CST8234 – C Programming F17 (Lab 2)

Programming Exercise

In this lab, we will gain experience with

- Making use of scanf/printf for getting input and formatting output
- Starting thinking about programming against a test suite
- Learning to adapt to an imposed programming style

Statement of the problem

North American telephone numbers always follow the following structure

> 3-digit area code + 3-digit central office code + 4-digit subscriber

For example, the Algonquin College main phone number is "(613) 727-4723" where the area code is "613", the central office (or exchange) is "727" and the subscriber is "4723". Neither the area code nor central office will ever start with the digit "0" or "1". See https://en.wikipedia.org/wiki/North_American_Numbering_Plan#Modern_plan.

However, despite a consistent structure, there's a huge variety of ways of presenting these numbers. I.e., it's not unfamiliar to see "(613) 727-4723" or "613-727-4273" or "613.727.4273", etc. But the presentation will always illustrate the structure of the number. I.e., you'd never see a North American phone number presented like "61-37-274-273"!

Historically, it has been common to ignore the area code when communicating phone numbers, provided both people had a common understanding of what the area code was (e.g., if you told people that the Algonquin College phone number was "727-4723", most people in Ottawa would figure out that there is an implicit "613" at the front). This is referred to 7-digit representation, as opposed to the more modern 10-digit representation that makes more sense now that many people (particularly students!) have mobile phones that are actually registered in a different area code (e.g., a student from Toronto will likely have a "416" number, despite the fact that they are living in Ottawa).

But because presentation is generally a user or application preference, and consistency is makes for easier reading, applications that record/show phone numbers *generally* store the phone numbers in a canonical representation with all formatting removed (e.g., "6137274723"), and then re-apply the formatting when the number has to be presented to a user.

In this lab we will simulate the process of parsing and formatting a phone number represented in canonical format.

Requirements

1. Create a folder called algonquinUserID_L2 (e.g., "mynam00123_L2"). Do all of your work in this folder, and when complete, submit the zipped folder as per the "Lab Instructions" posted on Blackboard.
2. Write a program that will continually loop, accepting a phone number entered on stdin in 7-digit canonical format, and then
    a. If the entered number is 0, your program will exit
    b. If the entered number is invalid, or isn't a legal 7-digit phone number, you will report the problem
    c. Otherwise, you will format the number in a hyphenated 7-digit representation. E.g., turn "7274723" into "727-4723"
3. Conform to the coding conventions used in K&R.

Example execution

```
# ./phone
Enter phone number: 1
Invalid number '1'
Enter phone number: 7274723
727-4723
Enter phone number: 0
#
```

Note: do NOT try to support area codes. You'd then have to get into "long" vs "int" variables, and it would likely make the whole program a little more complicated to write and debug. I.e., assume we are all working with in a single area code, so that the entered numbers can be expected to be 7-digit canonical numbers, and need to be formatted in 7-digit.

You will compile with gcc's "`-pedantic -ansi -Wall -W`" options.

Hint: you can spend ages making sure that you format the phone number correctly, or you can read up on the very flexible formatting support of printf.

Hint: you can test your program over and over by hand, or you can make use of linux/cygwin's ability to redirect input (remember your Linux fundamentals course?). So you may want to create a file with a test suite of numbers. If you do, include it in the submission.

A comment on coding conventions: Yes, there are quite a few different coding conventions in the world, and some are slightly better than what is shown in K&R, but when graduate and find employment, you will be required to adopt the convention that is used by the other members of your team. So yes, you may *prefer* a different style over K&R, but part of this exercise is getting you to be able to adapt to a pre-established convention.

Marking

This assignment is out of 40

- 10 for coding correctness (i.e., correct results)
- 10 for appropriate and efficient logic (i.e., no spaghetti code!)
- 10 for adhering to K&R conventions
- 10 for test suite coverage (i.e., are you catching all possible errors?)

You can also lose marks for incorrect submission (e.g., including unnecessary files in the zipped folder), compiler warnings, typographic errors (including in comments).

Submission

When you are done, submit your program to Blackboard. Make sure that you have the appropriate header in your source file(s), and have zipped up the appropriately name directory. Only include the source code (.c, .h) and a test file if you used one.