Name : Abhishek Sunil Tiwari

Reg no : 18BCE1156

1

#include <iostream>

using namespace std;

class Fraction {

private:

// Calculates the greates common divisor with

// Euclid's algorithm

// both arguments have to be positive

long long gcd(long long a, long long b) {

while (a != b) {

if (a > b) {

a -= b;

} else {

b -= a;

}

}

return a;

}

public:

long long numerator, denominator;

Fraction() {

numerator = 0;

denominator = 1;

}

Fraction(long long n, long long d) {

if (d==0) {

cerr << "Denominator may not be 0." << endl;

exit(0);

} else if (n == 0) {

numerator = 0;

denominator = 1;

} else {

int sign = 1;

if (n < 0) {

sign \*= -1;

n \*= -1;

}

if (d < 0) {

sign \*= -1;

d \*= -1;

}

long long tmp = gcd(n, d);

numerator = n/tmp\*sign;

denominator = d/tmp;

}

}

operator int() {return (numerator)/denominator;}

operator float() {return ((float)numerator)/denominator;}

operator double() {return ((double)numerator)/denominator;}

};

Fraction operator+(const Fraction& lhs, const Fraction& rhs) {

Fraction tmp(lhs.numerator\*rhs.denominator

+rhs.numerator\*lhs.denominator,

lhs.denominator\*rhs.denominator);

return tmp;

}

Fraction operator+=(Fraction& lhs, const Fraction& rhs) {

Fraction tmp(lhs.numerator\*rhs.denominator

+rhs.numerator\*lhs.denominator,

lhs.denominator\*rhs.denominator);

lhs = tmp;

return lhs;

}

Fraction operator-(const Fraction& lhs, const Fraction& rhs) {

Fraction tmp(lhs.numerator\*rhs.denominator

-rhs.numerator\*lhs.denominator,

lhs.denominator\*rhs.denominator);

return tmp;

}

Fraction operator-=(Fraction& lhs, const Fraction& rhs) {

Fraction tmp(lhs.numerator\*rhs.denominator

-rhs.numerator\*lhs.denominator,

lhs.denominator\*rhs.denominator);

lhs = tmp;

return lhs;

}

Fraction operator\*(const Fraction& lhs, const Fraction& rhs) {

Fraction tmp(lhs.numerator\*rhs.numerator,

lhs.denominator\*rhs.denominator);

return tmp;

}

Fraction operator\*=(Fraction& lhs, const Fraction& rhs) {

Fraction tmp(lhs.numerator\*rhs.numerator,

lhs.denominator\*rhs.denominator);

lhs = tmp;

return lhs;

}

Fraction operator\*(int lhs, const Fraction& rhs) {

Fraction tmp(lhs\*rhs.numerator,rhs.denominator);

return tmp;

}

Fraction operator\*(const Fraction& rhs, int lhs) {

Fraction tmp(lhs\*rhs.numerator,rhs.denominator);

return tmp;

}

Fraction operator/(const Fraction& lhs, const Fraction& rhs) {

Fraction tmp(lhs.numerator\*rhs.denominator,

lhs.denominator\*rhs.numerator);

return tmp;

}

std::ostream& operator<<(std::ostream &strm, const Fraction &a) {

if (a.denominator == 1) {

strm << a.numerator;

} else {

strm << a.numerator << "/" << a.denominator;

}

return strm;

}

int main() {

Fraction a(1,3);

Fraction b(3,28);

Fraction c;

c = a + b;

cout << c << "\t(should be 37/84)" << endl;

c = a - b;

cout << c << "\t(should be 19/84)" << endl;

c = a \* b;

cout << c << "\t(should be 1/28)" << endl;

c = a / b;

cout << c << "\t(should be 28/9)" << endl;

c = -1 \* b;

cout << c << "\t(should be -3/28)" << endl;

c = b \* (-1);

cout << c << "\t(should be -3/28)" << endl;

c = Fraction(-100,3);

cout << (int)c << "\t(should be -33)" << endl;

cout << (float)c << "\t(should be -33.3...)" << endl;

cout << (double)c << "\t(should be -33.3...)" << endl;

a -= b;

cout << a << "\t(should be 19/84)" << endl;

return 0;

}

2

#include<iostream>

using namespace std;

class point

{

Static int x, y;

Public:

Point ()

{

x++;

y++;

}

Void show(){

Cout<<x<<y;

}

};

int main()

{

Point o1;

}