Lab 4

Due Feb 13 by 11:59pm **Points** 25 **Submitting** a file upload **File Types** cpp and h

Structures, Functions, and Pointers

Chapters 4, 5, and 6

The General Programming Lab Instructions (%24CANVAS_OBJECT_REFERENCE%24/assignments/g802605efa941ab100e32a8a582cfaea3) apply to this assignment.

Please note that the extended time between the due dates for this and the previous lab is because this lab utilizes concepts for three chapters: 4, 5, and 6. However, you may begin this lab after finishing chapters 4 and 5, and then complete it after finishing chapter 6 (especially argument passing).

For many years the British Empire used a three-unit monetary system based on pounds, shillings, and pence. 12 pence = 1 shilling and 20 shillings = 1 pound. Amounts in this system are printed out in a two-decimal point notation: `£6.03.05` (meaning 6 pounds, 3 shillings, and 5 pence). This is just enough detail (data and functions) to make British currency an interesting demonstration of structures and functions.

Examples

Two versions of the the Time program serve as examples for this lab. Each version illustrates a different aspect of the assignment. Please study both versions.

- **American** (http://icarus.cs.weber.edu/~dab/cs1410/textbook/5.Structures/progexample/american.html) Chapter 5. The video is short and should be helpful. The problem with the Time example is that the conversion factors are both 60 (60 seconds in a minute and 60 minutes in an hour). The American example has different conversion factors that will help you with the unit conversions (pence to shillings and shillings to pounds and back again)

- **Time (http://icarus.cs.weber.edu/~dab/cs1410/textbook/6.Functions/progexample/time.html)** Chapter 6 (Functions), demonstrates pass-by-pointer and pass-by-reference. Picks up where the chapter 5 video ends.
- Review how the compiler builds multi-file programs (**Figure 4 (http://icarus.cs.weber.edu/~dab/cs1410/textbook/1.Basics/compiler_op.html#compillus)**).
  - Do not #include "Time.**cpp**" in driver.cpp

Hints

Compare the the chapter 5 American example with the Lab 5 sterling problem:

	American	strerling
smallest unit	3 (teaspoon)	12 (pence)
medium unit	2 (table spoon)	20 (shillings)
largest unit	6 (2 * 3, fl oz)	240 (12 * 20, pounds)

Compare the numbers with where they appear in the following code fragments from the American example:

from make_American(int)

```
temp.ounce = tsp / 6;
tsp %= 6;
temp.tablespoon = tsp / 3;
temp.teaspoon = tsp % 3;
```

from add

```
int i1 = am1.ounce * 6 + am1.tablespoon * 3 + am1.teaspoon;
```

You can follow the same *pattern* to create the corresponding functions needed in lab 5.

Program Requirements

The program will consist of three files as follows:

1. In a file named `sterling.h`
   a. Create a structure named `sterling` (begins with a lower-case "s") with three fields to represent pounds, shillings, and pence.
   b. Add function prototypes for the functions defined in step 2 below
2. Define the functions that work with sterling structures in a file named `sterling.cpp`
   a. Define a `make_sterling` function that takes thee integers representing pounds, shillings, and pence (in that order) and returns a sterling structure
   b. Define a `make_sterling` function that takes a single integer for pence and converts it into an equivalent value represented by pounds, shillings, and pence, which it returns as a sterling structure.
   c. Define a function named `add`
      i. that accepts two sterling structure arguments **passed by value (http://icarus.cs.weber.edu/~dab/cs1410/textbook/6.Functions/value.html)** (chap 6)
      ii. adds them together
      iii. returns the result as a new sterling structure (the function must not change either argument). The result should have an appropriate value for pounds; the value of pence must in the range 0-11, and the value of shillings must in the range 0-19. That is, represent the sum with the least possible amount of pence and shillings.
   d. Define a function named `print`
      i. accepts one sterling structure **passed as a reference (http://icarus.cs.weber.edu/~dab/cs1410/textbook/6.Functions/reference.html)** (chap 6)
      ii. has a return type of void
      iii. prints the argument in the 3-decimal point notation with a leading pound currency sign: £6.03.05 (pounds, shillings, and pence). Print the pound sign with this code: `cout << (char)156;` (works on Windows but may not work on other platforms - **but the lab will be scored on Windows**)
      iv. The second and third values, the shilling and the pence, always displays as a 2-digit number. If the value is less than 10, then print the value with a leading 0: `£6.03.05` (use the **cout fill function (http://icarus.cs.weber.edu/~dab/cs1410/textbook/3.Control/more_io.html#io_member)**).
   e. Define a function named `read` that
      i. has a single sterling structure **passed by pointer (http://icarus.cs.weber.edu/~dab/cs1410/textbook/6.Functions/pointer.html)** (chap 6)
      ii. has a return type of void

   iii. reads a sterling structure from the keyboard (3 separate prompts and 3 separate reads - in the order of pounds, shillings, and pence)
   iv. returns the sterling structure through the argument (chap 6)
3. Create a driver program in a file named `driver.cpp` to test your functions. How you test the functions (i.e., what you do in the driver) is largely up to you. A good approach is follow the Chapter 6 Time example

Grading

Upload three files (`sterling.h`, `sterling.cpp`, and `driver.cpp`) to Canvas for grading. Please make sure that the files are named correctly. In addition to your driver program, I will link my own code with your sterling code, so it is important the structure name and function signatures are as described above. (Remember that C++ is a case sensitive language.)

Test Cases

The test cases are formatted as they are to help you see the carry operations. You *may* format your output as illustrated below but it is *not* required.

```
  £1.05.05
+ £1.08.06
---------
  £2.13.11


  £1.10.06
+ £1.10.06
---------
  £3.01.00


  £5.15.09
+ £6.14.08
---------
 £12.10.05
```

Lab 4 Scoring

Criteria	Ratings	Pts
File Names Files are named correctly: sterling.h, sterling.cpp, and driver.cpp		3.0 pts
Structure Name The structure is named correctly (begins with a lower case "s").		1.0 pts
Structure Content Fields and function prototypes are correct, including names.		4.0 pts
sterling.cpp make_sterling Function name and all operations are correct.		3.0 pts
sterling.cpp add Function name, arguments, and all operations are correct.		3.0 pts
sterling.cpp print Function name, arguments, and all operations are correct.		3.0 pts
sterling.cpp read Function name, arguments, and all operations are correct.		3.0 pts
driver.cpp Operations Structure is correct and tests all sterling functions.		3.0 pts
Miscellaneous Each file is structure correctly as demonstrated in the examples. The files are clean and easy to read: indentation is consistent, blank lines are used appropriately; the program is created "Empty," and all "pause" statements and/or dummy reads are commented out or removed, etc.		2.0 pts
		Total Points: 25.0