CSCI 1312 (Introduction to Programming for Engineering), Fall 2016

Homework 4

Credit: 35 points.

1 Reading

Be sure you have read (or at least skimmed) the assigned readings from chapter 4, and section 9 (about recursion) from chapter 6.

2 Honor Code Statement

Please include with each part of the assignment the Honor Code pledge or just the word "pledged", plus one or more of the following about collaboration and help (as many as apply).[1] Text *in italics* is explanatory or something for you to fill in. For written assignments, it should go right after your name and the assignment number; for programming assignments, it should go in comments at the start of your program.

- This assignment is entirely my own work.
- This assignment is entirely my own work, except for portions I got from the assignment itself *(some programming assignments include "starter code")* or sample programs for the course *(from which you can borrow freely — that's what they're for).*
- I worked with *names of other students* on this assignment.
- I got help with this assignment from *source of help — ACM tutoring, another student in the course, the instructor, etc.*
- I got significant help from *outside source — a book other than the textbook (give title and author), a Web site (give its URL), etc.. ("Significant" here means more than just a little assistance with tools — you don't need to tell me that you looked up an error message on the Web, but if you found an algorithm or a code sketch, tell me about that.)*
- I provided significant help to *names of students* on this assignment. *("Significant" here means more than just a little assistance with tools — you don't need to tell me about helping other students decipher compiler error messages, but beyond that, do tell me.)*

3 Programming Problems

Do the following programming problems. You will end up with at least one code file per problem. Submit your program source (and any other needed files) by sending mail to `bmassing@cs.trinity.edu` with each file as an attachment. Please use a subject line that mentions the course and the assignment (e.g., "csci 1312 hw 4" or "CS1 hw 4"). You can develop your programs on

[1] Credit where credit is due: I based the wording of this list on a posting to a SIGCSE mailing list. SIGCSE is the ACM's Special Interest Group on CS Education.

any system that provides the needed functionality, but I will test them on one of the department's Linux machines, so you should probably make sure they work in that environment before turning them in.

1. (15 points) Write a C program that defines and tests a function for converting Fahrenheit temperatures to Celsius. (You should be able to reuse your code from Homework 2, perhaps with some slight modifications.) Your program should include three functions:

   - A conversion function to perform the actual conversion, declared as follows:

     ```
     double f_to_c(double f);
     ```

     where the single parameter `f` represents a Fahrenheit temperature and the return value represents the equivalent Celsius temperature. (So, for example, `f_to_c(32.0)` should evaluate to 0.0.0

   - A convert-and-print function declared as follows:

     ```
     void convert_and_print(double f);
     ```

     that calls the conversion function and prints its input and output nicely. So for example `convert_and_print(32.0);` might print

     ```
     32 degrees Fahrenheit is 0 degrees Celsius
     ```

     (Don't worry too much about printing appropriate numbers of digits after the decimal point; you can do whatever is easy, or read the man page or other documentation for `printf` to find out how to get more control.)

   - A main function that calls `convert_and_print()` at least four times with different inputs that you think will illustrate that the conversion is working right. (So this program is self-contained and doesn't prompt the user for anything!)

     NOTE that the point of this problem is for you to practice defining and using functions, so you will not get full credit unless your program includes functions as described.

2. (20 points) Write a C program that asks the user for two non-negative integers, call them a and b, not both zero, and computes and prints $\gcd(a, b)$, the greatest common divisor of a and b, using a recursive version of Euclid's algorithm. Print an error message if what was entered is not two integers, or either input is negative, or both are zero.

   Euclid's algorithm can be described recursively thus: For non-negative integers a and b, not both zero, with $a \geq b$,

   $$\gcd(a, b) = \begin{cases} a & \text{if } b = 0 \\ \gcd(b, a \bmod b) & \text{otherwise} \end{cases}$$

   where $a \bmod b$ is the remainder when a is divided by b. (You don't actually have to understand this algorithm to turn it into code, but if you want to and don't, a Web search will likely turn up some good explanations of how/why it works.)

   NOTE that the point of this problem is for you to practice defining and using a recursive function, so you will not get full credit unless you do. I recommend putting all the error checking in the main program and having a recursive function declared as

   ```
   int gcd(int a, int b);
   ```