A4 – Visualizing and Analyzing Big Data Sets

Background

- You are given daily temperatures for 210 U.S. cities, dating back 50 – 100+ years, and ending at the end of 2021.
- Each city's data is in its own file.
- The file "city_info.csv" is the information directory where you can find the file matching each city, as well as the date range of the temperature information for that city.

Part A – Consolidate the Data for a Set of Cities and a Date Range

1. Write the function *consolidateData(cities, startDate, endDate, outputFileName)*.

   The function should create a single file containing the average daily temperatures, in Celsius, for each city for each date in the date range. (Note that the source files contain high and low temperatures, in Fahrenheit. So, you should average those two numbers together and convert the average to Celsius.)
   - The function should abort if it **<u>any</u>** data is missing – i.e., if a high or low temperature is missing for any city on any date. (In other words, we want a complete data set.)
   - The first line of the output file should be the column headers "CITY,DATE,TEMPERATURE".
   - The remainder of the file should be sorted first by city (alphabetically ascending), then within each city sorted by date (from earliest to latest).
   - Make sure there are no leading or trailing spaces.
   - Make sure the dates are in YYYY-MM-DD format (as in the source files).

2. Use your *consolidateData* function to create a file containing temperatures for 4 cities and at least 20 years. Note that you might have to experiment with the choice of cities and dates because of missing data. Keep experimenting until you get a complete set.
3. Write a helper script to prove that your data set is complete. The number of lines in the file should be (C * YN * 365) + (C * YL * 366) + 1, where C is the number of cities, and YN is the number of regular years and YL is the number of leap years.
4. Hand in your source code, a sample file, and a screen capture of a script run proving that the file is complete.

Part B – Create a Helper class

1. To make the data easier to work with, and to avoid having to re-read the file every time we want to access the data, create a helper class, called **TemperatureHelper**, that:

   - loads the data from a consolidated file (created in Part A) when an instance is constructed
   - separates the dates into year, month, and day
   - stores the data as nested dictionaries, where:
     - the city name is the key for level 1, and the value is a dictionary of years
     - the year is the key for level 2, and the value is a dictionary of months
     - the month is the key for level 3, and the value is a dictionary of days
     - the day is the key for level 4, and the value is the temperature.
     - **Example**: if a consolidated file contains data for Chicago for all dates in 1995, and if the helper class stores this in a field called `data`, then:
       - `data` is a dictionary
       - `data['Chicago']` is a dictionary with the years as keys
       - `data['Chicago'][1995]` is a dictionary with the months as keys
       - `data['Chicago'][1995][2]` is a dictionary with the days as keys
       - `data['Chicago'][1995][2][28]` is an integer (the temperature)
   - Provides a method, *getDailyTemperature(city, year, month, day)*. This method returns the temperature for the specified city and date.
   - Provides a method, *getYearlyTemperatures(city, year)*. This method returns a NumPy array containing the temperatures for the specified city for each day of the specified year.
     - You will have to import NumPy.
     - A NumPy array is similar to a list, but it's easier to perform mathematical operations on it. We'll use it later.

2. Write a script that tests the two functions with a few different calls each.
3. Hand in your source code, your test script, and a screen capture showing the testing.

Part C – Create Graphs

1. Learn to use Matplotlib (https://matplotlib.org/) and write the following functions.

```
# Create and save one graph per city, each showing the
# annual mean temperatures for that city.
def annualMeansPerCity(cities, years):
    # your code here
    pass # remove me
```

```
# Create and save a single graph showing the annual means
# for all the cities, one series per city.
# Use the specified patterns for the legend.
def annualMeansCombined(cities, years, patterns):
    # your code here
    pass # remove me
```

```
# Create and save one graph per city, each showing the
# temperature on the specified day for each specified year.
def singleDayPerCity(cities, years, month, day):
    # your code here
    pass # remove me
```

```
# Create and save a single graph showing the temperatures on
# the specified day for the specified years, one series per city.
# Use the specified patterns for the legend.
def singleDayCombined(cities, years, month, day, patterns):
    # your code here
    pass # remove me
```

2. Choose four dates and use your data set to create charts showing:
    a. the temperatures on day1 for each city (four charts)
    b. the temperatures on day2 for each city (four charts)
    c. the temperatures on day3 for each city (four charts)
    d. the temperatures on day4 for each city (four charts)
    e. the temperatures on day1 for all cities on a single chart
    f. the temperatures on day2 for all cities on a single chart
    g. the temperatures on day3 for all cities on a single chart
    h. the temperatures on day4 for all cities on a single chart
    i. the annual means per city, one chart per city (four charts)
    j. the annual means of all cities as four series on a single chart
3. Hand in your source code and your 25 charts.