```cpp
//
//  main.cpp
//  AbsoluteCpp_ch7_5
//
//

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

class BankAccount
{
 public:
    BankAccount(double balance, double rate);
    BankAccount(int dollars, int cents, double rate);
    BankAccount(int dollars, double rate);
    BankAccount( );
    void update( );
    void input( );
    void output( ) const;

    double getBalance( ) const { return (accountDollars +
     accountCents*0.01);}

    int getDollars( ) const { return accountDollars; }

    int getCents( ) const { return accountCents; }

    double getRate( ) const { return rate; }

    void setBalance(double balance);
    void setBalance(int dollars, int cents);
    void setRate(double newRate);
 private:
    int accountDollars; //of balance
    int accountCents; //of balance
    double rate;//as a percent

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

    int centsPart(double amount) const;

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

    double fraction(double percent) const { return (percent/100.0); }
};

//Returns true if the balance in account1 is greater than that
//in account2. Otherwise returns false.
bool isLarger(const BankAccount& account1, const BankAccount& account2);
```

```
void welcome(const BankAccount& yourAccount);

int main( )
{
    BankAccount account1(6543.21, 4.5), account2;
    welcome(account1);
    cout << "Enter data for account 2:\n";
    account2.input( );
    if (isLarger(account1, account2))
        cout << "account1 is larger.\n";
    else
        cout << "account2 is at least as large as account1.\n";

        return 0;
}

bool isLarger(const BankAccount& account1, const BankAccount& account2)
{
    return(account1.getBalance( ) > account2.getBalance( ));
}

void welcome(const BankAccount& yourAccount)
{
    cout << "Welcome to our bank.\n"
         << "The status of your account is:\n";
    yourAccount.output( );
}

//Uses iostream and cstdlib:
void BankAccount::output( ) const
{
    int absDollars = abs(accountDollars);
    int absCents = abs(accountCents);
    cout << "Account balance: $";
    if (accountDollars < 0)
        cout << "-";
    cout << absDollars;
    if (absCents >= 10)
        cout << "." << absCents << endl;
    else
        cout << "." << '0' << absCents << endl;

    cout << "Rate: " << rate << "%\n";
}



BankAccount::BankAccount(double balance, double rate)
 : accountDollars(dollarsPart(balance)), accountCents(centsPart(balance))
{
    setRate(rate);
}

BankAccount::BankAccount(int dollars, int cents, double rate)
{
```

```
    setBalance(dollars, cents);
    setRate(rate);
}

BankAccount::BankAccount(int dollars, double rate)
                              : accountDollars(dollars), accountCents(0)
{
    setRate(rate);
}

BankAccount::BankAccount( ): accountDollars(0), accountCents(0), rate(0.0)
{/*Body intentionally empty.*/}

void BankAccount::update( )
{
    double balance = accountDollars + accountCents*0.01;
    balance = balance + fraction(rate)*balance;
    accountDollars = dollarsPart(balance);
    accountCents = centsPart(balance);
}

//Uses iostream:
void BankAccount::input( )
{
    double balanceAsDouble;
    cout << "Enter account balance $";
    cin >> balanceAsDouble;
    accountDollars = dollarsPart(balanceAsDouble);
    accountCents = centsPart(balanceAsDouble);
    cout << "Enter interest rate (NO percent sign): ";
    cin >> rate;
    setRate(rate);
}

void BankAccount::setBalance(double balance)
{
    accountDollars = dollarsPart(balance);
    accountCents = centsPart(balance);
}

//Uses cstdlib:
void BankAccount::setBalance(int dollars, int cents)
{
    if ((dollars < 0 && cents > 0) || (dollars > 0 && cents < 0))
    {
        cout << "Inconsistent account data.\n";
        exit(1);
    }
    accountDollars = dollars;
    accountCents = cents;
}

//Uses cstdlib:
void BankAccount::setRate(double newRate)
{
```

```
    if (newRate >= 0.0)
        rate = newRate;
    else
    {
        cout << "Cannot have a negative interest rate.\n";
        exit(1);
    }
}

//Uses cmath:
int BankAccount::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;
}
```