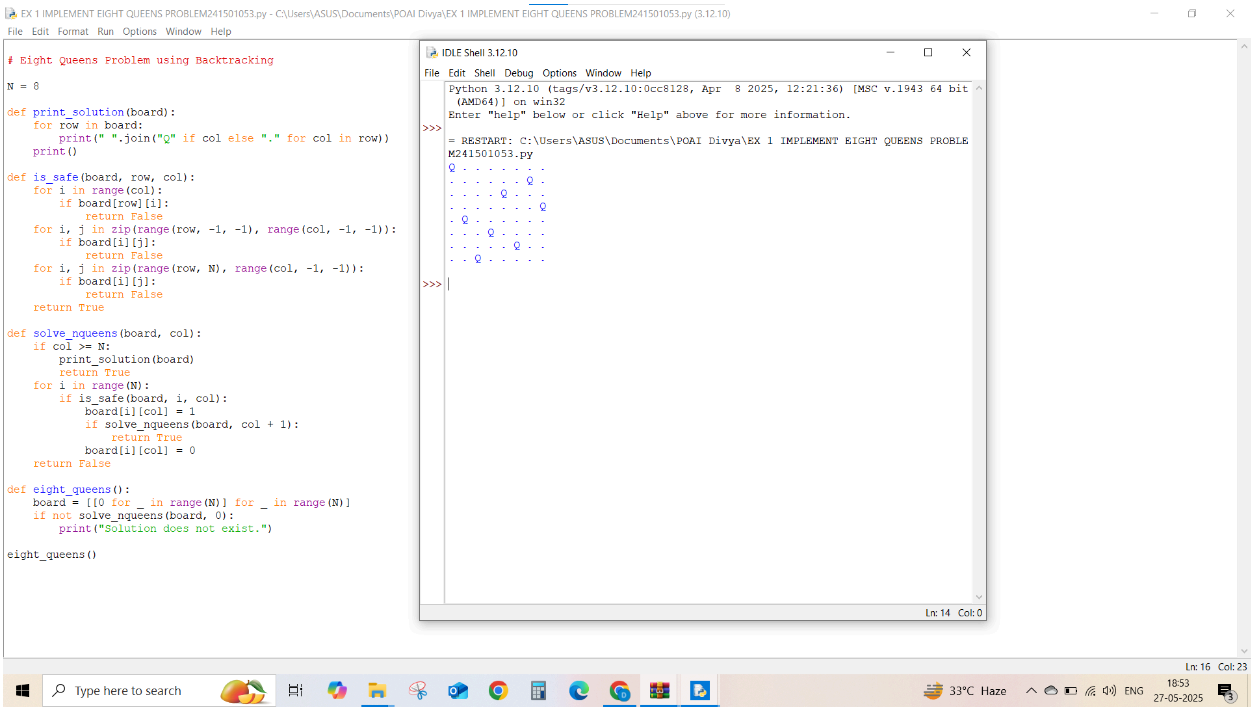
Program

```
import copy


N = 8 # Size of the chessboard (8x8)


# Function to print the solution
def printSolution(board):
 for row in board:
 for i in range(N):
 print("Q" if row[i] else ".", end=" ")
 print()
 print() # Add a newline for readability


# Function to check if a queen can be placed on
board[row][col]  def isSafe(board, row, col):
 # Check the column
  for i in range(row):
  if board[i][col]:
  return False


  # Check the upper left diagonal
  for i, j in zip(range(row - 1, -1, -1), range(col - 1, -1,
 -1)):   if board[i][j]:
  return False
 # Check the upper right diagonal
 for i, j in zip(range(row - 1, -1, -1), range(col + 1,
N)):   if board[i][j]:
 return False
 return True



 # Function to solve the 8 Queens problem using backtracking
 def solve(board, row, solutions):
  if row == N:
  solutions.append(copy.deepcopy(board)) # Deep copy of the
 board   printSolution(board)
  return


   for col in range(N):
   if isSafe(board, row, col):
   board[row][col] = 1 # Place queen
   solve(board, row + 1, solutions) # Recur to place next queen
   board[row][col] = 0 # Backtrack (remove queen)
  # Main function to initialize the board and start solving the problem
  def eightQueens():
   board = [[0 for _ in range(N)] for _ in range(N)]
   solutions = [] # Store all solutions
   solve(board, 0, solutions)
   print(f"Total solutions found: {len(solutions)}")
  # Calling the function
  eightQueens()
```

Output:



```
Q . . . . . . .
. . . . Q . . .
. . . . . . . Q
. . . . . Q . .
. . Q . . . . .
. . . . . . Q .
. Q . . . . . .
. . . Q . . . .
```



EX 1 IMPLEMENT EIGHT QUEENS PROBLEM241501053.py - C:\Users\ASUS\Documents\POAI Divya\EX 1 IMPLEMENT EIGHT QUEENS PROBLEM241501053.py (3.12.10)
File Edit Format Run Options Window Help
IDLE Shell 3.12.10
File Edit Shell Debug Options Window Help
Ln: 14 Col: 0
Ln: 16 Col: 23
Type here to search
33°C Haze
ENG
18:53
27-05-2025

```python
# Eight Queens Problem using Backtracking

N = 8

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

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, N), range(col, -1, -1)):
        if board[i][j]:
            return False
    return True

def solve_nqueens(board, col):
    if col >= N:
        print_solution(board)
        return True
    for i in range(N):
        if is_safe(board, i, col):
            board[i][col] = 1
            if solve_nqueens(board, col + 1):
                return True
            board[i][col] = 0
    return False

def eight_queens():
    board = [[0 for _ in range(N)] for _ in range(N)]
    if not solve_nqueens(board, 0):
        print("Solution does not exist.")

eight_queens()
```

```
Python 3.12.10 (tags/v3.12.10:0cc8128, Apr  8 2025, 12:21:36) [MSC v.1943 64 bit
 (AMD64)] on win32
Enter "help" below or click "Help" above for more information.
>>>
= RESTART: C:\Users\ASUS\Documents\POAI Divya\EX 1 IMPLEMENT EIGHT QUEENS PROBLE
M241501053.py
Q . . . . . . .
. . . . . . Q .
. . . . Q . . .
. . . . . . . Q
. Q . . . . . .
. . . Q . . . .
. . . . . Q . .
. . Q . . . . .

>>>
```