CS173: Intermediate Programming

Complex ADT

Overview
The Complex data structure stores and manipulates complex numbers of the form $a + bi$ where a and b are real numbers. This Complex type supports a multitude of arithmetic operations pertinent to complex numbers; these operations are described below.

Constructors	
default	A new Complex data type should default to the value of $0 + 0i$. `Complex c;`
copy	Create a new Complex type from an existing one. `Complex c1(c2);`
Complex(a,b)	We should be able to specify a new Complex type by giving its real and imaginary components. The imaginary part defaults to 0. `Complex c1(2,3.1); Complex c2(2.5);`

Operators	
addition	Should support addition between two complex numbers, complex and int, and complex and float. `c1 = c2 + c3; c1 = c2 + 5.5;`
subtraction	Should support subtraction between two complex numbers, complex and int, and complex and float. `c1 = c2 - c3; c1 = c2 - 5;`
multiplication	Should support multiplication between two complex numbers, complex and int, and complex and float. `c1 = c2 * c3; c1 = c2 * 5.5;`
division	Should support division between two complex numbers, complex and int, and complex and float. `c1 = c2 / c3; c1 = c2 / 5;`
conjugate	The $\sim$ operator returns the complex conjugate. `c1 = ∼c2;`
negation	The $-$ operator returns the negative of a complex. `c1 = -c2;`
exponentiation	The $\wedge$ operator should raise a complex number to an integer power. `c1 = c2∧x;` (where x is integer only)
abs	The abs method should return the distance from the origin. `c1.abs()`

Modifiers and Accessors	
setReal	Sets the real part of the complex number. `c1.setReal(5);`
getReal	Gets the real part of the complex number. `float f = c1.getReal();`
setImag	Sets the imaginary part of the complex number. `c1.setImag(5);`
getImag	Gets the imaginary part of the complex number. `float f = c1.getImag();`

Other	
assignment operator	Allows assignment of values between complex numbers. `c1 = c2;`
destructor	Cleans up the complex class.
equality operator	Equal if both real and imaginary parts are equal. `c1 == c2;`
inequality operator	True if either real or imaginary parts are not equal. `c1 != c2;`
greaterThan operator	Returns true if abs(c1) > abs(c2), false otherwise. `c1 > c2;`
greaterEqual operator	Returns true if abs(c1) >= abs(c2), false otherwise. `c1 >= c2;`
lessThan operator	Returns true if abs(c1) < abs(c2), false otherwise. `c1 < c2;`
lessEqual operator	Returns true if abs(c1) <= abs(c2), false otherwise. `c1 <= c2;`
`cout <<`	Allows printing of a complex number as a string ”$a + bi$”. `cout << c1 << endl;` print `a+0i` as a print `0+bi` as bi print `a+-bi` as a-bi
`cin>>`	Allows reading of a complex number as a string ”a+bi”. `cin >> c1;` reads: `a+bi, a-bi, -a+bi, -a-bi, a+-bi, +a+bi` reads: `a, -a, +a, bi, -bi, +bi` where `a,b` can be integers or reals with decimal points