COMP161 - Lab 4 & Homework 4

Spring 2016

This week you'll be building basic interfaces for the functions you developed last week. You're welcome to use the instructor's solutions to last week's lab and homework as a basis for this week's assignments if you need or want to.

The Programs

During lab you'll get started on building the following programs:

1. A program that provides a basic CLI interface to the charge payback function from lab/hwk 2.
   The CLI program takes a single command line argument, the charge amount. The executable should be named *payback*. When the user enters a negative charge amount, the program should print an error message and close without reporting a payback amount. Otherwise, the program should behave like this example:

   ```
   $> ./payback 250.0
   $250.0 will payback $0.625
   $>
   ```

2. A program that provides a basic REPL interface for some of the distance conversion functions from lab/hwk2.
   The executable program should be named *convert*. The program should prompt the user for three things: the distance to be converted, its units, and the target units. Your program should only allow the conversion to and from inches, centimeters, and feet. Units can be specified by a single letter[1]: i for inches, c for centimeters, and f for feet. If the user enters a negative value for the distance or a unit not covered by the program, then the program should write an error message to the standard error and continue to the next loop without attempting a conversion. Otherwise, the program should produce descriptive output. A execution of the program that terminates after a single loop might look like this:

   [1] i.e. a char value

   ```
   $> ./convert
   What would you like to convert: 12 i
   What's the target unit: f
   12i = 1f
   $>
   ```

Lab 4

Before you dive into writing the actual implementations for main you should get the basic boilerplate for each program setup and ready to go.

1. Stub out two files, one for each *main* // This means creating two files, adding the appropriate comments at the top, add the appropriate includes, and stubbing out *main* in each file.
2. Add rules to your Makefile for building each of the executable programs

When these steps are complete you should be able to build three executable programs using your Makefile: the gTest program that runs your tests from last week and two programs that do nothing. Once again, your goal is to quickly get a working foundation upon which you can build your program(s).

Once you've got your stubbed out programs ready to go you should work on both programs incrementally. Write code to do just a little bit, then compile and test that code. Each of these programs is built up from a collection of discrete tasks. Your job is to identify those tasks so that you can complete the programs in an ITERATIVE fashion. Rather than write "all" the code then debug "all" the code, your process should allow you to write some of the code then debug it and repeat until "all" the code is done.

When the lab period is over, submit your source documents and your Makefile as *lab4* using handin[2]. I'll be looking for evidence of iterative development. I shouldn't see one nearly complete program and one stub of a program but should see two in-progress programs each with a comparable amount of progress.

[2] Be certain to clean out your working directory of non-source files before submitting

Homework 4

Due by 8am on Monday 2/7

Complete both programs described under lab 4. Submit the source code and Makefile as *hwk4* using handin[3].

[3] Be certain to clean out your working directory of non-source files before submitting