**project2Phase1.py**

The file contains functions designed to read geographic data from a file named miles.txt, process and store this data in an appropriate data structure (a dictionary), and provide utilities for retrieving and manipulating this data.

**Key Components**

1. **Data Loading Functions:**
   * loadCityData(): Reads city names and states from the miles.txt file and returns a list of these names.
   * loadInformation(): Reads coordinates, population data, and distances between cities from the miles.txt file and returns three lists: coordinateList, populationList, and distanceList.
   * listAllInformation(): Combines city data into a comprehensive dictionary called cityDataDict, which includes coordinates, population, and distance information for each city.

**Dictionary Structure:**

1. Keys: City names combined with state codes (e.g., "Wilmington DE").
2. Values: Lists containing coordinates, population, and a nested dictionary of distances to other cities.

Example entry: "Wilmington DE": [[3975, 7555], 70195, distanceDictionary]

1. **Data Access and Manipulation Functions:**
   1. getCoordinates(cityDataDict, cityName): Returns the coordinates (latitude and longitude) of a given city.
   2. getPopulation(cityDataDict, cityName): Returns the population of a given city.
   3. getDistance(cityDataDict, cityName1, cityName2): Returns the distance between two cities. If the cities are the same, it returns 0. If either city is not in the dictionary, it returns None.
   4. nearbyCities(cityDataDict, cityName, r): Returns a list of cities that are within a distance of r radius from the given city.

**project2Phase2.py**

**Overview**

The file contains various functions aimed at reading geographic data, processing it, and implementing both a greedy and a brute force algorithm to solve the facility location problem. The goal is to determine the fewest number of facilities needed so that every city is within a specified radius r of a facility.

**Key Components**

1. **Data Loading Functions:**
   * loadCityData(): Reads city names and states from the miles.txt file.
   * loadInformation(): Loads coordinates, population data, and distances between cities from the miles.txt file.
   * listAllInformation(): Combines city data into a comprehensive dictionary, including coordinates, population, and distance information for each city.
2. **Greedy Algorithm:**
   * Implements a greedy approach to solve the facility location problem by repeatedly selecting the city that serves the most unserved cities within the radius r.
3. **Brute Force Algorithm:**
   * **Helper Functions:**
     + actualSolution(cityDataDictionary, cityList, r): Checks if a given set of cities can serve all cities within the radius r.
     + generateAllSolutions(cityList, k): Generates all possible combinations of cities of size k.
     + generate\_All\_Solutions\_Of\_1\_to\_K(cityList, k): Generates all possible combinations of cities from size 1 to k.
   * **Main Function:**
     + optimalFacilitySet(cityDataDictionary, r, oneSolution): Finds the optimal set of facilities by evaluating all possible combinations of city placements and selecting the smallest set that serves all cities.