Assignment 0: Hill Climbing

In this assignment, you will implement a hill climbing (or hill descending) algorithm to optimize the first three DeJong functions. Your hill climber should demonstrate its ability to find optimal or near-optimal solutions for each function.

DeJong Functions

For this assignment, we will focus on minimizing the following three DeJong functions:

- **DeJong's Function #1 (Sphere Function):**
  - $$f(x) = \sum_{i=1}^{n} x_i^2$$
  - Typically evaluated in a 3-dimensional space ($n = 3$), with a search range of $[-5.12, 5.12]$ for each dimension. The global minimum is at $f(0, 0, 0) = 0$.
- **DeJong's Function #2 (Rosenbrock Function):**
  - $$f(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$$
  - Typically evaluated in a 2-dimensional space ($n = 2$), with a search range of $[-2.048, 2.048]$ for each dimension. The global minimum is at $f(1, 1) = 0$.
- **DeJong's Function #3 (Step Function):**
  - $$f(x) = \sum_{i=1}^{n} \text{floor}(x_i)$$
  - Typically evaluated in a 5-dimensional space ($n = 5$), with a search range of $[-5.12, 5.12]$ for each dimension. The global minimum is near $f(-5.12, ..., -5.12) = -25$.

Assignment Requirements

1. **Hill Climber Implementation:**
   - Implement a hill climbing algorithm. You will need to define:
     - **Initial Solution Generation:** How will you randomly generate a starting point in the search space?
     - **Neighbor Generation:** How will you generate a neighboring solution from the current solution? Consider different strategies (e.g., small random perturbation, fixed step size in each dimension).
     - **Acceptance Criteria:** For a basic hill climber, you will only accept better solutions.
     - **Termination Criteria:** When does your algorithm stop? (e.g., maximum number of iterations, no improvement after a certain number of steps).

2. **Function Implementations:**
    - Implement each of the three DeJong functions as separate methods or functions.
3. **Experimentation and Analysis:**
    - For each DeJong function, run your hill climber multiple times (e.g., 20-30 runs) with different random seeds.
    - Record the best solution found, the number of iterations taken, and the final function value for each run.
    - Calculate and report the average best solution, average number of iterations, and standard deviation for each function across all runs.
    - Discuss the performance of your hill climber on each function. Why might it perform better or worse on certain functions? Consider the landscape of each function (e.g., unimodal vs. multimodal, presence of flat regions).
4. **Report:**
    - Submit a clear and concise report detailing your implementation, experimentation, and analysis.
    - Include pseudo-code or a high-level description of your hill climbing algorithm.
    - Present your results in tables for clarity.
    - Discuss any challenges encountered and potential improvements to your hill climbing approach.

Submission Guidelines

Submit the following on the class canvas page.

- Assignment report, a technical report (pdf) that details your implementation (use pseudo code), experimentation, and analysis. We recommend LaTeX for technical report writing. Do not submit code.

Good luck!