Implement Simulated Annealing to solve n − queens problem

```
import random
import math

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]
            neighbors.append(neighbor)
    return neighbors

def acceptance_probability(old_cost, new_cost, temperature):
    if new_cost < old_cost:
        return 1.0
    return math.exp((old_cost - new_cost) / temperature)

def simulated_annealing(n, initial_state, initial_temp, cooling_rate, max_iterations):
    state = initial_state
    current_cost = count_conflicts(state)
    temperature = initial_temp

    for iteration in range(max_iterations):
        neighbors = generate_neighbors(state)
        random_neighbor = random.choice(neighbors)
        new_cost = count_conflicts(random_neighbor)

        if acceptance_probability(current_cost, new_cost, temperature) > random.random():
            state = random_neighbor
            current_cost = new_cost

        temperature *= cooling_rate

        if current_cost == 0:
            return state
    return None

def get_user_input(n):
    while True:
        try:
            print("Output: 1BM22CS290")
            user_input = input(f"Enter the column positions for the queens (space-separated integers between 0 and {n-1}): ")
            initial_state = list(map(int, user_input.split()))
            for row in range(n):
              board = ['Q' if col == initial_state[row] else '.' for col in range(n)]
              print(' '.join(board))
            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 = 8
initial_state = get_user_input(n)

initial_temp = 1000
cooling_rate = 0.99
max_iterations = 10000
```

```
solution = simulated_annealing(n, initial_state, initial_temp, cooling_rate, max_iterations)

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 within the given iterations.")
```

```
Output: 1BM22CS290
Enter the column positions for the queens (space-separated integers between 0 and 7): 0 1 2 3 4 5 6 7
Q . . . . . . .
. Q . . . . . .
. . Q . . . . .
. . . Q . . . .
. . . . Q . . .
. . . . . Q . .
. . . . . . Q .
. . . . . . . Q
Solution found!
. . . . Q . . .
. . . . . . Q .
Q . . . . . . .
. . Q . . . . .
. . . . . . . Q
. . . . . Q . .
. . . Q . . . .
. Q . . . . . .
```