

Lab 11

Weather stats

10 April 2025

This lab will give you continued practice working with structs and with vectors and with vectors of structs (and writing functions for them), and additionally will involve reading in data rather than solely hand-building the vectors and structs in a test suite.

1. To get started on this part of the lab, copy some files into your directory. The task of the week will involve processing weather data, so first, you'll copy a set of weather data, found in

```
/home/shared/160/weather-big.txt  
/home/shared/160/weather-small.txt
```

into your own working directory for this lab.

2. Look at (but don't change) the contents of the files, then continue reading for their description.

These are data about the weather for the month of August 2009 in Galesburg, IL (where I worked at the time). Each line of the file contains information about the date and time it represents, as well as the temperature (in degrees Fahrenheit) and the wind speed (in miles per hour). This is a sample line illustrating the format:

```
8 2 2009      18 00   76      12
```

The date is first, then the time: this line represents August 2, 2009, at 18:00 (6pm). The temperature was 76°F, and the wind was blowing at 12mph. The “big” file contains measurements for every hour that month; the “small” file contains two days. (You're also encouraged to create your own files, even smaller, for testing purposes, but make sure they follow the same format.)

3. Based on the description above, and using the examples from class (and the struct you wrote for last week's lab), define a **struct** called **Weather** that is capable of holding all the data on each line of our data file. This definition should go in a file **Weather.h** (which will also be where the related function headers will eventually go).

4. Create a file `test_Weather.u`, make a test suite with only (for now) a `fixture` section (see below if you don't know or don't remember where that goes), and in that section make at least two examples of `Weather` objects, and one `vector` that contains those two `Weather` objects.
5. Compile it! With just the `.h` and the fixture in the `.u` there is enough to compile. No tests to run yet, but you can verify if your syntax is correct.
6. Add a function header to `Weather.h` for a function named `ever_below_55` that will determine whether the temperature in a given vector of weather values ever dipped below 55 degrees.
7. Create a file `Weather.cpp` with a stub for the function you designed in the previous step.
8. Go back to the file `test_Weather.u` and add a `tests` section to your suite, with tests for the `ever_below_55` function. Use the vector that you already put in the fixture, and add another vector so that you can more thoroughly test the function.
9. Compile again—note that your compile line will need to include both the `.u` file and the newly made `Weather.cpp`, or you'll get a linker error. (Don't forget to update the readme!) Remember that it's ok that the tests are failing now (and if they're not, they're broken tests, and you should resolve that).
10. Finish writing the function.

Boilerplate that makes use of fixtures (or, just put all the examples directly into test blocks):

```
//any necessary #include

test suite some_name
{
  fixture:
    type var = value; // any number of examples to be used in test cases

  tests:
    test any_name
    {
      // any needed c++ code
      check ( value-or-expression ) expect == value; // OR...
      check ( value-or-expression ) expect true; // or false
    }

    //add additional test blocks here, as many as necessary/reasonable
}
```

This document was written and prepared without the use of generative AI.