Where in the world is CS?

Data cleansing and data validation are two important workaday tasks in data science and computer science more generally, whether in dealing with large collections of inconsistently formatted data, data scraped from the web, or just trying to make users' experiences as pleasant and trouble-free as possible by not requiring strict formats. This étude introduces the delights of this kind of work.

The primary problem is to process individual strings that are supposed to represent a labelled *horizontal coordinate* on the earth, i.e., a latitude and a longitude, and to represent them in a particular standard form within the GeoJSON framework. You are notionally producing a tool that takes such a string and tries to convert it into that standard form.

The standard form for the output will be a feature entry (see below) containing a pair of numbers for the latitude and longitude each with exactly six digits after the decimal point separated by a comma and a single space. The first number represents the latitude and should be between -90.000000 and 90.000000. Positive values represent locations in the northern hemisphere and negative ones in the southern hemisphere (0.000000 represents the equator). Except for points whose latitude is in the range -1 to 1 there should not be leading 0's. The second number represents the longitude and should be between -180.000000 and 180.000000 with positive values denoting locations in the eastern hemisphere (east of the prime meridian) and negative ones denoting locations in the western hemisphere. As for latitude, there should not be leading 0's except when required.

The expectation is that each line *should* consist of a coordinate (in some form), optionally followed by a label. For instance:

```
45.9 S, 170.5 E Dunedin
 23.6 W, -50.00
```

Your aim should be to translate as much input as possible into standard form and display it on a map. I'm not going to tell you all the things you should be able to deal with, but here are some that you should definitely be able to handle:

- Standard form.
- Standard form except the number of decimal points given differs from 6.
- Standard form except the comma is missing.
- Standard form, except the numbers are non-negative and followed by N or S (for latitude) and E or W (for longitude), possibly in the wrong order.
- "Degrees, minutes, seconds" form with or without decimal places on the seconds, and with or without the standard markers for degrees, minutes and seconds.
- Degrees and decimal minutes form.

Remember that the objective is to be as inclusive as possible. If, knowing that something is supposed to represent a horizontal coordinate on the earth you can immediately see what that coordinate should be, then your program should be able to do so as well.

Recommended reading

- Geographic coordinate systems
- Geographic coordinate conversion – just the part on change of units and format.
- GeoJSON specification

Task

Provide a program that reads a sequence of lines from `stdin` and, where possible, converts them into GeoJSON features (with names if given) and writes those to a valid GeoJSON file that contains a `featurecollection`. If you are given input that is impossible to interpret this way, e.g., `Ha, tricked you!` then write `Unable to process:` followed by the offending input line. You should visualise the saved GeoJSON file on a map using a tool of your choice. You are free to select the way you want to implement the map visualisation, but a few options are given below. As well as the source code, you should include screenshots and a description of your approach in your report for this étude.

Recommended options

- Leaflet Javascript library
- Mapbox Javascript library
- Python geomapping library
- Tool for viewing map data supporting GeoJSON

Relates to Objectives

1.1, 1.2, 1.3, 1.4, 2.2, 2.3, 2.7, 2.8, 3.4, 3.5, 4.1, 4.5

(Individual)