United International **University (UIU)**
Dept. of Computer Science & Engineering (CSE)

Assignment-2
CSE 3812: Artificial Intelligence Lab
Total Marks: **10**

Assignment Title: Optimizing Delivery Routes Using Hill Climbing and Simulated Annealing

Assignment Instructions:

- Submit 3 files: 1 file containing the code for hill climbing, 1 file containing the code for simulated annealing, 1 pdf containing the rest of the requirements mentioned below.
- Don't submit a zip file.
- Rename the files as ***StudentId_hill_climb.py***

Scenario:
Imagine you work for a local courier company that must deliver packages to various clients scattered throughout a city. Each delivery location must be visited exactly once, and the starting point is the company's warehouse. Your goal is to find a delivery sequence that minimizes total travel distance. Perfect optimization is difficult, especially as the number of delivery points grows large, so heuristic methods are employed to find good (if not always perfect) solutions.

In this assignment, you will implement two heuristic search algorithms—Hill Climbing and Simulated Annealing—to approach the Traveling Salesman Problem (TSP). The assignment will help you understand the principles behind heuristic optimization and the trade-offs between greedy improvements and accepting occasional "bad" moves to escape local optima.

Task Description

1. Problem Definition:
    - You are given a set of delivery locations, each with its coordinates (e.g., latitude and longitude, or simplified x-y coordinates on a plane).
    - You must determine an order in which to visit all locations starting and ending at the warehouse.
    - Your objective is to minimize the total path length (the sum of the distances between consecutive stops).
2. Algorithms to Implement:
    - Hill Climbing:
        - Start with a random route (a permutation of the delivery points).

        - Consider small modifications (e.g., swapping two locations) to generate neighboring solutions.
        - If a neighbor improves the total distance, move to that neighbor. Repeat until no improving neighbor can be found.
    - Simulated Annealing:
        - Start with a random route (same as above).
        - Consider random modifications to generate neighboring solutions.
        - Accept any improving move.
        - Occasionally accept a worse solution based on a probability that decreases over time (the "temperature" parameter).
        - After a set number of iterations or when temperature is sufficiently low, stop and return the best found solution.
3. Distance Metric:
    - Use Euclidean distance or a simple "Manhattan" distance depending on the coordinate system provided.
    - For Euclidean: distance = sqrt((x2 - x1)^2 + (y2 - y1)^2).
4. Initial Solution:
    - Generate a random permutation of the delivery points. The warehouse (first location in the input) will be the start and end point.
5. Termination Criteria:
    - Hill Climbing: Stop when no improvement can be found after checking a reasonable set of neighbors.
    - Simulated Annealing: Stop after a fixed number of iterations or when the temperature approaches zero.
6. Comparison of Results:
    - Run both algorithms on the same input.
    - Report the best route distance found by each algorithm.
    - Discuss which approach found a better solution and why.

Input Format

- The first line: the number of locations `N` (including the warehouse).
- The next `N` lines: Each line contains two floating-point numbers (or integers) representing the x and y coordinates of a location.
- Note: The first location in the list is the warehouse. You must return to this point after visiting all others.

Example Input:

```
5

0.0 0.0         # Warehouse at (0,0)

2.0 3.0         # Client A at (2,3)
```

```
4.0 0.0         # Client B at (4,0)

4.0 3.0         # Client C at (4,3)

6.0 1.0         # Client D at (6,1)
```

This means there are 5 locations total: 1 warehouse + 4 clients.

Output Format

- Print the best route found and the corresponding total distance.
- For both algorithms, produce:
    - The order of locations visited (by their line index in the input file, starting from 1 for the warehouse).
    - The total travel distance.

Example Output:

```
=== Hill Climbing Result ===

Route: 1 -> 3 -> 5 -> 4 -> 2 -> 1

Total Distance: 15.27



=== Simulated Annealing Result ===

Route: 1 -> 2 -> 4 -> 5 -> 3 -> 1

Total Distance: 14.93
```

(Indices refer to input lines: 1=Warehouse, 2=Client A, 3=Client B, etc.)
