Project #10 (50 points, 10% bonus for exact match): Classes

Submit Your Solution Using Canvas by 10 PM on Friday, 4/21/23

(late submissions will be accepted on 4/21/23 from 10PM to Midnight)

Obtaining Project 10 Input Order and Output:

Your program's output should look like that produced by the provided sample solution. You can run the sample solution **Project_10_solution** by typing the following at a command prompt in a terminal window**:**

```
/home/work/cpe211/Executables/Project_10/Project_10_solution
```

The comparison script can be run by using the following command

```
/home/work/cpe211data/Project_10/CompareSolution.bash   Project_10.cpp
```

Project 10 Restrictions

Any C++ technique covered in Chapters 1 through 12 is allowed except for global variables ***You are not allowed to use any global variables.*** If necessary, global constants may be used.

Project 10 Description

For this project a Money class (specified in the file **Project_10.h**) will be constructed and all function definitions will be written in the file Project_10.cpp. A **makefile** is provided to compile this project. The file **Project_10_main.cpp** contains the main function that is used to run the program.

From Canvas, download the files Project_10.h, Project_10_main.cpp and Makefile. Save these files in your Project 10 directory. **Do not modify these files.**

Another file is provided as well. It **is Project_10.cpp. Download this and all the code you write will be in this file.** This file contains all the function definitions for the methods for the class specification provided in the header file Project_10.h

This project is similar to the Time class described in the classes power point slides and gone over in class

Note: do not use any variable/parameter in your functions or program that are named pennies, nickels, dimes or quarters – these are used as private members of the class and you want to make sure that you know which variable or parameter you are talking about.

For this project the following functions are written:

1) For class constructors:
    a. A default constructor called Money() – default amount is 0 for each attribute

    b. A 4 parameter parameterized constructor Money(int, int, int, int) – initialize the attributes pennies, nickels, dimes and quarters of a Money object to the values provided when it is declared(i.e. Money myMoney(1,2,3,4))

c. A 3 parameter parameterized constructor Money(int, int, int) – initialize the attributes pennies, nickels and dimes of a Money object with the values provided when it is declared (i.e. Money myMoney(1,2,3)).
The quarters attribute is set to 0

d. A 2 parameter parameterized constructor Money(int, int) – initialize the attributes pennies and nickels of a Money object with the values provided when it is declared (i.e. Money myMoney(1,2)). The quarters and dimes attributes are set to 0

e. A 1 parameter parameterized constructor Money(int) – initialize the attribute pennies of a Money object with the value provided when it is declared (i.e. Money myMoney(1)). The quarters, dimes and nickels attributes are set to 0

2) For the class Transformers

a. A parameterized function SetMoney(int,int,int,int) – this sets the attributes in a Money object to the values specified. The order is pennies, nickels, dimes and quarters

b. A 4 parameter function AddMoney(int,int,int,int) that adds the number of pennies, nickels, dimes and quarters specified to the Money object attributes for pennies, nickels, dimes and quarters.

c. A 3 parameter function AddMoney(int,int,int) that adds the number of pennies, nickels and dimes specified to the Money object attributes for pennies, nickels and dimes.

d. A 2 parameter function AddMoney(int,int) that adds the number of pennies and nickels specified to the Money object attributes for pennies and nickels.

e. A 1 parameter function AddMoney(int) that adds the number of pennies specified to the Money object attribute for pennies.

3) For the class observers –two functions for writing out the information of a Money object. One function will output the total number of each coin type – one value per line as shown by the sample solution. The other function will output the total dollar amount represented by the coins in a Money object.

a. WriteCoinTotals() – writes out the coin totals for each attribute one value per line with an identifying phrase as shown by the sample solution (i.e Number of Quarters: 23)

b. WriteDollarTotal() – writes out the dollar and cents amount contained by the attributes. For example if it is 2 pennies, 5 nickels, 10 dimes and 5 quarters, then the output is: The dollar amount of the coins is: $2.52.
Note on this function if the cents are less than 10, then the cents part needs a leading 0. For example the output will look like $##.0# - where # just indicates there is a numeric value not being shown.

After downloading all 4 files (Project_10.h, Makefile, Project_10_main.cpp and Project_10.cpp), start your work by making empty function stubs in Project_10.cpp. The function headings are determined from the public functions listed in Project_10.h. Once you have all the function stubs written, you should be able to run the Makefile (type make at the command prompt) to generate the program executable Project_10. You can run the program, but not much happens.

Now add code to each of the functions described above and after completing each function, run the makefile to make sure the program still compiles. If it does not compile fix the errors until it does compile. I recommend writing the default constructor, the 4 parameter constructor, WriteCoinTotals and WriteDollarTotal functions first.
Then the remaining functions can be written in any order.

Project 10 Function Contents

1) The **constructor functions** set the private members pennies, nickels, dimes and quarters to the default values(all 0) or the values specified.
   a. The parameter order for the 4 parameter parameterized constructor is pennies, nickels, dimes and quarters .
   b. The parameter order for the 3 parameter parameterized constructor is pennies, nickels and dimes. The quarters parameter is set to 0.
   c. The parameter order for the 2 parameter parameterized constructor is pennies and nickels. The quarters and dimes parameters are set to 0.
   d. The parameter order for the 1 parameter parameterized constructor is pennies. The quarters, dimes and nickels parameters are set to 0.

2) The **SetMoney** function takes in the number of pennies, nickels, dimes and quarters (in that order) and stores them in the appropriate private members.

3) The **AddMoney** function adds the values provided to the Object attributes. There are 4 versions of this function with 4,3,2 and 1 parameters. The order of the parameters is pennies, nickels, dimes and quarters. In the case of 3 parameters, there are no quarters to add to the quarters attribute. For the 2 parameter version, there are not quarters or dimes to add to the attributes. For the 1 parameter version there are no quarters, dimes or nickels to add to the attributes.

4) The **WriteCoinTotals** function takes the values of the private members pennies, nickels, dimes and quarters and outputs them one per line with a leading phrase before writing out the quantity of that particular coin type.

5) The **WriteDollarTotal** function uses the values of the private members pennies, nickels, dimes and quarters to determine a total dollar amount. This dollar amount is then output with a leading phrase and then the dollar amount in the form of $##.## where the # indicates a digit. This amount will be in dollars and cents with the cents between 0 and 99. For cent values less than 10, a leading 0 is required – i.e $23.05

When completed, the program output shall match that of the sample solution. There are no input files for this program. All operations controlling the output are run from Project_10_main.cpp.

Output that does not match that of the sample solution will lose points.

All of these functions are relatively short (less than 10-15 lines)

Submit your Project_10.cpp file only on Canvas.

Project 10 Helpful Information

- After all stubs have been created and the Makefile compiles the program without errors, write one function and then run the Makefile by typing make on the command line. Fix all errors in the function before moving on to the next function.

- **Do not modify the Makefile, Project_10.h or the Project_10_main.cpp.**
- **The only file that you should be changing is your Project_10.cpp file**