```cpp
//
//  main.cpp
//  AbsoluteCpp_ch8_5
//

#include <iostream>
#include <cstdlib>
#include <cmath>
using namespace std;

//Class for amounts of money in U.S. currency.
class Money
{
public:
    Money( );
    Money(double amount);
    Money(int theDollars, int theCents);
    Money(int theDollars);
    double getAmount( ) const;
    int getDollars( ) const;
    int getCents( ) const;
    friend const Money operator +(const Money& amount1, const Money&
     amount2);
    friend const Money operator -(const Money& amount1, const Money&
     amount2);
    friend bool operator ==(const Money& amount1, const Money& amount2);
    friend const Money operator -(const Money& amount);
    friend ostream& operator <<(ostream& outputStream, const Money& amount);
    friend istream& operator >>(istream& inputStream, Money& amount);
private:
    int dollars; //A negative amount is represented as negative dollars and
    int cents; //negative cents. Negative $4.50 is represented as -4 and -50

    int dollarsPart(double amount) const;
    int centsPart(double amount) const;
    int round(double number) const;
};

int main( )
{
    Money yourAmount, myAmount(10, 9);
    cout << "Enter an amount of money: ";
    cin >> yourAmount;
    cout << "Your amount is " << yourAmount << endl;
    cout << "My amount is " << myAmount << endl;

    if (yourAmount == myAmount)
        cout << "We have the same amounts.\n";
    else
        cout << "One of us is richer.\n";

    Money ourAmount = yourAmount + myAmount;
    cout << yourAmount << " + " << myAmount
         << " equals " << ourAmount << endl;
    Money diffAmount = yourAmount - myAmount;
    cout << yourAmount << " - " << myAmount
         << " equals " << diffAmount << endl;

    return 0;
}

ostream& operator <<(ostream& outputStream, const Money& amount)
{
    int absDollars = abs(amount.dollars);
    int absCents = abs(amount.cents);
    if (amount.dollars < 0 || amount.cents < 0)
        //accounts for dollars == 0 or cents == 0
        outputStream << "$-";
    else
        outputStream << '$';
    outputStream << absDollars;

    if (absCents >= 10)
        outputStream << '.' << absCents;
    else
        outputStream << '.' << '0' << absCents;

    return outputStream;
}

//Uses iostream and cstdlib:
istream& operator >>(istream& inputStream, Money& amount)
{
    char dollarSign;
    inputStream >> dollarSign; //hopefully
    if (dollarSign != '$')
    {
        cout << "No dollar sign in Money input.\n";
        exit(1);
    }


    double amountAsDouble;
    inputStream >> amountAsDouble;
    amount.dollars = amount.dollarsPart(amountAsDouble);
    amount.cents = amount.centsPart(amountAsDouble);


    return inputStream;
}

const Money operator +(const Money& amount1, const Money& amount2)
{
    int allCents1 = amount1.cents + amount1.dollars*100;
    int allCents2 = amount2.cents + amount2.dollars*100;
    int sumAllCents = allCents1 + allCents2;
    int absAllCents = abs(sumAllCents); //Money can be negative.
    int finalDollars = absAllCents/100;
    int finalCents = absAllCents%100;
```

```cpp
    if (sumAllCents < 0)
    {
        finalDollars = -finalDollars;
        finalCents = -finalCents;
    }

    return Money(finalDollars, finalCents);
}

//Uses cstdlib:
const Money operator -(const Money& amount1, const Money& amount2)
{
    int allCents1 = amount1.cents + amount1.dollars*100;
    int allCents2 = amount2.cents + amount2.dollars*100;
    int diffAllCents = allCents1 - allCents2;
    int absAllCents = abs(diffAllCents);

    int finalDollars = absAllCents/100;
    int finalCents = absAllCents%100;

    if (diffAllCents < 0)
    {
        finalDollars = -finalDollars;
        finalCents = -finalCents;
  }

    return Money(finalDollars, finalCents);
}

bool operator ==(const Money& amount1, const Money& amount2)
{
    return ((amount1.dollars == amount2.dollars)
           && (amount1.cents == amount2.cents));
}

const Money operator -(const Money& amount)
{
    return Money(-amount.dollars, -amount.cents);
}


Money::Money( ): dollars(0), cents(0)
{/*Body intentionally empty.*/}

Money::Money(double amount)
              : dollars(dollarsPart(amount)), cents(centsPart(amount))
{/*Body intentionally empty*/}

Money::Money(int theDollars)
              : dollars(theDollars), cents(0)
{/*Body intentionally empty*/}

//Uses cstdlib:
Money::Money(int theDollars, int theCents)
```

```
{
    if ((theDollars < 0 && theCents > 0) || (theDollars > 0 && theCents <
     0))
    {
        cout << "Inconsistent money data.\n";
        exit(1);
    }
    dollars = theDollars;
    cents = theCents;
}

double Money::getAmount( ) const
{
    return (dollars + cents*0.01);
}

int Money::getDollars( ) const
{
    return dollars;
}

int Money::getCents( ) const
{
    return cents;
}

int Money::dollarsPart(double amount) const
{
    return static_cast<int>(amount);
}

int Money::centsPart(double amount) const
{
    double doubleCents = amount*100;
    int intCents = (round(fabs(doubleCents)))%100;//% can misbehave on
     negatives
    if (amount < 0)
        intCents = -intCents;
    return intCents;
}

int Money::round(double number) const
{
    return static_cast<int>(floor(number + 0.5));
}
```