UNBC Laboratory Assignment

Computer Science 100—Fall 2011

Converting Co-ordinates

Purpose:

To practise using file input and output,together with classes.

Due Date:

The completed lab assignment is due Monday 2011-11-21 *at the beginning of lecture*.

A Conversion Problem

Frequently, small programs to convert data from one format to another are incredibly useful. Although the details are not accurate, the following problem is similar to one encountered by BC Forest fire-fighters who collect GPS data in format, and need to convert it to a different form for use with mapping software.

The input data format

Input data are longitudes and latitudes (similar to those Lab 3) stored in a text file. Each line of the file consists of one longitude, followed by a space, followed by one latitude. Both the longitudes and the latitudes are in decimal degrees, with positve numbers denoting East longitudes and North latitudes, and negative numbers denoting West longitudes or South latitudes. An example file is shown in Figure 2.

sample-in.txt

```
135.0990  65.0915
 24.6912 -39.6077
-44.7398  68.4376
 57.0247  -7.5715
 72.2593  74.3285
-66.7411  68.2415
```

Figure 1: Sample input data

The output data format

The output data are also longitudes and latitudes, but in a slightly odder format.

Longitudes are written in a format like `135E5.9’` or `66W44.5’`, that is, as an integral number of degrees, followed by a direction letter (`E` or `W`), followed by a number of minutes (there are sixty minutes to a degree) that is accurate to one decimal place.

Latitudes are written in a similar format, but with `N` or `S` for the direction letter. A longitude-latitude pair is written in parentheses.

The data in Figure **??** correspond to those in Figure 2.

sample-out.txt

```
(135E 5.9’,65N 5.5’)
( 24E41.5’,39S36.5’)
( 44W44.4’,68N26.3’)
( 57E 1.5’, 7S34.3’)
( 72E15.6’,74N19.7’)
( 66W44.5’,68N14.5’)
```

Figure 2: Sample input data

Programming tasks

⇒ Write a [`static`] method (possibly with several sub-methods) that takes two arguments, an input file name, and an output file name, and that then reads longitude-latitude pairs from the input file and writes the re-formatted pairs to the output file.

⇒ Write a `main`-method that tests your program on the files `"sample1-in.txt` and `"sample2-in.txt` found on the blackboard site.

Take two

⇒ Write a `LonLatPair` *class* that has the following methods:

- a `LonLatPair` constructor (with two double arguments),
- a `readFromScanner` method that reads a longitude and latitude from a `Scanner` and uses them to set the internal data,
- a `boolean isNorth` method that returns `true` iff the latitude is > 0,
- a `boolean isEast` method that returns `true` iff the longitude is > 0,
- a `String`-producing `format2()` method that produces a `String` line one line of the output file above.

⇒ Rewrite your program above to take advantage of the `LonLatPair` class.

Hand in

- listings for all of your programs; and
- test runs of both programs for both test data files.