```
def count_conflicts(state):
    conflicts = 0
    n = len(state)
    for i in range(n):
        for j in range(i + 1, n):
            if state[i] == state[j]:
                conflicts += 1
            if abs(state[i] - state[j]) == abs(i - j):
                conflicts += 1
    return conflicts

def generate_neighbors(state):
    neighbors = []
    n = len(state)
    for i in range(n):
        for j in range(i + 1, n):
            neighbor = state[:]
            neighbor[i], neighbor[j] = neighbor[j], neighbor[i]  # Swap positions of queens i and j
            neighbors.append(neighbor)
    return neighbors

def hill_climbing(n, initial_state):
    state = initial_state
    while True:
        current_conflicts = count_conflicts(state)
        if current_conflicts == 0:
            return state
        neighbors = generate_neighbors(state)
        best_neighbor = None
        best_conflicts = float('inf')
        for neighbor in neighbors:
            conflicts = count_conflicts(neighbor)
            if conflicts < best_conflicts:
                best_conflicts = conflicts
                best_neighbor = neighbor
        if best_conflicts < current_conflicts:
            state = best_neighbor
        else:
            return None
print('Swapnil Sahil(1BM22CS300)')

def get_user_input(n):
    while True:
        try:
            user_input = input(f"Enter the row positions for the queens (space-separated integers between 0
            initial_state = list(map(int, user_input.split()))
            if len(initial_state) != n or any(x < 0 or x >= n for x in initial_state):
                print(f"Invalid input. Please enter exactly {n} integers between 0 and {n-1}.")
                continue
            return initial_state
        except ValueError:
            print(f"Invalid input. Please enter a list of {n} integers.")

n = 4
initial_state = get_user_input(n)

solution = hill_climbing(n, initial_state)

if solution:
    print("Solution found!")
    for row in range(n):
        board = ['Q' if col == solution[row] else '.' for col in range(n)]
        print(' '.join(board))
else:
    print("No solution found (stuck in local minimum).")
```

```
Swapnil Sahil(1BM22CS300)
Enter the row positions for the queens (space-separated integers between 0 and 3): 3 1 2 0
Solution found!
. Q . .
. . . Q
Q . . .
. . Q .
```