(1) SIMULATED ANNEALING FOR N-QUEENS CODE

INPUT-

```
import random
import math

# Function to generate an initial random state (placement of queens)
def initial_state(N):
    return [random.randint(0, N-1) for _ in range(N)]  # Random row positions for each column

# Function to compute the cost (number of conflicts) of a given state
def cost(state):
    conflicts = 0
    N = len(state)
    for i in range(N):
        for j in range(i + 1, N):
            # Check if queens share the same row or diagonal
            if state[i] == state[j] or abs(state[i] - state[j]) == j - i:
                conflicts += 1
    return conflicts

# Function to generate a neighbouring state by randomly moving one queen
def generate_neighbour(state):
    new_state = state[:]
    col = random.randint(0, len(state) - 1)
    new_row = random.randint(0, len(state) - 1)
    new_state[col] = new_row
    return new_state

# Simulated Annealing function
def simulated_annealing(N, initial_temp=1000, alpha=0.95, max_iter=10000):
```

```
    current_state = initial_state(N)
    current_cost = cost(current_state)
    temp = initial_temp
    iteration = 0

    while current_cost > 0 and iteration < max_iter:
        neighbour = generate_neighbour(current_state)
        neighbour_cost = cost(neighbour)

        delta_cost = neighbour_cost - current_cost

        # Accept the neighbour with probability depending on temperature
        if delta_cost < 0 or random.random() < math.exp(-delta_cost / temp):
            current_state = neighbour
            current_cost = neighbour_cost

        # Decrease temperature
        temp *= alpha
        iteration += 1

    return current_state, current_cost

# Function to print the solution as a matrix (chessboard representation)
def print_solution(state):
    N = len(state)
    board = [['.' for _ in range(N)] for _ in range(N)]

    # Place queens on the board (represented as 'Q')
    for col, row in enumerate(state):
        board[row][col] = 'Q'
```

```
    # Print the board
    for row in board:
        print(' '.join(row))

# Example usage
N =int(input("Enter the size of the board:"))  # 8-Queens problem
solution, cost_value = simulated_annealing(N)

if cost_value == 0:
    print(f"Solution found: {solution}")
    print_solution(solution)  # Print the solution as a matrix if found
else:
    print(f"No solution found. Final cost: {cost_value}")
```

OUTPUT-