**AI Assignment 4**

**Neeti Kurulkar**

**N-Queen’s Problem**

**Code:**

**def print\_solution(board):**

**"""Function to print the chessboard solution."""**

**for row in board:**

**print(" ".*join*("Q" if cell else "\_" for cell in row))**

**print() # Print a newline for better readability**

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

**"""Function to check if a queen can be placed at board[row][col]."""**

**# Check the current column upwards**

**for i in range(row):**

**if board[i][col]:**

**return False**

**# Check upper-left diagonal**

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

**if board[i][j]:**

**return False**

**# Check upper-right diagonal**

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

**if board[i][j]:**

**return False**

**return True # If no conflicts, the position is safe**

**def solve\_n\_queens(board, row, N):**

**"""Recursive function to solve the N-Queens problem using backtracking."""**

**# Base case: If all queens are placed, print the solution**

**if row == N:**

**print\_solution(board)**

**return True # Return True to indicate a solution is found**

**found\_solution = False # To track if at least one solution exists**

**# Try placing a queen in each column of the current row**

**for col in range(N):**

**if is\_safe(board, row, col, N): # Check if placing is valid**

**board[row][col] = True # Place the queen**

**# Recur for the next row**

**if solve\_n\_queens(board, row + 1, N):**

**found\_solution = True # Mark that a solution is found**

**board[row][col] = False # Backtrack (remove the queen)**

**return found\_solution # Return True if at least one solution exists**

**# Take user input for board size**

**N = int(input("Enter the size of the chessboard (N): "))**

**if N < 10:**

**# Initialize an N x N board with False (no queens placed)**

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

**# Start solving from the first row**

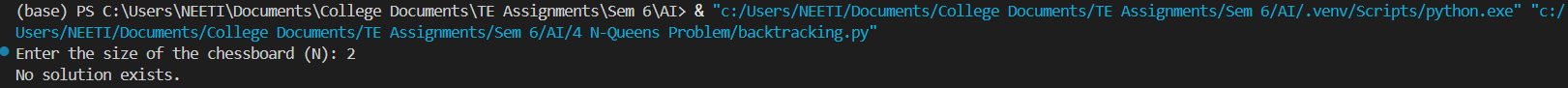
**if not solve\_n\_queens(board, 0, N):**

**print("No solution exists.")**

**else:**

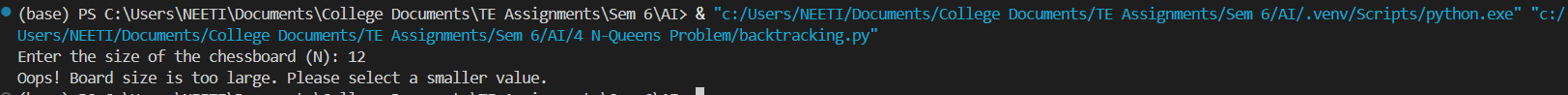
**print("Oops! Board size is too large. Please select a smaller value.")**

**Output:**

****

**A computer screen with blue and white text

AI-generated content may be incorrect.**

****

**A black background with blue and white lines

AI-generated content may be incorrect.**