Part C - Encapsulation   
  
**Member Operators / Helper Functions**  
  
Workshop 6 V1.1

In this workshop, you are to overload operators to work with a class called Account.

**Learning Outcomes**

Upon successful completion of this workshop, you will have demonstrated the abilities to

* overload an operator as a member function
* overload an operator as a helper function
* overload an operator as a friend function
* reflect on what you have learned in this workshop

**SUBMISSION POLICY**

The “in-lab” section is to be completed **during your assigned lab section**. It is to be completed and submitted by the end of the workshop. If you do not attend the workshop, you can submit the “in-lab” section along with your “at-home” section (a 30% late deduction will be assessed). The “at-home” portion of the lab is **due the day before you next scheduled workshop**

**Account Class**

Download or clone workshop 5 from <https://github.com/Seneca-OOP244/Workshop6.git>

Open Workshop6/in\_lab directory and view the code in **Account.h** and **Account.cpp.**

**Account** is designed and coded to hold and display information about an Account in a bank. These information are: name and balance.  
**Account** has three constructors and a display function.

Your task is to complete the code of the Account class or add helper functions to be able to work with following operators;  
 **“+”, “+=”, “=” and “<<”**

The overload of the above operators should make the following code possible:

If A and B and C are accounts and “value” is a double variable:

B + C: this operator returns an account with an empty name and a balance of the sum of two balances of B and C. This operator could be non-member and friend.  
  
B += C: adds the balance of C to B and returns the reference of B. This operator is a member operator.

A = "new name": Sets the name of A to “new name” and returns the reference of A.   
This operator is a member operator.

value += A; adds the balance of A to the “value” variable and returns the value.  
This operator is non-member and could be a friend.

cout << A; calls the display member function of A and returns the reference of cout.   
This operator is a non-member operator.

// OOP244 Workshop 6: operators

// File: w6\_in\_lab.cpp

// Version: 1.0

// Date: 2015/11/03

// Author: Fardad Soleimanloo

// Description:

// This file tests in-lab section of your workshop

/////////////////////////////////////////////

#include <iostream>

#include "Account.h"

using namespace sict;

using namespace std;

int main(){

Account A;

Account B("Saving", 10000.99);

Account C("Checking", 100.99);

double value = 0;

cout << A << endl << B << endl << C << endl << "--------" << endl;

A = B + C;

A = "Joint";

cout << A << endl << B << endl << C << endl << "--------" << endl;

A = B += C;

cout << A << endl << B << endl << C << endl << "--------" << endl;

value += A;

value += B;

value += C;

cout << "Total balance: " << value << endl;

return 0;

}

Output Example:  
(Your output should exactly match the following)

: $0.00  
Saving: $10000.99  
Checking: $100.99  
--------  
Joint: $10101.98  
Saving: $10000.99  
Checking: $100.99  
--------  
Saving: $10101.98  
Saving: $10101.98  
Checking: $100.99  
--------  
Total balance: 20304.95  
  
**in-lab SUBMISSION (80%)**

If not on matrix already, upload **Contact.h**, **Contact.cpp**, **PhoneNumber.h**, **PhoneNumber.cpp** and **w5\_in\_lab.cpp** to your matrix account. Compile and run your code and make sure everything works properly.

Then run the following script from your account:

**Sections SAA and SBB:  
~fardad.soleimanloo/submit w6\_in\_lab <ENTER>   
Section SCC and SDD:  
~ronald.burton/submit w6\_in\_lab <ENTER>**

and follow the instructions.

**AT-HOME Section: under construction.**