**Homework 2 – CS60 Linnell**

**Arjun Kohli**

**W1579330**

**Problem 1:**

**Header File:**

#ifndef PROBLEM1\_H

#define PROBLEM1\_H

#include <string>//ADT Style: no using namespace std;

#include <iostream>

class Rational{

public:

Rational(int n, int d);

Rational();

int numer(){ return numer\_; }//ADT Style:give the getter the variable name minus the \_

int denom(){ return denom\_; }//ADT Style:Don't bother with forward declaration for one-liner

void operator \*=(Rational b);

void operator +=(Rational b);

private:

int numer\_;//ADT Style: trailing \_ means this is a private variable

int denom\_;

};

//pre: none

//post: returns a Rational that is the result of multiplying this Rational and the Rational passed in.

//The result is not reduced; the denominator will be the product of the denominators of the operands.

Rational operator \*(Rational a, Rational b);

//pre: none

//post: returns a Rational that is the result of adding this Rational and the Rational passed in.

//The result is not reduced; the denominator will be the product of the denominators of the operands.

Rational operator +(Rational a, Rational b);

std::ostream& operator <<(std::ostream& out, Rational a);

//The problem was that I was missing the std:: in front of ostream

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

//pre: none

//post: returns true if same, returns false if different

bool operator ==(Rational a, Rational b);

#endif

**Class File:**

#include "problem1.h"

Rational::Rational(int n, int d){//We need the Rational:: because it looks like we're defining this at the global scope, but it BELONGS in Rational's scope.

numer\_ = n;

denom\_ = d;

}

Rational::Rational(){

numer\_ = 0;

denom\_ = 1;

}

Rational operator \*(Rational a, Rational b){

Rational ans(a.numer()\*b.numer(), a.denom()\*b.denom());

ans.reduce();

return ans;

}

void Rational::operator \*=(Rational b){

numer\_ = (numer\_\*b.numer\_);

denom\_ = (denom\_\*b.denom\_);

reduce();

}

Rational operator +(Rational a, Rational b){

Rational ans((a.numer()\*b.denom()+b.numer()\*a.denom()), (a.denom()\*b.denom()));

ans.reduce();

return ans;

}

std::ostream& operator <<(std::ostream& out, Rational a){

out<<a.numer()<<"/"<<a.denom()<<std::endl;

return out;

}

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

//a

bool operator ==(Rational lhs, Rational rhs) {

if (lhs.numer() == rhs.numer() && lhs.denom() == rhs.denom()) {

return true;

}

else {

if (lhs.numer()/rhs.numer() == lhs.denom()/lhs.denom() || rhs.numer()/lhs.numer() == rhs.denom()/lhs.denom()) {

return true;

}

}

return false;

}

//b

void Rational::operator +=(Rational& rhs) {

numer\_ = numer\_ + rhs.numer();

denom\_ = denom\_ + rhs.denom();

reduce();

}

//c

void Rational::reduce() {

for (int i = numer\_; i > 0; i--) {

if (denom\_ % i == 0 && numer\_ % i == 0) {

denom\_ = denom\_/i;

numer\_ = numer\_/i;

}

}

}

**main.cpp:**

#include "problem1.h"

#include <iostream>

using namespace std;

int main() {

Rational r1 = Rational(1, 2);

Rational r2(3, 4);

Rational r3 = r1 + r2;

cout << r3 << endl;

cout << (r1==r2) << endl;

r3+=r1;

cout << r3 << endl;

}

**Output:**

5/4

0

7/4

**Problem 2:**

**Header File:**

#ifndef ATM\_H

#define ATM\_H

class Atm {

private:

int twenties\_;

int tens\_;

public:

Atm(int t20, int t10);

//Pre: t20 >= 0, t10 >= 0

//Post: Sets twenties\_ = t20, tens\_ = t10

Atm();

//Pre: None

//Post: Sets twenties\_ = 0, tens\_ = 0

int get\_twenties() {return twenties\_;}

//Pre: None

//Post: Returns twenties\_

int get\_tens() {return tens\_;}

//Pre: None

//Post: Returns tens\_

BankAccount get\_cash(int amount, BankAccount b);

//Pre: amount < (20\*twenties\_ + 10\*tens\_), amount < balance

//Post: balance - amount, twenties\_ - total/20, tens\_ - 1

void operator +=(Atm& rhs);

//Pre: rhs.get\_twenties() >= 0, rhs.get\_tens() >= 0

//Post: twenties\_ += rhs.get\_twenties(), tens\_ += rhs.get\_tens()

void restock(int new20s, int new10s);

//Pre: new20s >= 0, new10s >= 0

//Post: Adds new20s to twenties\_, Adds new10s to tens\_

};

bool operator ==(Atm lhs, Atm rhs);

std::ostream& operator <<(std::ostream& out, Atm a);

#endif

**Class File:**

#include "atm.h"

#include "bankaccount.h"

Atm::Atm() {

twenties\_ = 0;

tens\_ = 0;

}

Atm::Atm(int t20, int t10) {

twenties\_ = t20;

tens\_ = t10;

}

BankAccount Atm::get\_cash(int amount, BankAccount b) {

if (amount < b.balance) {

int total = amount;

if ((get\_twenties() - total/20) < 0) {

std::cout << "Not enough bills" << std::endl;

} else {

twenties\_ = get\_twenties() - total/20;

}

if (total % 20 != 0) {

if ((get\_tens() - 1) < 0) {

std::cout << "Not enough bills" << std::endl;

} else {

tens\_ = get\_tens() - 1;

}

}

b.balance = b.balance - (20 \* get\_twenties()) - (10 \* get\_tens());

return b;

} else {

std::cout << "Invalid amount" << std::endl;

}

}

void Atm::operator +=(Atm& rhs) {

twenties\_ = get\_twenties() + rhs.get\_twenties();

tens\_ = get\_tens() + rhs.get\_tens();

}

void Atm::restock(int new20s, int new10s) {

twenties\_ += new20s;

tens\_ += new10s;

}

bool operator ==(Atm lhs, Atm rhs) {

return (lhs.get\_twenties() == rhs.get\_twenties() && lhs.get\_tens() == rhs.get\_tens());

}

std::ostream& operator <<(std::ostream&out, Atm a) {

out << "20 dollar bills: " << a.get\_twenties() << std::endl;

out << "10 dollar bills: " << a.get\_tens() << std::endl;

return out;

}

**main.cpp:**

#include "atm.h"

#include "bankaccount.h"

#include <iostream>

using namespace std;

int main() {

Atm a(300, 500);

Atm b(400, 600);

cout << a << endl;

cout << (a==b) << endl;

a+=b;

cout << a << endl;

a.restock(100, 100);

cout << a << endl;

return 0;

}

**Output:**

20 dollar bills: 300

10 dollar bills: 500

0

20 dollar bills: 700

10 dollar bills: 1100

20 dollar bills: 800

10 dollar bills: 1200