CS 135 -- Lab 4

The purpose of this lab is to create a beginning version of our **Calculator** project (Project 1) using the concept of functions. We will use last week's programs as building blocks.

Last week we created two stand-alone programs: **Add_2_Numbers** & **Get_Choice**
to illustrate if-then-else logic and switches for making choices.
The final results of these were posted on Canvas in `Assignments > Lab 3`
as: ***Add_2_Numbers.gif*** & ***Get_Choice_v6.gif***

1) Create a new folder: `Desktop > code_2020-09-14`

2) Create a new folder named **Calculator** within `code_2020-09-14`.
Start the EditPlus editor: `File > Open`
Browse to: `Desktop > code_2020-09-14 > Calculator`
Create a new source code file named *Calculator_v1.cpp*.

3) Build the program according to the general outline below.
Use what you have learned in Chapters 2-4, and what we discussed in class to fill in the details.

General Outline

File Header

```
#include's

// prototypes go here
```

Function Header

```
int main()                    //   new "driver" function
...
char Get_Choice()             //   previous Get_Choice_v6.cpp
...
void Add_2_Numbers()          //   previous Add_2_Numbers.cpp
...
```

4) More details on the `main()` driver function: Below is a sketchy, bare bones outline.

```
int main()
{
    while(true)               //   while loop to keep the program open
    {
        choice = Get_Choice();
        switch (choice)
        {
            case 'a':         //   goes to Add_2_Numbers();

            case 'q':         //   goes to quit. You need some "exit" strategy here.
        }
    }                         //   close while loop
                              //   say "Goodbye"
}
```

5) Now you can just copy and paste the code created for:
Add_2_Numbers.cpp & *Get_Choice_v6.cpp* into the appropriate place under their function names.
You should be able to get the code to compile with a couple of minor adjustments to the functions.

`char Get_Choice()`	needs to return `choice` and
`void Add_2_Numbers()`	does not return anything.

And don't forget to put prototypes where they belong.

6) Once the code is compiled and running, add a few things to make it behave nicer.
- Add `'q'` to the menu code with a "quitting" message.
- Comment out clearing the screen so you can see the result of the calculations.
- Come up with a "clean" exit from the `while` loop. Don't use the `exit()` function.
  That is equivalent to pushing the ejector seat button. (It is always better to land the plane.)

For Day 2: We started work on *Calculator_v2.cpp*.

Remember, last week we learned how to create a second version of source code within the current Project.

7) Add the following features to version 2.
   • A *bool* flag in the `main()` function to exit the `while` loop.
   • Cut the code for inputting *x* and *y* out of `Add_2_Numbers()` and put it in a new function:
   `void Get_x_and_y()` to create a stub of "reusable code", to be used by the future math functions.

   This raises the dilemma of "scope", and how to get the *x* and *y* values between functions.
   An allowable solution to this is on p. 342 of the textbook.

8) Save all the code in `code_2020-09-14` folder as a zip file.

Submission Instructions

Make sure the code has the appropriate file comment headers at the beginning, and function headers before each function. (samples on Canvas).
Compile and test it in the MinGW environment we use at TMCC. That is how it will be graded.

The zip file should contain the `code_2020-09-14` folder containing the **Calculator** project folder, containing the source code and executables for:

- ***Calculator_v1.cpp***
- ***Calculator_v1.exe***
- ***Calculator_v2.cpp***
- ***Calculator_v2.exe***

Both versions should be compiled and running, and will be tested for grading.

Submit the zip file to the Drop box.

Future Work

You are now in a position to add functions and flesh out your **Calculator** for Project 1.
So, go for it!

9) Unresolved issues that you might want to think about solving in future versions.
   • Giving incorrect input can cause an infinite loop with the menu.
   • Project 1 requirements say you must optionally be able to handle integer and floating point arithmetic.