HW2: Classes and Operator Overloading

New Attempt

Due Apr 14 by 4:45pm **Points** 25 **Submitting** a file upload

For problems 1b and 2c, before writing code, please first write out your idea for your program on paper. This could be pseudocode, or a couple of sentences describing your approach. Please name the file(s) you upload for your explanation(s): Problem1bidea.(pdf, jpg, etc), Problem2cidea.(pdf, jpg, etc); or Problem1b2cidea.(pdf, jpg, etc) if you include your ideas for multiple problems in the same file.

Note: For this week, you may have your code for each problem all in one file. If you do, please name your files HW1P1Lastname.cpp and HW1P1Lastname.cpp (where Lastname is your last name)

If you choose to separate your code into multiple files, please name them Rational.h, Rational.cpp, main1.cpp for problem1, and ATM.h, ATM.cpp, main2.cpp for problem 2.

Warm-up:

Problem 1 (10 points). This problem will build on the Rational class (Download the code from Monday 4/5). Implement the following functions, including pre and post comments for each.

a. Non-member: bool operator ==(Rational lhs, Rational rhs); Returns true if a==b and false if not. Note; if a is 12/36 and b is 1/3, it should return true. You should NOT divide the numbers and compare the decimal results – this process is prone to errors because of rounding errors.

b. Add another member function to your class: void reduce() - A member function that puts the rational into its most reduced form. For instance, it would change 12/36 to 1/3. Note: you will have to find the greatest common divisor to do this.

Modify your +=, *=, + and * functions so that they return the reduced version of the result.

Main Event:

Problem 2. (15 points) Create a class Atm that represents an ATM machine. This class will have private member variables twenties_, representing the number of twenty-dollar bills the machine has available,

and tens_, representing the number of ten-dollar bills the machine has available. You will implement the following member functions:

- a. One constructor that takes no arguments, and one that takes two arguments
- b. Getters for the private variables (no setters)
- c. BankAccount get_cash(int amount, BankAccount b) that takes in the amount of money desired and the BankAccount the money should be withdrawn from, and prints out a message telling the user how many of each kind of bill they will receive. This function should give as few bills as possible. You should update member variables as appropriate. Your function should return a modified version of b. Use the BankAccount class at this URL: **http://math.scu.edu/~linnell/cs60resources/HW-BankAccount-post.zip** **(http://math.scu.edu/~linnell/cs60resources/HW-BankAccount-post.zip)** (you will need to copy-paste the URL into your browser because Camino won't open a zip file). Be sure to handle the cases where the ATM doesn't have enough bills or the user requested an invalid amount.
- d. void operator +=(Atm& rhs) that transfers all bills from rhs to the ATM the function was called on
- e. void restock(int new20s, int new10s) that adds new20s twenties and new10s tens to the ATM.

And non-member functions:

- f. bool operator ==(Atm lhs, Atm rhs) that returns true if the two ATMs contain the same number of 10s and 20s, and false otherwise.
- g. ostream& operator <<(ostream& out, Atm a) that prints out information about a in a sensible format.

Finally, be sure to write a main function that sensibly tests all your functions.