CS173 Intermediate Computer Science
Project 8: Complex ADT

OVERVIEW

This is a group project. You will work in a pre-assigned team of three or four students sharing the workload equally to complete this project. It is not acceptable for one or two members of the team to complete the majority of the work. Your team should manage the project with an agreed upon timeline and division of labor. Use the provided group contract template to set these expectations.

Complex ADT

The goal of this project is to implement the Complex number Abstract Data Type in a C++ class that we design and build. You are given an ADT design document that describes the target class. Our class will build a new numeric datatype that holds complex numbers of the form a+bi where a is called the real part of the complex number and b is the imaginary part.

For background on complex numbers (especially arithmetic), watch these videos from Khan Academy:

Part 1 Khan Academy Video on Complex Numbers
https://www.youtube.com/watch?v=kpywdu1afas

Part 2 Khan Academy Video on Complex Numbers
https://www.youtube.com/watch?v=bPqB9a1uk_8

LOGISTICS

I have given you a Complex.h header file. **You are to use this file and not make any changes to it.** It contains the class specification (the class declaration). Your job is to create the Complex.cpp implementation file that accompanies this Complex.h file. You will need to implement all the functions in the ADT and in the Complex.h class declaration.

I have also given you a driver main.cpp file which calls and tests some of the Complex class. You may make changes to this file if you wish to perform additional testing.

I suggest that you start by commenting out most of the function calls to Complex methods in the main driver program, and then comment statements back in (uncommenting them), one at a time, as you implement them. It is often best to start with the constructors, then add the ability to print (the cout method). Since the full-blown correct cout method is actually quite tricky with all the special cases, you can create a "quick" cout function which displays the information but maybe not in the right format. You can go back later to complete the cout method with the correct formatting.

Then you can add in one method at a time, test it, and move on in the same way we did with the Rational class. **I suggest you resist the temptation to implement multiple methods at a time and then test them all at once.**

I have also given you a makefile that you can use to compile your project. You can make changes to the makefile if you need to compile other files.

You will submit ONLY the Complex.cpp file.

All methods should be properly commented with:

- Method name
- **Name of developer** (person who coded the method)
- Description of what the method does
- Parameter list – type and any constraints of the parameters
- Return value – type and description of the return value. Note where calling objects are changed since this is a "hidden return".

Download all the files and put them in a directory called Complex. This way you have a separate directory for this project.

```
unzip complex.zip
```

Compile with

```
make
```

Run with

```
./complex
```

PROJECT MANAGEMENT

I suggest nominating a team leader. Start the project by assigning tasks to each team member and set up a timeline where you all agree to meet your part of the deadline. Save ample time for combining the work and for testing.

Here is one way to divide up the work among three people:

Member 1	Member 2	Member 3
manage group chat	create test suite	lead team meetings
default constructor	copy constructor	constructor (a,b)

setReal	getReal	destructor
getImag	op ==	setImag
assignment operator	op+ with float	op !=
op+ with Complex	op- with Complex	op+ with int
op- with float	op* with Complex	op- with int
op* with int	op/ with float	op* with float
op/ with int	cin >>	op/ with Complex
op^	negation	cout <<
conjugate		abs

Be sure that each person completes the tasks they are responsible for. Other members of the team can provide conceptual assistance but must not provide code.

TESTING

In the example above, team members took on a responsibility other than programming the core functionality of the Complex class. These responsibilities are an important part of the software development process.

Team Member 2 will create an extensive test suite that is designed to test each method. They will want to augment the main.cpp file extensively. Think hard about how to verify that each method works correctly. There are many special cases to consider, especially with cin and cout. The team will have to collaborate with a schedule that allows for proper testing time.

Team Member 1 will manage the project group chat, arranging times to meet and helping set deadlines.

Team Member 3 will lead group meetings, setting the agenda and making sure everyone has a chance to contribute to those meetings.

LEVELS OF ACHIEVEMENT

- D-level: An incomplete (but still well-documented) Complex class. Points will be awarded for progress made toward implementing the class.
- C-level: A well-documented Complex class that has some serious logical errors in implementation.
- B-level: A well-documented Complex class that nearly correctly implements all methods as specified but perhaps has some logical errors or formatting errors.
- A-level: A well-documented Complex class that correctly implements all the constructors and methods as specified here, in the ADT document, and the given specification (h) file.