```cpp
#include <iostream>
using namespace std;

class comp {
private:
   double real;
   double imag;

public:
   comp();
   void setpara(double r, double im);
   void print() const;

   // Overloaded insertion and extraction operators
   friend ostream& operator<<(ostream& out, const comp& c);
   friend istream& operator>>(istream& in, comp& c);

   comp operator+(const comp&);
   comp operator*(const comp&);
   bool operator==(const comp&);
   comp& operator=(const comp&);
};

comp::comp() : real(0.0), imag(0.0) {}

void comp::setpara(double r, double im) {
   real = r;
   imag = im;
}

void comp::print() const {
   cout << real << " + " << imag << "i" << endl;
}

// Overloaded insertion operator definition
ostream& operator<<(ostream& out, const comp& c) {
   out << c.real << " + " << c.imag << "i";
   return out;
}

// Overloaded extraction operator definition
istream& operator>>(istream& in, comp& c) {
   cout << "Enter real part: ";
   in >> c.real;
   cout << "Enter imaginary part: ";
```

```
    in >> c.imag;
    return in;
}

// Operator overloading definitions (unchanged)
comp comp::operator+(const comp& a) {
    comp temp;
    temp.real = real + a.real;
    temp.imag = imag + a.imag;
    return temp;
}

comp comp::operator*(const comp& a) {
    comp result;
    result.real = (real * a.real) - (imag * a.imag);
    result.imag = (real * a.imag) + (imag * a.real);
    return result;
}

bool comp::operator==(const comp& ap2) {
    return (real == ap2.real) && (imag == ap2.imag);
}

comp& comp::operator=(const comp& ap2) {
    if (this != &ap2) {
        real = ap2.real;
        imag = ap2.imag;
    }
    return *this;
}

int main() {
    comp n1, n2, n3;

    // Using extraction operator to input values
    cin >> n1;
    cin >> n2;

    // Printing n1 and n2
    cout << "n1: " << n1 << endl;
    cout << "n2: " << n2 << endl;

    n3 = n1;
    n3.print();
```

```
    (n1 + n2).print();

    n3 = n1 * n2;
    n3.print();

    if (n1 == n2)
        cout << "Both numbers have the same real and imag part." << endl;
    else
        cout << "Unequal!!!" << endl;

    return 0;
}
```