```
# Name: Aniket Sarjerao Sable
# Roll No: 54

# ==============================================
# AI Practical 02-A
# Problem Statement:
# Write a program to solve the N-Queens Problem
# using the Hill Climbing algorithm as a heuristic-based local search technique
# ==============================================

import random

# ---------------------------------------------
# Function to check if a board configuration is valid (no queens attack each other)
# ---------------------------------------------
def is_valid(board):
    n = len(board)
    for i in range(n):
        for j in range(i + 1, n):
            if board[i] == board[j] or abs(board[i] - board[j]) == abs(i - j):
                return False
    return True

# ---------------------------------------------
# Heuristic function: number of conflicting pairs of queens
# ---------------------------------------------
def calculate_heuristic(board):
    n = len(board)
    conflicts = 0
    for i in range(n):
        for j in range(i + 1, n):
            if board[i] == board[j] or abs(board[i] - board[j]) == abs(i - j):
                conflicts += 1
    return conflicts

# ---------------------------------------------
# Hill Climbing Algorithm for N-Queens Problem
# ---------------------------------------------
def hill_climbing(n):
    current_board = [random.randint(0, n - 1) for _ in range(n)]  # Initial random board
    current_heuristic = calculate_heuristic(current_board)

    while current_heuristic > 0:
        found_better = False
        next_board = list(current_board)

        # Try moving each queen to every other row in its column
        for i in range(n):
            for j in range(n):
                if current_board[i] != j:
                    test_board = list(current_board)
                    test_board[i] = j
                    test_heuristic = calculate_heuristic(test_board)

                    if test_heuristic < current_heuristic:
                        next_board = test_board
                    current_heuristic = test_heuristic
                    found_better = True

        if not found_better:
            break  # No better neighbor found — local minimum

        current_board = next_board

    return current_board

# ---------------------------------------------
# Example Usage
# ---------------------------------------------
n = 6
solution = hill_climbing(n)

# Output the result
print("N-Queens Solution:", solution)
print("Is solution valid?", is_valid(solution))

# ---------------------------------------------
# Sample Output:
# N-Queens Solution: [1, 3, 5, 0, 2, 4]
# Is solution valid? True
# ---------------------------------------------
```