assgn5_D1171708 Brian

In this assignment. Firstly. we have to define the real part and the image part with the private.

```
class Complex{
    private:
        double re; // real part
        double im; // image part
```

Next, we have to complete each class to use the overloading on complex operation

```
public:
    Complex();
    Complex(double re, double im);
    double getRe();
    double getIm();
    void setRe(double re);
    void setIm(double im);

    Complex operator-();//unary operator
    Complex operator+(const Complex &c);//Complex
    Complex operator-(const Complex &c);
    Complex operator*(const Complex &c);
    Complex operator/(const Complex &c);
    bool operator==(const Complex &c);
    bool operator!=(const Complex &c);
```

Use the friend function to consider the different order of complex+double and double+complex

```
friend Complex operator+(const Complex &c,const double d); //Complex+double
friend Complex operator-(const Complex &c,const double d);
friend Complex operator*(const Complex &c,const double d);
friend Complex operator/(const Complex &c,const double d);
friend bool operator==(const Complex &c,const double d);
friend bool operator!=(const Complex &c,const double d);

friend Complex operator+(const double d,const Complex &c);//double + Complex
friend Complex operator-(const double d,const Complex &c);
friend Complex operator*(const double d,const Complex &c);
friend Complex operator/(const double d,const Complex &c);
friend bool operator==(const double d,const Complex &c);
friend bool operator!=(const double d,const Complex &c);

double abs();
double square();
Complex& operator=(const Complex &c);
Complex& operator+=(const Complex &c);
Complex& operator-=(const Complex &c);
Complex& operator*=(const Complex &c);
Complex& operator/=(const Complex &c);

friend ostream& operator<<(ostream &out, const Complex &c);
friend istream& operator>>(istream &in,  Complex &c);
```

Finish all the classes defined in complex overloading.h in complex overloading C++

```
Complex root1, root2;
Complex a_complex(a,0.0);
Complex b_complex(b,0.0);
Complex c_complex(c,0.0);
Complex d_complex = b_complex*b_complex - Complex(4.0, 0.0)*a_complex*c_complex;

if(d_complex.getRe()>0){
    root1 =  Complex(0.0,0.0) - (b_complex - d_complex.square())/(Complex(2.0, 0.0)*a_complex);
    root2 =  Complex(0.0,0.0) - (b_complex + d_complex.square())/(Complex(2.0, 0.0)*a_complex);
}
if(d_complex.getRe()==0){
    root1 =  Complex(0.0,0.0) - (b_complex)/(Complex(2.0, 0.0)*a_complex);
    root2 =  Complex(0.0,0.0) - (b_complex)/(Complex(2.0, 0.0)*a_complex);
}

if(d_complex.getRe()<0){
    d_complex = Complex(0.0,0.0)-(d_complex);
    root1 = Complex(0.0,0.0) - (b_complex - Complex(0.0,1.0)*d_complex.square())/(Complex(2.0, 0.0)*a_complex);
    root2 = Complex(0.0,0.0) - (b_complex + Complex(0.0,1.0)*d_complex.square())/(Complex(2.0, 0.0)*a_complex);
}
```

Define the root1 and root2 as a complex number then use the overloading operation of the complex function I made to calculate the result and verify.

```
Complex verify1 = a_complex * root1 * root1 + b_complex * root1 + c_complex;
Complex verify2 = a_complex * root2 * root2 + b_complex * root2 + c_complex;


cout << "    " << root1 <<" and "<<root2 << endl;
if (verify1.abs() < 0.000001)
    if (verify2.abs() < 0.000001){
        cout << "Verification of the two quadratic equation roots PASSES."<<endl;
    }
else{
    cout << "Verification of the two quadratic equation roots FAILED."<<endl;
}
```

Verify the result by plugging the root into the Quadratic Equation a*root1^2 + b*root1 + c= 0 and use abs to get the number if < 10^-6 will give the verification succeeds.