

CSCI 1120 (Low-Level Computing), Fall 2021

Homework 2

Credit: 5 points.

1 Reading

Be sure you have read, or at least skimmed, the assigned readings for classes through 09/01, including the video lectures.

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) one of two ways:

- using my `mail-files` script, linked from the course Web site under “Links”.
- by putting them in your course “TurnIn” folder on Google Drive. (Note that I want plain-text files, ideally with an extension of `.c`, but if Google Drive balks at that, rename to have an extension of `.txt`. I want something I can compile as is, except for possibly a change of filename. So no screenshots!)

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. (5 points) Write a C program to convert seconds into years, days, hours, minutes, and seconds. Your program should prompt the user for a number of seconds, get the number entered, and print the equivalent number of years, days, etc. (e.g., 100 seconds is 0 years, 0 days, 0 hours, 1 minute, and 40 seconds, while 100000000 seconds is 3 years, 62 days, 9 hours, 46 minutes, and 40 seconds). Assume 365 days in a year (not quite right but makes the calculations simpler). *For this assignment only*, you do not need to do any kind of checking that what the user enters is actually an integer and non-negative, since we haven’t yet talked about conditional execution. Just assume it is and do the required calculations.

Hints:

- Probably the best way to do the required calculations is with the integer-division (`/`) and remainder (`%`) operators.
- Be advised that a C-idiomatic way to define constants is with `#define`, e.g.,

```
#define SECONDS_PER_MINUTE 60
```

usually before the first function that uses them. This can help make code more human-readable. *Note* that the definition (60 in the above example) can be an expression, *but* because of the way this works you should enclose it in parentheses (more when we talk about the `#` directives). Example:

```
#define SECONDS_PER_HOUR (60*60)
```

3 Essay and pledge

Include with your assignment the following information.

For programming assignments, please put it a separate file. (I strongly prefer plain text, but if you insist you can put it in a PDF — just no word-processor documents or Google Drive links please.) For written assignments, please put it in your main document.

3.1 Pledge

This should 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). Text *in italics* is explanatory or something for you to fill in; you don’t need to repeat it!

- I did not get outside help *aside from course materials, including starter code, readings, sample programs, the instructor.*
- 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, 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.)*

3.2 Essay

This should be a brief essay (a sentence or two is fine, though you can write as much as you like) telling me what if anything you think you learned from the assignment, and what if anything you found interesting, difficult, or otherwise noteworthy.