#include<iostream>

using namespace std;

class num {

private:

int a;

int b;

public:

num() {

a = 0;

b = 1;

}

num(int valueA, int valueB) {

a = valueA;

b = valueB;

}

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

void setA(int a) {

this->a = a;

}

void setB(int b) {

this->b = b;

}

int getA() {

return a;

}

int getB() {

return b;

}

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

bool operator ==(const num& other) {

return (this->a / this->b) == (other.a / other.b);

}

bool operator !=(const num& other) {

return !((this->a / this->b) == (other.a / other.b));

}

bool operator >(const num& other) {

return (this->a / this->b) > (other.a / other.b);

}

bool operator <(const num& other) {

return (this->a / this->b) < (other.a/other.b);

}

bool operator >=(const num& other) {

return (this->a / this->b) >= (other.a / other.b);

}

bool operator <=(const num& other) {

return (this->a / this->b) <= (other.a / other.b);

}

num operator +(num & other) {

return num(this->a \* other.b + this->b \* other.a, this->b \* other.b);

}

num operator -(num& other) {

return num(this->a \* other.b - this->b \* other.a, this->b \* other.b);

}

num operator \*(num& other) {

return num(this->a \* other.a, this->b \* other.b);

}

num operator /(num& other) {

return num(this->a \* other.b, this->b \* other.a);

}

num & operator ++() {

a += b;

return \*this;

}

num & operator ++(int) {

num temp(\*this);

this->a += this->b;

return temp;

}

num& operator --() {

a -= b;

return \*this;

}

num& operator --(int) {

num temp(\*this);

this->a -= this->b;

return temp;

}

friend istream& operator >>(istream& is, num& other) {

is >> other.a >> other.b;

return is;

}

friend ostream& operator <<(ostream& out, num& other) {

out << " " << other.a << endl << "---" << endl << " " << other.b << endl << endl;

return out;

}

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

void optimize() {

for (int i = max(a, b); i > 1; i--) {

if (a % i == 0 && b % i == 0) {

a /= i;

b /= i;

i = max(a, b);

}

}

}

void showContent() {

cout << " " << a << endl << "---" << endl << " " << b << endl << endl;

}

};

void main() {

num a(1, 200);

num b(2, 6);

}