CSCI 1120 (Low-Level Computing), Fall 2018

Homework 3

Credit: 10 points.

1 Reading

Be sure you have read, or at least skimmed, the assigned readings for classes through 9/05.

2 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 1120 hw 3" or "LL hw 3"). You can develop your programs on 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.

Note about error checking: Starting with this assignment, I want you to do at least minimal checking that input from standard input is sensible. As mentioned in class, doing a really great job of parsing and validating input is not especially easy, but for our purposes I say it's enough to check that `scanf` was able to get value(s) of the requested type(s) and that the values meet whatever other criteria the problem states (e.g., not negative for the second problem).

1. (5 points) Write a C program that asks the user for three integers and prints them in order from smallest to largest, or an error message if what was entered is something other than three integers. (You may recognize this problem as a special case of one you probably spent some time on in CS1. Don't try to solve that problem in general for this assignment; just solve the "three numbers" problem *using what we've discussed in class so far* — in particular I'd rather you not use arrays.) Note that the numbers do not have to be distinct — for example, the user could enter three 0s.

2. (5 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. (*Note* that you will not get full credit unless you use recursion.)

   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.

3 Honor Code Statement

Include the Honor Code pledge or just the word "pledged", plus *at least one 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 programming assignments, this should go in the body of the e-mail or in a plain-text file `honor-code.txt` (no word-processor files please).

- This assignment is entirely my own work. *(Here, "entirely my own work" means that it's your own work except for anything you got from the assignment itself — some programming assignments include "starter code", for example — or from the course Web site. In particular, for programming assignments you can copy freely from anything on the "sample programs page".)*
- 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. (Here, "help" means significant help, beyond a little assistance with tools or compiler errors.)*
- I got help from *outside source — a book other than the textbook (give title and author), a Web site (give its URL), etc.. (Here too, you only need to mention significant help — 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 help to *names of students* on this assignment. *(And here too, you only need to tell me about significant help.)*

4 Essay

Include a brief essay (a sentence or two is fine, though you can write as much as you like) telling me what about the assignment you found interesting, difficult, or otherwise noteworthy. For programming assignments, it should go in the body of the e-mail or in a plain-text file `essay.txt` (no word-processor files please).

[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.