1) Describe Local Search and Optimization:

Local search is a heuristic optimization technique used to solve complex optimization problems where the goal is to find the best solution (max or min) within a given solution space. It starts from a first solution and iteratively explores neighboring solutions, gradually moving towards the optimal solution. Local search algorithms dont guarantee finding the global optimum but aim to find a good solution quickly, making them suitable for large solution spaces where an exhaustive search is impractical.

The core of local search is:

- **Current Solution:** Start with an initial solution.
- **Neighbor Solutions:** Generate neighboring solutions by making small modifications to the current solution.
- **Objective Function:** Evaluate the quality of each solution using an objective function (a cost or fitness function).
- **Move Selection:** Choose a neighboring solution that improves the objective function (or meets specific criteria).
- **Termination Condition:** Continue searching until a termination condition is met (e.g., a maximum number of iterations, a time limit, or no further improvement).

Local search algorithms include Hill Climbing, Simulated Annealing, Genetic Algorithms, and more. These techniques differ in their strategies for selecting and exploring neighboring solutions.

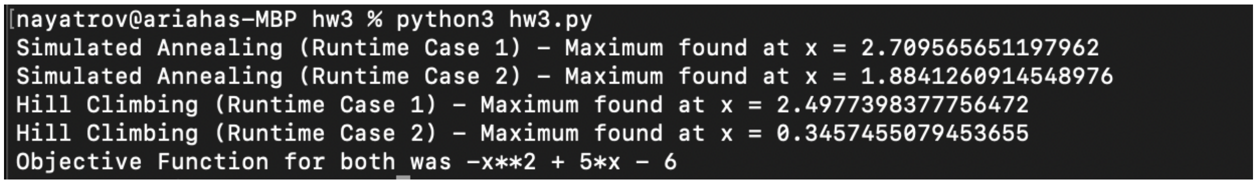
2) Describe Hill Climbing with examples. Write algorithms for hill climbing. (Algorithm Included in hw3.py) (Output included at the bottom of this document)

As discussed above, Hill Climbing is a local search algorithm used to find the local optimum (max or min) of an objective function within a solution space. It starts from an initial solution and iteratively makes small adjustments to the current solution, moving in the direction of steepest ascent (for maximization) or steepest descent (for minimization). Hill Climbing can get stuck in local optima and does not guarantee finding the global optimum.

3) Describe Simulated Annealing with examples. Write algorithms for simulated annealing. (Algorithm Included in hw3.py) (Output included at the bottom of this document)

As briefly discussed in question 1, Simulated Annealing is a probabilistic optimization algorithm inspired by the annealing process in metallurgy. It is used to find the global optimum (max / min) of a cost or objective function within a given solution space. Unlike Hill Climbing, Simulated Annealing can escape local optima by allowing for uphill moves with a decreasing probability as the algorithm progresses. It is particularly useful when dealing with complex, non-convex, and multimodal objective functions.

4) Write a python program to implement Hill Climbing and Simulated annealing with two different runtime cases. (Algorithm Included in hw3.py) (Output included below)


```
[nayatrov@ariahas-MBP hw3 % python3 hw3.py
Simulated Annealing (Runtime Case 1) - Maximum found at x = 2.709565651197962
Simulated Annealing (Runtime Case 2) - Maximum found at x = 1.8841260914548976
Hill Climbing (Runtime Case 1) - Maximum found at x = 2.4977398377756472
Hill Climbing (Runtime Case 2) - Maximum found at x = 0.3457455079453655
Objective Function for both was -x**2 + 5*x - 6
```