ECE 2036 Fall 2017

Lab 4: Complex and ComplexMatrix classes

Assigned: October 6, 2017 *Due October 25, 2017*

In this lab, you will be coding two classes with overloaded operators: a Complex class to represent complex numbers, and a ComplexMatrix class to represent square matrices populated with complex numbers. You must use composition; i.e. the ComplexMatrix class must include a private member which is a 2d array of Complex objects, along with a private data member which is an integer representing the size of the matrix.

The code you write should be contained in four files, named as follows: Complex.h and ComplexMatrix.h, which will contain the class definitions; and Complex.cpp and ComplexMatrix.cpp, which will contain the function implementations. I have coded a main.cpp file which is designed to test the functionality of your classes. You can download it from t-square. Your completed code will be submitted using the turnin script on Deepthought; therefore, it must compile and run correctly on Deepthought.

Feel free to modify main.cpp as needed while coding your classes. You should write and test a single function at a time, so I recommend commenting out most of main.cpp and uncommenting only the parts that utilize the class components that you have already written. Alternatively, you could write your own main.cpp to test your classes. However, for the purposes of grading, the TA will use an unmodified copy of the main.cpp file provided on t-square and compile it along with your code. The TA will then check that the combined program (my main.cpp plus your four class files) provides the correct output in response to values they input. Sample output is provided below; obviously the TA or grader will use different inputs when testing your code.

Below is a table listing all of the public member functions and overloaded operators that your classes must provide. You may provide more if desired.

Class Complex

Function or operator	Functionality
constructor	Takes two double precision arguments which are assigned to the real and imaginary parts of the Complex variable. Without an argument, the values should default to zero.
>>	Stream a complex value entered at the command prompt into the Complex variable. The complex value must be entered in the format shown in the sample below.
<<	Stream the Complex variable to the output in

	the format shown in the sample below.
-- (predecrement)	Subtract one from the real part of the Complex variable, immediately.
-- (postdecrement)	Subtract one from the real part of the Complex variable, after executing the rest of the statement in which it is contained.
+ (double + complex)	Add a double precision number to the real part of a Complex variable (double precision before the plus sign).
+ (complex + double)	Add a double precision number to the real part of a Complex variable (double precision after the plus sign).
+ (complex + complex)	Add two Complex variables.
* (complex * complex)	Multiply two Complex variables.
abs()	Return the absolute value of a Complex variable as a double precision number.

Class ComplexMatrix

Function or operator	Functionality
default constructor	Takes a single integer argument (with default value zero) which is the desired size of the square matrix. The constructor must dynamically allocate memory to create the matrix of Complex objects.
copy constructor	Takes a single ComplexMatrix argument which is copied into a new ComplexMatrix object. You must also use dynamic memory allocation here.
destructor	Deallocate the memory allocated in the constructor and print a confirmation message as shown in the sample below.
= (assignment operator)	Copies one ComplexMatrix object to another existing ComplexMatrix object. According to the 'rule of three', we should overload this operator if we are overloading the copy constructor and destructor. Be careful to avoid memory leaks, and to avoid self-assignment!
<<	Stream the ComplexMatrix to the output in the format shown in the sample below.
fillMatrix()	Prompt the user to input each Complex matrix element individually, as shown in the sample below.
+	Add two ComplexMatrix objects.
*	Multiply two ComplexMatrix objects.

()	Return a reference to a single matrix element of the ComplexMatrix object, with row and column specified by the two integer arguments.

You should create a folder in your Deepthought home directory called ‘Lab4’. The four class code files (Complex.h, Complex.cpp, ComplexMatrix.h, and ComplexMatrix.cpp) must be contained in that folder. You must use the appropriate turnin script for your section to submit the folder containing your code by the due date and time. (See Lab0 for detailed instructions.) The TA or grader will then compile and test the compiled program on Deepthought. Therefore, to ensure your code runs correctly for the TA or grader, it is strongly recommended that you test your code on Deepthought before submission. You may choose to do all of the coding on Deepthought at your discretion.

Sample output from completed code (user input indicated in red)

```
Initial variable values:

c1 = 1 + i5, c2 = 0 + i0


Please enter complex number c3, in the format X + iY, or X - iY:
0.5 - i2.3

You entered c3 = 0.5 - i2.3

The absolute value of c3 = 2.35372

c3 is postdecremented in this statement.  c3 = 0.5 - i2.3

c3 is predecremented in this statement.  c3 = -1.5 - i2.3


Please enter a double precision number d1: 5.1

d1 + c3 = 3.6 - i2.3

c3 + d1 = 3.6 - i2.3


Please enter another complex number c4, in the format X + iY, or
X - iY: -7.7 + i3.0
```

```
c3 + c4 = -9.2 + i0.7

c3 * c4 = 18.45 + i13.21


Please enter an integer size for the complex matrix M1: 3

You will now be prompted to enter the complex elements of M1.

Please enter the matrix elements in the order indicated.

Element (0, 0): 1 + i1

Element (0, 1): 2 + i2

Element (0, 2): 3.3 + i4.4

Element (1, 0): 0 + i0

Element (1, 1): -10 + i0

Element (1, 2): 0 - i6

Element (2, 0): -1 - i2

Element (2, 1): 1 + i2

Element (2, 2): 5 - i0.5

You entered M1 =

         1 + i1         2 + i2       3.3 + i4.4

         0 + i0       -10 + i0         0 - i6

        -1 - i2         1 + i2         5 - i0.5


After copying from M1, M2 =

         1 + i1         2 + i2       3.3 + i4.4

         0 + i0       -10 + i0         0 - i6

        -1 - i2         1 + i2         5 - i0.5


We will now modify a single element of M2.
```

```
Please enter the row of the element of M2 to modify, remembering
that we are counting from 0 in C++: 2

Please enter the column of the element of M2 to modify: 1

Please enter a new complex value for the element: 0.3 + i0.4

After modifying the element, M2 =

         1 + i1          2 + i2       3.3 + i4.4

         0 + i0        -10 + i0         0 - i6

        -1 - i2        0.3 + i0.4         5 - i0.5



M3 = M1 + M2 =

         2 + i2          4 + i4       6.6 + i8.8

         0 + i0        -20 + i0         0 - i12

        -2 - i4        1.3 + i2.4        10 - i1

M4 = M2 * M3 =

        11 - i18    -46.27 - i18.36      59.2 + i32.1

       -24 + i12     214.4 - i7.8          -6 + i60

       -10 - i25       5.7 - i8.65       65.3 - i35.6

The memory allocated for the matrix has been deleted.

The memory allocated for the matrix has been deleted.

The memory allocated for the matrix has been deleted.

The memory allocated for the matrix has been deleted.
```

Lab Grading Rubric

If a student's program runs correctly and produces the desired output, the student has the potential to get a 100 on his or her lab; however, TA's will **randomly** look through this set of "perfect-output" programs to look for other elements of meeting the lab requirements. The table below shows typical deductions that could occur.

In addition, if a student's code does not compile, then he or she will have an automatic 30% deduction on the lab. Code that compiles, but does not match the sample output can incur a deduction from 10% to 30% depending on how poorly the output matches the output specified by the lab. This is in addition to the other deductions listed below or due to the student not attempting the entire assignment.

AUTOMATIC GRADING POINT DEDUCTIONS

Element	**Percentage Deduction**	**Details**
Does Not Compile	30%	Program does not compile on deepthought cluster!
Does Not Match Output	10%-30%	The program compiles but doesn't match all output exactly

ADDITIONAL GRADING POINT DEDUCTIONS FOR RANDOMLY SELECTED PROGRAMS

Element	**Percentage Deduction**	**Details**
Correct file structure	10%	Does not use both .cc and .h files, implementing class prototype correctly
Encapsulation	10%	Does not use correct encapsulation in object-oriented objects
Constructors	10%	Does not implement constructors with the correct functionality.
Member or Global Function Specifications	10% for each missing	Does not implement each member function or global function specified in the lab assignment
Clear Self-Documenting Coding Styles	5%-15%	This can include incorrect indentation, using unclear variable names, unclear comments, or compiling with warnings. (See next page)

LATE POLICY

Element	Percentage Deduction	Details
First Day	15%	This is within 24 hours of program due date
Each Additional Day	20% per day	The weekend (Sat/Sun) will count as one day