CSci 1113: Introduction to C/C++
Programming for Scientists and Engineers
Homework 2
Spring 2017

Due Date: Friday, Feb. 17, 2016 before 6:00pm.

Purpose: In this program you get the chance to write more C++ programs. These program will involve some of the basic C++ constructs we have studied so far, including arithmetic operations and more complicated use of branching statements.

Instructions: This is an individual homework assignment. There are two problems worth 20 points each. Solve each problem below by yourself (unlike the labs, where you work collaboratively), and submit each solution as a separate C++ source code file. Here are a few more important reminders:

1. Unlike the computer lab exercises, this is **not a collaborative assignment**. You must design, implement, and test the solution to each problem on your own without the assistance of anyone other than the course instructor or TAs. In addition, you may not include solutions or portions of solutions obtained from any source other than those provided in class: examples from the textbook, lectures, or code you and your partner write to solve lab problems. Otherwise obtaining or providing solutions to any homework problems for this class is considered academic misconduct. See the “collaboration rules” file on the class Moodle page for more details, and ask the instructor if you have questions.
2. Because all **homework assignments are submitted and tested electronically**, the following are **important**:
   - You follow any naming conventions mentioned in the homework instructions.
   - You submit the correct file(s) through Moodle by the due deadline.
   - You follow the example input and output formats given in each problem description.
   - **Regardless of how or where you develop your solutions, your programs compile and execute on cselabs computers running the Linux operating system.**
3. The problem descriptions will usually show at least one test case and the resulting correct output. However, you should test your program on other test cases (that you make up) as well. Making up good test cases is a valuable programming skill, and is part of ensuring your code solution is correct.

Problem Introduction

While many measurement systems use a simple linear scale, others are, at least in part, cyclic. For example, two hours later than 11PM is not 13PM, it is 1AM. As a second example, the amount of money obtained by adding 76 cents and 48 cents is usually not written as 124 cents, but instead as 1 dollar and 24 cents.

This homework involves longitude, which is one way to represent east-west positions on the Earth. Longitude can be represented in a variety of ways. In this problem we will use a representation consisting of three values: degrees, an integer between 0 and 180, inclusive; minutes, an integer between 0 and 59, inclusive; and a direction, a character which can be either ‘E’ or ‘W’. For example, the longitude of Minneapolis is approximately 93 degrees, 16 minutes W. Suppose you start at one location and then move a certain number of degrees and minutes either east or west. What is the longitude of your new location? For example, if you start at 172 degrees 0 minutes W and move 10 degrees 0 minutes west your new longitude would be 178 degrees 0 minutes E. This homework asks you to write a program that has a user input an initial longitude and a change in longitude. It should then compute and output the new longitude. Because the entire program is a little complicated, this homework has you write the program in two parts.

Problem A: No Minutes (20 points)
In this part you work only with degrees and direction, and not minutes. Specifically, write a C++ program that asks a user to input an original longitude, plus a change in longitude. The original longitude should be in terms of degrees (but not minutes) and whether the direction is 'E' or 'W'. The change should just be a nonnegative number, that you can assume is in the same direction as the original longitude. The program should then calculate and output the new longitude, using the input/output format shown in the following example:

```
Original longitude degrees and direction: 172 W
Longitude change degrees: 10
178 degrees E
```

Here are some further clarifications, notes, and hints:

- You may assume the original longitude input is a valid longitude. That is, you do not need to check that the original values input are in the valid ranges. And you may assume, without checking, that the change degrees are nonnegative.
- However, do not make any further assumptions on the value of the change degrees. For example, a value of 720 is allowed (and corresponds to going around the world twice).
- The output longitude should always be between 0 degrees and 180 degrees, inclusive, regardless of whether the direction is east or west.
- Note that for 0 degrees, the direction is unnecessary. The same holds true for 180 degrees. So in inputting these as the original longitude or a change in longitude, the user may enter either direction. However, if your programs outputs either of these it should write them without the direction, for example just 180 degrees.
- Follow the input and output formats shown in the example above.
- C++ hint: you will likely find the remainder operator % useful.
- Start this problem soon. This Problem A program should not be difficult, but is not trivial either. Moreover, Problem B builds on Problem A. So you will need to give yourself enough time not only to get your Problem A code working correctly, but also to complete the more difficult Problem B.

Test your program on a variety of input to ensure it handles the different cases correctly. When you have written, tested, and corrected your solution, save your file as <username>_2A.cpp, where you replace <username> with your U of M username. As usual, be diligent in following this naming convention. Submission details are at the end of part two, as you will need to post both parts together.

Problem B: Including Minutes (20 points)
This is a challenging problem. Its purpose is to give you experience with complicated programming logic. Expect this problem to be longer and more difficult than the previous problems in this class.

To begin, copy your solution from Problem A. Name the copy <username>_2B.cpp, where you replace <username> with your U of M username. In this part **make sure you modify the copy, not your original Problem A file**.

This part introduces two requirements beyond those in Problem A:

1. Your program should include minutes. Specifically the user input should include minutes of the original longitude, and minutes of the change in longitude. And the new longitude calculation and output should include the minutes as well.
2. The change in longitude should specify not only degrees and minutes, but also a direction. For the change, the number of degrees can be any nonnegative integer (as in Part A, it does not need to be between 0 and 180), the minutes can be any integer between 0 and 59 inclusive, and the direction can be either ‘W’ or ‘E’.
   Example: if you start at 93 degree 16 minutes W and move 10 degrees 26 minutes E, your new longitude will be 82 degrees 50 minutes W.

Other than the changes due to these two requirements, your program should behave the same as in Part A. Here are a few additional clarifications:

- You may again assume without checking that the original longitude input is a valid longitude. And you may assume without checking that the change is longitude is a valid change, with the change being positive (degrees nonnegative, minutes between 0 and 59 inclusive, and direction being ‘E’ or ‘W’).
- The output longitude should always be between 0 degrees 0 minutes and 180 degrees 0 minutes, inclusive, regardless of whether the direction is east or west. So, for example, suppose the user inputs an original longitude of 175 degrees 0 minutes E, and a change of 5 degrees 25 minutes E. The output should be 179 degrees 35 minutes W, not 180 degrees 25 minutes E.
- The number of minutes output should always be between 0 and 59, inclusive.
- Follow the input and output formats shown in the example below.
- Similar to Problem A, for 0 degree 0 minutes and for 180 degrees 0 minutes, the direction is unnecessary and should not be output.
- Problem solving hint: while this problem might seem easy at first, the logic is nontrivial. This program is intended to be challenging. Start this problem early, and think (before writing code) of what steps your solution will need to perform, and in what order. Your program will need to handle a number of different situations, so make sure you understand the requirements and the solution logic carefully before writing code.
- If you cannot get the entire program to work, get as much of it to work as you can. Specifically, try to get the minutes working correctly even if you do not get the change direction requirement implemented. Turning in a program that compiles and handles some of the possible cases correctly will give you more points than turning in nothing, or turning in a program that does not compile.

Here is an example:

```
Original longitude degrees, minutes, and direction: 93 16 W
Longitude change degrees, minutes, and direction: 10 26 E
New longitude: 82 degrees 50 minutes W
```

Test your program on a variety of input to ensure it handles the different cases correctly. When you have written, tested, and corrected your solution, save your file as <username>_2B.cpp, where you replace <username> with your U of M username. Upload both part A and part B files (that is, two separate CPP files) using the Homework 2 link in Moodle. It will allow you to upload at most two files. As usual, be diligent in following this naming convention.