OA2801: Computational Methods I **Handout #**
Prof. Alderson / MAJ Schuchard **November 2019**

Lab 6

The Scenario: You are working as an analyst at the Pentagon. Earlier today, a two-star stopped by and asked your commanding officer a few questions about the location of airports in the Eastern hemisphere relative to the position of recent earthquakes. (The media is currently buzzing about the logistics surrounding humanitarian relief efforts that have occurred at high elevations, and there is speculation that the DoD may be asked to comment.)

You have been asked to work with the following data:

- a list of airports in the Eastern hemisphere, named `easternAirports.csv`
- a list of "target" locations, representing recent earthquake sites, named `targets2019.csv`

You have been tasked with the responsibility to answer the following questions:

1. What are the `N` highest elevation airports in the Eastern hemisphere?
2. For ***each*** target location, what are the `N` nearest airports, and what are their elevations?

Task 0: As you have learned from your OA training, the first step in a problem like this is to become familiar with the data of interest.

Airport Locations: The file `easternAirports.csv` contains information approximately 2400 airports in the Eastern hemisphere. Data for each airport includes the following:

- Field 01 - ICAO Code: 4 character ICAO code
- Field 02 - IATA Code: 3 character IATA code
- Field 03 - Airport Name: string of varying length
- Field 04 - City,Town or Suburb: string of varying length
- Field 05 - Country: string of varying length
- Field 06 - Latitude Degrees: 2 ASCII characters representing one numeric value
- Field 07 - Latitude Minutes: 2 ASCII characters representing one numeric value
- Field 08 - Latitude Seconds: 2 ASCII characters representing one numeric value
- Field 09 - Latitude Direction: 1 ASCII character either N or S representing compass direction
- Field 10 - Longitude Degrees: 2 ASCII characters representing one numeric value
- Field 11 - Longitude Minutes: 2 ASCII characters representing one numeric value
- Field 12 - Longitude Seconds: 2 ASCII characters representing one numeric value
- Field 13 - Longitude Direction: 1 ASCII character either E or W representing compass direction
- Field 14 - Altitude: varying sequence of ASCII characters representing a numeric value corresponding to the airport's altitude from mean sea level (ie: "123" or "-123")

The data for each airport is contained on a single line, with successive fields delimited by commas.

Also, this file contains comments (text that should be ignored as code). Specifically, each line beginning with a '#' is a comment. Lines of data that start with this symbol should be ignored when reading these input files. **In addition to ignoring comments, you may need your program to "clean" this data and ensure that it is represented in the correct format in order to apply a distance or other calculation.**

Target Locations: Your instructions are to consider specific target locations, representing the sites of recent earthquake activity. Each location is specified by a (latitude,longitude) pair, measured in ***degrees***. The file `targets2019.csv` contains information about 15 target locations.

Your Task: Write a Python program called `ParseDisasters.py` that takes as inputs two files (airports and targets) and a desired number of airports to report (an integer N) and provides the following information (program outputs) in an easily read format:

- A list of the of the N highest airports in the provided airport list, sorted in decreasing order of elevation, with all information available about each airport printed to the console in an easily read format.
- For each of the target locations provided in the target file, provide a list of the N nearest airports (sorted with nearest airport printed first). Each target location should be clearly identified, and its corresponding list should include all information available about each airport, with everything printed to the console in an easily read format.
- Your program should also provide a print statement at the end of program execution that reports the Big-O complexity of your program.
- The program should take three inputs as command line arguments in the following order:
  - **The name of the airport file.** You may assume the format of the airport table will always be the same but that the length of the file (number of rows) is likely to change as airports are opened or closed.
  - **The name of the target file.** You may assume the format of the target table will always be the same but that the length of the files (number of rows of data) is likely to change as more earthquakes/disasters either occur or are simulated.
  - **The parameter N (an integer),** representing the number of nearest airports desired to be reported.

  Note: I will test your program by running it with different command-line arguments, and by looking at what is printed to the console.

Note: You may (should) use the great circle distance approach developed during Lab 1 but be wary of formatting differences between the various data sources and references.

Helpful Hints:

This assignment is highly modular. It is strongly recommended that you break the task down into (combinations of) functions that can be tested/debugged and used by yourself and others for other similar tasks in the future. Useful modules would be:

- Reading/formatting the airports file into an appropriate data structure (what data structure will you choose?)
- Reading/formatting the targets file into an appropriate data structure (what data structure will you choose?)
- Conversion of lat/lon units to an appropriate format (you've seen this before)
- Calculating the distance between two points on the globe (you've seen this before)
- Printing the first and/or last N values in a sorted data structure (list or dictionary; implementation depends on data structure)

Assignment Due: 2300 Wednesday, Nov 27

Lab Submission: Submit your lab files via Sakai. Include with your submission both the source code (named `Yourlastname_ParseDisasters.py`) as well as a text file containing your output (named `Yourlastname_Lab6_output.txt`) containing the output for the case when N = 5. Include in comments at the top of your code a statement that clearly documents the Big-O complexity of your program.