CST8234 – C Programming F17 (Lab 4)

Programming Exercise

In this lab, we will gain experience with

- Using function pointers
- Using map/lookup tables
- Formatting floating point numbers

Statement of the problem

We'll be building a very simple calculator. Your program will ask the user for two numbers, and the operation that is to be performed, based on a list of hotkeys you'll present to the user. Study the examples below:

```
$ ./lab4.exe
Enter first operand: 3
Enter second operand: 5
Hotkeys:
0 - Add
1 - Subtract
2 - Multiply
3 - Divide
Enter hotkey: 3
3.000 / 5.000 = 0.600

$ ./lab4.exe
Enter first operand: 3
Enter second operand: 5
Hotkeys:
0 - Divide
1 - Multiply
2 - Add
3 - Subtract
Enter hotkey: 3
3.000 - 5.000 = -2.000
```

Note that each time you execute the program, the hotkeys must be randomized. Luckily, you all know how to shuffle a sequence of numbers now!

There are four operations (as listed) and once you have acquired and validate the two operands and the hotkey, you must print the final line with a single line of code. I.e., the goal here is to get all the desired information from a look, so no if-then-else's are permitted once you have all the information you need.

Data Structures

At the heart of this assignment is creating a "HotKey" data structure that has an operation name, and operation symbol, and a function pointer that is initialized to a simple function that accomplishes the

desired calculation with the provided operands. You'll need an array of four of these structures. When the user selects their operands and hotkey, you'll be able dereference the appropriate hotkey and execute the function pointer to get the arithmetic result.

Randomizing

You'll need to either randomize your hotkeys, in place, or use a randomized lookup table that will shuffle the order in which the operations are presented to the user. Regardless of the methods, remember that on Friday we discussed two super-easy (and fast) ways to shuffle a list using swaps. I'd recommend using one of those.

Requirements

1. Create a folder called algonquinUserID_L4 (e.g., "mynam00123_L4"). 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
    a. Randomizes a "deck" of integers that will map a number from 0 – 3 to possibly different number from 0 – 3.
    b. Asks the use for the two floating point operands, and an integer hotkey (from 0 – 3), and check the validity of each input, and report any problems and repeat until valid
    c. When acceptable inputs have been acquired, perform the requested arithmetic operation, and print a pleasing description of the calculation (see examples above) using three decimal places.
3. This is such a short lab, that you can implement in in one function if you want, however you still must provide a makefile with the standard compiler flags (`-g -Wall -w -pendantic -ansi`)

Marking

This assignment is out of 30

- 10 for coding correctness (i.e., correct results)
- 10 for appropriate and efficient logic (i.e., no spaghetti code!)
- 10 for clear comments and coding convention (not necessarily K&R, but it should be clean and consistent)

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 your makefile (mandatory!).

You ***must*** include a makefile, as part of your submission! When grading your reports I will unpack your zip file and type '`make`'… and if I don't end up with a 'lab4.exe' to run, ***I will not grade your assignment***.