CSCI 1312 (Introduction to Programming for Engineering), Fall 2018

Homework 4

Credit: 50 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 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 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 must 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.)

   - 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. (15 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);
   ```

3. (20 points) Write a C program that uses a recursive function to "count" up or down, specifying a starting value `start`, an ending value `end`, and an increment `incr`. The program should prompt for the three values and (of course?) just print an error message if what is entered is anything but integers, or if `incr` is 0 (think about why). The function should then count starting at `start`, incrementing by `incr`, and stopping when the next value would go beyond `end`. Examples:

   - `count(1, 6, 1)` should print the values 1, 2, 3, 4, 5, 6.
   - `count(6, 1, -1)` should print the values 6, 5, 4, 3, 2, 1.
   - `count(1, 10, -1)` should print the value 1.
   - `count(4, 10, 2)` should print the values 4, 6, 8, 10.
   - `count(4, 10, 4)` should print the values 4, 8.

   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

   ```
   void count(int start, int end, int incr);
   ```

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).

[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.

- 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).