CST8234 – C Programming F17 (Lab 3)

Programming Exercise

In this lab, we will gain experience with

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

This assignment is to be done in pairs.

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 or +: Add
1 or -: Subtract
2 or *: Multiply
3 or /: Divide
Enter hotkey: 3
3.000 / 5.000 = 0.600

$ ./lab4.exe
Enter first operand: 3
Enter second operand: 5
Hotkeys:
0 or /: Divide
1 or *: Multiply
2 or +: Add
3 or -: Subtract
Enter hotkey: -
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 lookup table and invoke a function pointer, so no if-then-else's or switches are permitted once you have all the information you need. If you use any if-then-else or switch to do the requested calculation once you have the user's chosen operation, you will get 0 for the implementation!

Data Structures

At the heart of this assignment is creating a "HotKey" data structure that has

- an operation name (e.g., "Divide"),
- an operation symbol (e.g., '/'), 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 to access the appropriate hotkey structure from the array, and execute (i.e., dereference) the function pointer to get the arithmetic result in a single line of code.

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. Remember that in class we discussed a super-easy (and fast) way to shuffle a list using swaps.

Selecting an Operation

The user can enter either an integer (e.g., '0', '3') or symbol (e.g., '*', '+') to select an operation. You'll have to figure out which they entered, and then look up the desired hotkey structure by indexing into the array using the entered integer, or by iterating over the array looking for the entered symbol.

Requirements

1. Create a folder based on both of your Algonquin User ID's (e.g., "smit9112_sing0003"). Do all of your work in this folder, and when complete, submit the zipped folder as per the "Lab Instructions" posted on Brightspace.
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 report any problems and repeat until valid.
    c. Print out a list of shuffled operations (using the operation name for each element in the array of data structures), and asks for an indicator of the desired operation. This indicator can be either an integer OR a symbol. If the user enters something other than a valid symbol or integer, you will report the problem and repeat the prompt.
    d. When acceptable inputs have been acquired, get the appropriate element from the hotkey structure lookup table (using either an integer or symbol).
    e. Use the hotkey structure to perform the requested arithmetic operation, and print a pleasing description of the calculation (see examples above) using three decimal places for both the inputs and the result.
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)
- A -3 penalty applies for work submitted individually

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 Brightspace. 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 'lab3.exe' to run, ***I will not grade your assignment***.