CSci 1113: Introduction to C/C++, Spring 2019
Programming for Scientists and Engineers
Homework 1

Due Date: Monday, Feb. 11, 2019 before 5:30pm.

Purpose: Welcome to Homework 1. Its purpose is give you a chance to write C++ programs that solve problems requiring C++ arithmetic operators, simple input and output, different variable types, and selection statements (i.e., `if` statements). In doing these problems you should get practice both with these C++ constructs, and with computational problem-solving in general.

Instructions: This is an individual homework assignment. There are two problems worth 30 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 reminders about a few more important details:

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 website 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 the class website 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. Here are some hints on this assignment:

   - Start early. One common problem in this class is students waiting until soon before the assignment is due, and then running out of time. By the time this homework is posted you will have seen enough C++ in lecture and the textbook reading to begin the problems here.
   - For each problem, make sure you understand it and design a step-by-step solution before starting to write C++ code.
   - Ask questions during office hours as needed. Also, if you have questions for office hours, remember there is usually office hour congestion on the day when the assignment is due; so go to office hours on days prior to the due date.
   - Homework problems are usually based on material from the previous week's lecture and labs. So you do not need to wait until after the current week's lab to start the homework.

4. Please make your source code easy to read. In particular, start each file with comments giving the program name, your name, and the date you completed the program. And use good coding style (as discussed in lecture and the textbook) to ensure that the graders can understand your code.

5. 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.

6. Have fun! Solving computational problems and writing code can be difficult and, at times, frustrating. But it can also be fun and satisfying.

Problem A. Color Components, Once Again (30 points)

Recall that colors are often represented by (r, g, b) values, where the coordinates are sometimes ints and sometimes doubles. In this problem, use ints for the color coordinates. That is, each of r, g, and b can take on any integer value between 0 and 255, inclusive.

There are a variety of effects that are performed on images. Sometimes the effects are done for artistic reasons, (e.g., enhancing the contrast in a photograph), sometimes for analysis (e.g., filtering out certain colors in a satellite image), sometimes for other reasons.

Write a C++ program that does the following: First, it should ask the user to input the red, green, and blue components. You may assume each of these is between 0 and 255 — you do not need to check for bad input. Then the program should ask the user to choose one of these effects:

- Enhance contrast. Replace the red component r by the value

$$(127.5)\left[\sin\left(\left(\frac{r}{255.0} - .5\right)\pi\right)\right] + 127.5$$

  and then round it to the nearest integer. This will move lower values for r closer to 0, and higher ones closer to 255. Do the same for the green and blue components.

- Decrease contrast. This operation should do the opposite of the previous operation: Replace the red component r by the value

$$(255.0)\left[\frac{1}{\pi}\left(\sin^{-1}\left(\frac{r - 127.5}{127.5}\right)\right) + .5\right]$$

  and then round it to the nearest integer. This will move lower values for r farther from 0, and higher ones farther from 255. Do the same for the green and blue components.

The user should input one of 'E' (for enhance contrast) or 'D', (for decrease contrast) and then the program should do the related operation, and then output the new color's coordinates. If the user inputs a character other than those two, however, the program should print "Invalid option" and terminate. Here are a few examples of input and output. As usual, user input is underlined.

Your program's input and output should follow the format shown in the examples below, including the exact input and output prompts, use of whitespace, and comma separated output.

Example 1:

```
Input r, g, b:  20 200 255
Options:
Enhance contrast (E)
Decrease contrast (D)
Enter E or D: E
(4, 227, 255)
```

Example 2:

```
Input r, g, b:  4 227 255
Options:
Enhance contrast (E)
Decrease contrast (D)
Enter E or D: D
(20, 200, 255)
```

Example 3:

```
Input r, g, b:  0 0 0
Options:
Enhance contrast (E)
Decrease contrast (D)
Enter E or D: R
Invalid option
```

When you are done, name the source code file <username>_1A.cpp. Here you replace <username> with your U of M email address; for example, if your email address is smithx1234@umn.edu, your file should be named smithx1234_1A.cpp. Then submit your program using the HW 1 Problem A submission link on the class website.

Problem B: Race Pace (30 points)

Suppose you are running a 10K and want to run it in 50 minutes. How many minutes per mile is this on average? Or suppose you want to run a half marathon in 2 hours. How many minutes per mile is this on average? More generally, suppose you want to run a race that is a given distance (where the distance can either be in kilometers or miles) in a certain time. What is the average pace you'll need to have in minutes per mile?

Write a C++ program that first asks a user whether they want to input the race distance in kilometers ('k') or miles ('m'). After they input this, your program should have them input the distance in that unit. It should then ask them the hours, minutes, and seconds for the target time. Once they input these, it should calculate and output the pace per mile in minutes and seconds, with seconds reported with two digits to the right of the decimal point.

Your program might need to convert kilometers to miles. Use one kilometer equals 0.62137119 miles.

Input Validation: For this program you will need to verify that some, but not all, of the user input is reasonable.

- When input the distance options (kilometers ('k') or miles ('m')) your program should check that the user input one of the valid options. If the user inputs an invalid distance option, the program should print "Invalid distance option" and stop running (this can be accomplished with the return statement).

- Distance - you are not required to verify the distance is reasonable. (e.g., you do not have to check that a positive distance was entered)
- Time - you are not required to verify that the time units entered by the user are reasonable.

Output Formatting: When outputting the pace, the number of seconds should be greater than or equal to 0 but strictly less than 60. Additionally, the seconds should be reported with exactly two digits to the right of the decimal point.

Here is some example output, with the user input underlined. As usual, follow this input/output format carefully.

Example:

```
Is distance in kilometers (k) or miles (m)?  k
Input distance:  10.0
Input target time in hours, minutes, and seconds.
Hours:  0
Minutes:  50
Seconds:  0
The average time per mile is 8 minutes 2.80 seconds
```

When you are done, name the source code file <username>_1B.cpp. Here you replace <username> with your U of M email address; for example, if your email address is smithx1234@umn.edu your file should be named smithx1234_1B.cpp. Remember to follow this naming convention diligently. Then submit your file using the Homework 1 Problem B link on the class website.