```cpp
/*
280
Complex 1
Elliot Shaw
*/

#include <iostream>
using namespace std;

class Complex {
private:
    double rPart, iPart;
public:
    //constructors
    Complex();
    Complex(double, double);

    //getters/accessors
    //double getReal();
    double getReal();
    //double getImag();
    double getImag();

    //setters/mutators
    void set(double, double);
    //void  setReal(double);
    void setReal(double);
    //void setImag(double);
    void setImag(double);

    //utility
    //bool equal(Complex);
    //friend bool equal(Complex, Complex);
    void display();
    void load();

    //arithmetic
    void add(Complex);
    void sub(Complex);
    void mult(Complex);
    friend Complex add(const Complex, const Complex);
    friend Complex sub(const Complex, const Complex);
    friend Complex mult(const Complex, const Complex);
    friend Complex operator + (const Complex, const Complex);
    friend Complex operator - (const Complex, const Complex);
    friend Complex operator * (const Complex, const Complex);
    friend ostream& operator << (ostream&, const Complex&);
    friend istream& operator >> (istream&, Complex&);
};
```

```cpp

Complex::Complex() {
    rPart = iPart = 0;
} //Complex

Complex::Complex(double rVal, double iVal) {
    rPart = rVal;  iPart = iVal;
} //Complex

//Accessors
double Complex::getReal() {
    return rPart;
}//getReal

double Complex::getImag() {
    return iPart;
}//getImag

//setters
void Complex::set(double rVal, double iVal) {
    rPart = rVal;  iPart = iVal;
} //set

void Complex::setReal(double newR) {
    rPart = newR;
}//setReal

void Complex::setImag(double newI) {
    iPart = newI;
}//setImag


//utility
void Complex::load() {
    cout << "Enter rp and ip: ";
    cin >> rPart >> iPart;
} //load

void Complex::display() {
    cout << rPart;
    if (iPart >= 0) cout << " + " << iPart;
    else cout << " - " << -iPart;
    cout << "i" << endl;
} //display


//math
void Complex::add(const Complex val) {
    rPart += val.rPart;
    iPart += val.iPart;
} //add

Complex add(const Complex v1, const Complex v2) {
    Complex t = Complex();
    t.rPart = v1.rPart + v2.rPart;
    t.iPart = v1.iPart + v2.iPart;
    return t;
} //add friend

Complex operator + (const Complex v1, const Complex v2) {
    return add(v1, v2);
} //+

void Complex::sub(const Complex val) {
    rPart -= val.rPart;
    iPart -= val.iPart;
} //sub

Complex sub(const Complex v1, const Complex v2) {
    Complex t = Complex();
    t.rPart = v1.rPart - v2.rPart;
    t.iPart = v1.iPart - v2.iPart;
    return t;
} //sub friend

Complex operator - (const Complex v1, const Complex v2) {
    return sub(v1, v2);
} //-

void Complex::mult(const Complex val) {
    double finalR, finalI;
    finalR = (rPart * val.rPart) + (iPart * val.iPart);
    finalI = (rPart * val.iPart) + (iPart * val.rPart);
    rPart = finalR;
    iPart = finalI;
}//mult

Complex mult(const Complex v1, const Complex v2) {
    Complex t = Complex();
    t.rPart = (v1.rPart * v2.rPart) + (v1.iPart * v2.iPart);
    t.iPart = (v1.rPart * v2.iPart) + (v1.iPart * v2.rPart);
    return t;
}//mult friend

Complex operator * (const Complex v1, const Complex v2) {
    return mult(v1, v2);
}//*

```

```cpp
ostream& operator << (ostream& out, const Complex& v)
{
    out << v.rPart;
    if (v.iPart >= 0) out << " + " << v.iPart;
    else out << " - " << -v.iPart;
    out << 'i';
    return out;
}//<<

istream& operator >> (istream& in, Complex& v)
{
    cout << "Enter rPart ";
    in >> v.rPart;
    cout << "Enter iPart ";
    in >> v.iPart;
    return in;
}//>>

int main() {
    Complex c1 = Complex(1, -1);
    Complex c2 = Complex(3, -4);

    c1.add(c2);
    cout << c1 << endl;
    c1 = c1 - c2;
    cout << c1 << endl;

    c1.sub(c2);
    cout << c1 << endl;
    c1 = c1 + c2;
    cout << c1 << endl;

    Complex c3 = add(c1, c2);
    cout << c3 << endl;

    c3 = sub(c1, c2);
    cout << c3 << endl;

    c3 = mult(c1, c2);
    cout << c3 << endl;

    c1 = c1 * c2;
    cout << c1 << endl;

    Complex c4;
    cin >> c4;

    cout << c4;
} //main
```