# CSS342 Data Structures, Algorithms, and Discrete Mathematics I

**Autumn 2020**

**Assignment 1: Objects and Classes**
**Due date: Friday 17 Apr**

## Goal

This programming assignment exercises how to construct abstract data types through implementing a **complex** class in C++. It also reviews operator overloading, and input/output including the friend concept.

## Overview of complex Class

The **complex** class presents the complex number **X+Y*i***, where **X** and Y are real numbers and *i^2* is -1. Typically, X is called a real part and Y is an imaginary part of the complex number. For instance, **complex(4.0, 3.0)** means 4.0+3.0*i*. The **complex** class you will design should have the following features.

**Constructor**
Only one constructor with default value for Real = 0.0, Imaginary = 0.0; But it should construct the Complex numbers with, 1) no argument, 2) 1 argument, 3) 2 arguments.

**Data members**
The Complex class should have two members which should be represented with real values. For example, x+yi, Real = x, Imaginary = y.

**Member functions**
Provide the following two member functions:
**getReal**
This returns the real part of the complex number.
**getImaginary**
This returns the imaginary part of the complex number.

**Math operators**
The class must implement binary arithmetic operations such as addition, subtraction, multiplication, and division. You should be able to use operators (+, -, *, /).

**Addtion (+)**
Add two objects. For example, (a+bi) + (c+di) = (a+c) + (b+d)i.

**Subtraction(-)**
Performs a complex minus operation.

**Multiplication(*)**
Multiply the left-hand-side object by the right-hand-side and return **Complex** object.

**Division(/)**
Divide the left-hand-side object by the right-hand-side object. You have to know how to divide two complex numbers. If you don't know how, try googling it. (Hint: It's helpful to implement the conjugate function first.) **Division by zero error should be handled** in this method. That is, print an error message, and return the original left-hand-side object. Since it is an exception error, it shouldn't affect the following tasks. (Same for /= method)

**Conjugate**
The complex conjugate of the complex number z = x + yi is defined to be x − yi. This function returns the conjugate of the input complex.

**Comparison**
The class must be able to compare two complex numbers with these operators: **==**, !=.

**Assignment**
The class must implement these operators: **+=**, **-=**, ***=** and **/=**. (Division by zero error should be handled in **/=** method.)

**Stream I/O**
The class must implement the **<<** and **>>** operators.

**Output**
The format will be: **X+Y*i***, where **X** is a real value and **Y** is an imaginary one. Of course, if either **X** or **Y** is **0**, it should not be displayed. However, if both **X** and **Y** are **0**, the output should be **0**. Also note that if **X** is 1, it should be printed out as **1**, and that if **Y** is 1, its output should be ***i***. **Wrong examples:** 1+0 *i*, 0+2*i*, 1*i*, etc.. **In the case of Y being negative, it should not have "+" between the two. For example, print 2-3*i*, instead of 2+-3*i*.**

## Statement of Work
Design and implement a **complex** class according to the specification outlined below. The **main()** function in complexDriver.cpp is used to test your code.

## Deliverables

Clearly state in your code comments any other assumptions you have made. Assumptions about **complex** members are placed in the class definition (.h file).
Turn in:

1) **output (as text file, or captured image of console ): This is the result with a given test file (complexDriver.cpp) above.**
2) **complex.h file**
3) **complex.cpp file**
4) **(optional) You can submit your own complexDriver.cpp if your implementation is incomplete and won't work with the supplied driver file.**

**If your program is not compiled even with your own driver file, you will get zero points no matter what.**
**If you forgot to turn in your own driver file, and yours is not compiled with the provided driver file (complexDriver.cpp), then you get zero points.**
**If your program is compiled with the provided driver file (complexDriver.cpp), then you do not need to submit your own driver file.**
Make sure that your program can be compiled and executed on Linux.

**Grading Guide**
1. Correctness (16)
Compilation errors in linux (0)
Successful compilation(6)   + Correct output(10)
-1 per a wrong output or a wrong implementation of each operator.

2. Program Organization (4)
Proper comments (**Every method has to be commented** in the header and implementation file)
Good (2)   Poor(1)   No explanations(0)
Coding style (**proper identations, blank lines, variable names, and non-redundant code**)
Good (2)   Poor(1)   No explanations(0)