EECE 1080C / Programming for ECE

Summer 2019

Laboratory 10: C++ Operator Overloading

Plagiarism will not be tolerated:

all students who share files will receive a 100% penalty on this assignment

Topics covered:

- Classes
- Functions

Objective:

- To practice the use of overloaded operators when manipulating objects relative to a specific class.
- The main goals of this laboratory are as follows:
  - Gain hands-on experience with the creation and usage overloaded operators in C++.
  - Learn aspects of the programming language that are nearly exclusively associated with objects. These include:
    - Private and public settings
    - Modifying members
    - Information exchange in functions

Collaboration:

- As with most laboratory assignments in this course this laboratory assignment is to be performed by an individual student. You can help each other learn by reviewing assignment materials, describing to each other how you are approaching the problem, and helping each other with syntax errors. You can also get help from teaching assistants and instructors.
- Having slightly similar code for some assignments is expected but most assignments have multiple different solutions. Your code is expected to be different.
- Please document any help you receive from teaching assistants or instructors. Just add the names to the top of your source file.
- Having someone else code for you, sharing code with other students, or copy-pasting code from the internet or previous terms, **is cheating**. A helper should "teach you to fish, not feed you the fish". Laboratory assignments prepare you for exams so be smart.

Highlights:

- Please consider the following:
  - Add a main comment section at the beginning of each program you submit, specifying your name in Pinyin, the date it was last updated and a brief description of what it does.
  - Read problem statements carefully in order to fulfill all instructed requirements.
  - Remember to validate inputs accordingly. It is considered a good programming practice.
  - Run your program for all probable outcomes to confirm its functionality. If possible, have other people test your code. This can help identify and troubleshoot problems.
  - Comment important sections of your code to easily recognize your logic and approach to solving the tasks.
  - Be mindful of all details such as appropriate result display, concise prompts, variable handling, return instructions, etc.
  - Before uploading, be sure to test your code for the last time to verify that it encounters no errors and check that is the latest version of your program.
- To receive full credit for this laboratory please sign the attendance sheet.
- Please access the laboratory assignment via the canopy/blackboard link. The descriptions for each problem are contained within this document.
- Each part should be worked on separately. You will need a separate project for each part of this assignment when working within your IDE.
- **Submit your .cpp file on Blackboard using the assignment dropbox.**

Rubric: 100 points

- Part A = 35
- Part B = 65 or Part C = 65

Instructions: everyone should complet Part A. Then, you pick... which activity you would like to complete

Part B: is related algebra and geometry

Part C: is related to complex numbers

Choose to complete Part B or Part C

Tasks:

A. Time Conversion - Extended

1. Modify your Time Conversion program, developed in Task A of Lab 6, to get two dates and times from a text file created by the user following standard notation.

   YYYY/MM/DD – hh:mm:ss

   Assume all inputted dates and times are appropriate. There is no need to validate this condition. Overload the `operator>>` accordingly and extend your class to store days, months and years. Program the necessary constructors and set/get functions.
2. Overload the `operator<`, `operator>` and `operator==` to compare between the dates and times entered by the user. All these operations should return Boolean values.
3. Overload the `operator-` to determine the time between the dates entered by the user. This operation should return the number of seconds between both dates. Bear in mind that if you subtract a larger date from a smaller one, your answer might have no meaning. You do not need to validate this condition.
4. Implement another function to take the total seconds between the dates (from **3**) and output the time difference in number of years, months, days, hours, minutes and seconds. Overload the `operator<<` accordingly to quickly format and print your results. **See the examples below.**

- **Hint:** Assume that the average days per year is 365.25. And the average days per month is 30.4375. This should simplify all computations and estimate the solution accurately.
- **Hint:** You will have to pick a common unit when combining all the results. Consider using months or days as the common unit, to prevent the numbers from becoming "too big" or "too small"

Hint: Overload the (cin/cout) functions using the following code inside your "Point" class. This allows the user to quickly enter and print the pair of coordinates using a specific format, as shown.

```
// cout implementation for Point class
friend ostream& operator<<(ostream& output, const Point& P)
{
	output << "( " << P.x << " , " << P.y << " )";
	return output;
}

// cin implementation for Point class
friend istream& operator>>(istream& input, Point& P)
{
	input >> P.x >> P.y;
	return input;
}
```

This lets the programmer use the "cin/cout" commands as follows:

```
cin >> P1;
cout << "You have entered for P1: " << P1 << endl;
```

Bonus:

- See if you can convert these into input/output **files**
- `ostream` **becomes** `ofstream`
- `istream` **becomes** `ifstream`

- **Example 1:**

```
You have entered T1: [ 2019 / 6 / 27 - 19 : 37 : 25 ] and T2: [ 1990 / 4 / 25 - 7 : 32 : 49 ]
```

Comparison.

Date T1 is greater than Date T2.

Date subtraction.

[2019 / 6 / 27 - 19 : 37 : 25] - [1990 / 4 / 25 - 7 : 32 : 49]

Answer: Years: 29 - Months: 2 - Days: 2 - Hours: 12 - Minutes: 4 - Seconds: 36

- **Example 2:**

You have entered T1: [1990 / 4 / 25 - 7 : 32 : 49] and T2: [2019 / 6 / 27 - 19 : 37 : 25]

Comparison.

Date T2 is greater than Date T1.

Date subtraction.

[1990 / 4 / 25 - 7 : 32 : 49] - [2019 / 6 / 27 - 19 : 37 : 25]

Answer: Years: -30 - Months: 9 - Days: 27 - Hours: 22 - Minutes: 25 - Seconds: 24

This example illustrates the idea of negative time for the same two values in EXAMPLE 1. The answer is not the same and it has no meaning. The goal is to show the comparison between the two dates.

- **Example 3:**

You have entered T1: [2019 / 6 / 27 - 12 : 30 : 30] and T2: [2000 / 1 / 1 - 0 : 0 : 0]

Comparison.

Date T1 is greater than Date T2.

Date subtraction.

[2019 / 6 / 27 - 12 : 30 : 30] - [2000 / 1 / 1 - 0 : 0 : 0]

Answer: Years: 19 - Months: 5 - Days: 26 - Hours: 12 - Minutes: 30 - Seconds: 30

B. Point – Line - Triangle

1. Create a "Point" class which allows the programmer to store an *(x, y)* coordinate pair. It should have at least two constructors (default and user), at least one set function, and get functions for *x* and *y*. This means your class variables should be <u>private</u>.
   - it is okay to use one composite user/default constructor

2. Build a "Line" class that allows the programmer to store two coordinate pairs which will define a line in a rectangular coordinate system. This class must use the "Point" class developed in Part 1. Any "Line" object declared should have access to two constructors and appropriate set/get functions. The class should also include functions that return the proper value for the following:
   - **The slope of the line.**

     For any two given points (x_1, y_1) and (x_2, y_2), the slope of the line they define is calculated as:

     $$m = \frac{y_2 - y_1}{x_2 - x_1}$$

   - **The equation of the line.**

     For any points (x_1, y_1), the equation of the line is obtained as:

     $$y = mx + b$$

     Where *m* is the slope of the line and *b* is the *y* intercept of the line:

     $$b = y_1 - mx_1$$

   - **The length of the segment between the two points.**

     For any two given points (x_1, y_1) and (x_2, y_2), the length of the segment they define is:

     $$l = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

   - **Is the line horizontal?**

     A line is horizontal when:

     $$m = 0$$

   - **Is the line vertical?**

     A line is vertical when:

     $$m \to \infty \ so \ x_1 = x_2$$

3. Determine if the first line is parallel to another line defined <u>by two more points entered by the user.</u>
   - **Two lines are parallel when their slopes are equal, so:**

     $$m_1 = m_2$$

4. **Overload the "= =" operator (public) to compare two Point objects** and return either true or false. The operation should return true if both Points are equal and false otherwise. Validate if the points entered by the user are the same. The minimum requirement to define a line is two DIFFERENT points. If the points are equal, output an error message and prompt the user again.

- **<u>Example 1:</u>**

Please enter x and y for Point 1:

0

3

Please enter x and y for Point 2:

4

0

You have entered for P1: (0 , 3) and for P2: (4 , 0)

The slope of Line 1 for P1 and P2 is: -0.75
The y-intercept for Line 1 for P1 and P2 is: 3
Therefore, the equation of Line 1 is y = -0.75*x + 3
The distance between P1 and P2 is: 5
Line 1 is oblique.

Please enter x and y for Point 3:

2

0

Please enter x and y for Point 4:

2

1

You have entered for P3: (2 , 0) and for P4: (2 , 1)

The equation of Line 2 is x = 2
Line 1 and Line 2 are not parallel.

- **Example 2:**

Please enter x and y for Point 1:

1

2

Please enter x and y for Point 2:

3

2

You have entered for P1: (1 , 2) and for P2: (3 , 2)

The slope of Line 1 for P1 and P2 is: 0
The y-intercept for Line 1 for P1 and P2 is: 2
Therefore, the equation of Line 1 is y = 2
The distance between P1 and P2 is: 2
Line 1 is horizontal.

Please enter x and y for Point 3:

```
                                                                    -1
                                                                    -4
Please enter x and y for Point 4:
                                                                    -2
                                                                    -4

                                    You have entered for P3: ( -1 , -4 ) and for P4: ( -2 , -4 )

                                                    The equation of Line 2 is y = -4
                                                     Line 1 and Line 2 are parallel.
```

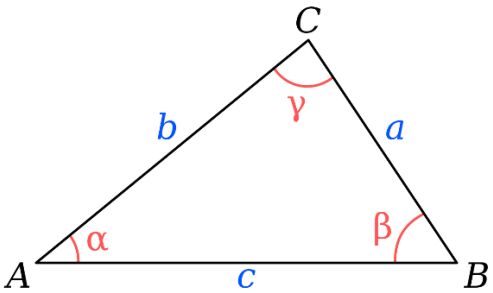
5. Design a C++ class to implement a "Triangle" which allows the program to store three coordinate pairs that will represent a triangle object in a rectangular coordinate system. The object should use the "Point" class developed previously. If needed, you may use the "Line" Class. **Remember to handle the validation of all inputs.** The object should have at least two constructors and appropriate set/get functions. It should also include functions that return the proper value for the following:

- **Is the shape a triangle? This function should return a Boolean value.**
  For any three given points (x_1, y_1), (x_2, y_2) and (x_3, y_3), a **collinearity test** can determine if a triangle defined by said points exists. Three points cannot form a triangle if they all lie on the same line. In other words, if the slopes of the lines formed by each pair of points are equal, the points are collinear. Hence, a triangle cannot be constructed.

  Let Line 1 be formed by P1 and P2 and Line 2 by P2 and P3. The coordinates P1, P2 and P3 are collinear if and only if:

  $$m_{P_1P_2} = m_{P_2P_3}$$

  If P1, P2 and P3 are collinear, they CANNOT form a triangle.
- **The perimeter of the triangle.**
  Consider the following triangle:



  The perimeter is:

  $$P = a + b + c$$

- **The area of the triangle.**
  Consider the triangle above. Using Heron's formula, the area of any triangle is:

$$A = \sqrt{s(s-a)(s-b)(s-c)}$$

Where *s* is the semi-perimeter of the triangle, defined as:

$$s = \frac{P}{2}$$

6. **Overload the "= =" operator (public) to compare two triangle objects** and return either true or false. The operation should return true if both triangles are congruent and false otherwise. Remember that to compare them, both triangles have to be valid. In geometry, two figures are congruent if they have the same shape and same size, or if one has the same shape and size as the mirror image of the other.

See the examples below.

- **Example 3:**

```
Please enter x and y for Point 5:
                                                                    0
                                                                    0
Please enter x and y for Point 6:
                                                                    0
                                                                   -3
Please enter x and y for Point 7:
                                                                   -4
                                                                    0

            You have entered for P5: ( 0 , 0 ), for P6: ( 0 , -3 ) and for P7: ( -4 , 0 )

                        The points are not collinear. Therefore, they can form a triangle.

                                                  The perimeter of Triangle 1 is: 12
                                                        The area of Triangle 1 is: 6

Please enter x and y for Point 8:
                                                                    0
                                                                    4
Please enter x and y for Point 9:
                                                                    0
                                                                    0
Please enter x and y for Point 10:
                                                                    3
                                                                    0

            You have entered for P8: ( 0 , 4 ), for P9: ( 0 , 0 ) and for P10: ( 3 , 0 )

                        The points are not collinear. Therefore, they can form a triangle.
```

Triangle 1 and Triangle 2 are congruent.

- **Example 4:**

Please enter x and y for Point 5:

1

1

Please enter x and y for Point 6:

2

2

Please enter x and y for Point 7:

3

3

You have entered for P5: (1 , 1), for P6: (2 , 2) and for P7: (3 , 3)

The points are collinear. Hence, they cannot construct a triangle.

Press any key to close this window . . .

Please enter x and y for Point 5:

-0.5

0

Please enter x and y for Point 6:

0

2

Please enter x and y for Point 7:

0.5

0

You have entered for P5: (-0.5 , 0), for P6: (0 , 2) and for P7: (0.5 , 0)

The points are not collinear. Therefore, they can form a triangle.

The perimeter of Triangle 1 is: 5.12311

The area of Triangle 1 is: 1

Please enter x and y for Point 8:

0

0

Please enter x and y for Point 9:

1

2

Please enter x and y for Point 10:

1

0-2

You have entered for P8: (0 , 0), for P9: (1 , 2) and for P10: (1 , -2)

The points are not collinear. Therefore, they can form a triangle.

Triangle 1 and Triangle 2 are not congruent.

C. Complex Numbers

1. Create a "Complex" class which allows the programmer to store complex numbers entered by the user in their rectangular form, noted as follows:

$$z = a + bi$$

Where *a* and *b* are two real numbers that are known as real and imaginary part, respectively; and *i* is the imaginary unit defined as:

$$i = \sqrt{-1}$$

The "Complex" class should have at least two constructors and get functions for the real *(a)* and imaginary *(b)* part. For this particular problem, assume the user will not enter **zero** for any complex number. This only means:

$$a = 0\,;b = 0$$

This validation is not required. Any other case is valid and should be implemented. **See the examples below.**

2. Overload the << and >> (cin/cout) operators to quickly read and print the complex numbers entered by the user. Make sure to format your complex numbers appropriately. Use the hint provided on the previous task.
3. Overload the **+, -, *** and **/** to perform the four basic math operations between two complex numbers entered by the user. All operations should return a third complex number. Derive and program the expressions accordingly.
   - **Hint:** Division between two complex numbers can be tricky in their rectangular form. Consider getting rid of *i* in the denominator using algebra.

$$\frac{z_1}{z_2} = \frac{a + bi}{c + di}$$

4. Create a function that computes the conjugate of both complex numbers entered by the user. The conjugate of a complex number is defined as follows:

$$z = a + bi \rightarrow z^* = a - bi$$

5. Create a "ComplexPolar" class which allows the programmer to store complex numbers in their polar form. Set a constructor to compute this conversion by taking in a type "Complex" object and implement get functions if needed. Using the **cis** function, complex numbers are noted as follows:

$$z = a + bi = |z|(\cos\varphi + i\sin\varphi) = |z|cis(\varphi)$$

Where $|z|$ is the modulus and φ the argument of the complex number. They are defined as:

$$|z| = \sqrt{a^2 + b^2}$$

$$\varphi = \begin{cases} \frac{\pi}{2} - \arctan\left(\frac{a}{b}\right), b \geq 0 \\ \frac{3\pi}{2} - \arctan\left(\frac{a}{b}\right), b < 0 \end{cases}$$

For this particular problem, the argument of all complex numbers should be in radians.

6. Overload the << (cout) operator to output complex numbers in their polar form. Make sure to format your complex numbers appropriately. **See the examples below.**
7. Overload the ^ operator to perform the exponentiation of complex numbers in their polar form. This operation returns another complex number in the polar form. The exponentiation of a complex number is defined as:

$$z^n = |z|^n cis(n\varphi)$$

Where n is a **real number**. For this particular problem, set $n = 7$.

8. Program one of the constructors of the "Complex" class to convert complex numbers from polar form to rectangular form, by using the following equations:

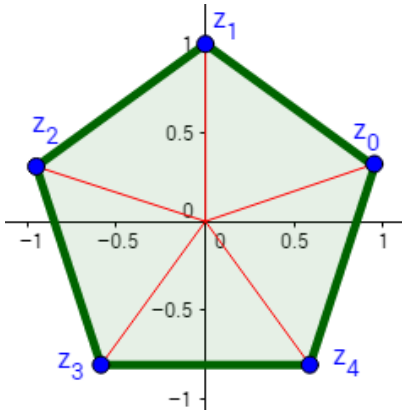
$$a = |z| \cos \varphi$$

$$b = |z| \sin \varphi$$

9. Overload the % operator to obtain the n-root of any complex number in its polar form. This operation should return a vector of complex numbers in the polar form. The n-root of a complex number is defined as follows:

$$\sqrt[n]{z} = |z|^{1/n} cis\left(\frac{\varphi + 2\pi k}{n}\right), 0 \leq k \leq n-1$$

Where n is a positive integer. For this particular problem, set $n = 5$. This means that a complex number has n-roots due to its properties. Moreover, representing each of those roots on the complex plane generates a regular polygon of n-sides and n-vertices centered on the origin of the plane. Each element of your vector should contain one root of said complex number. As an example, these are the solutions to $\sqrt[5]{i}$.



10. Add four non-member `operator+` `friend` functions that complete the following summation

ComplexPolar = Complex + ComplexPolar

ComplexPolar = ComplexPolar + Complex

Complex = Complex+ComplexPolar

Complex = ComplexPolar + Complex

11. Test your program by entering two complex numbers in their rectangular form. **See the examples below.**
    - Obtain the conjugate of each complex number.

- Perform all 4 basic operations between them. Output all results.
- Convert both complex numbers to their polar form.
- Compute z_1^7 and $\sqrt[5]{z_2}$ and convert these results into rectangular form.
- Print all results. Make sure to format all complex numbers appropriately.

- **Example 1:**

```
Please enter the real and imaginary part for Complex 1:
-1
1
Please enter the real and imaginary part for Complex 2:
0
1

You have entered for C1: [ -1 + 1i ] and for C2: [ 1i ]

C1 conjugate: [ -1 - 1i ]
C2 conjugate: [ -1i ]

[ -1 + 1i ] + [ 1i ] = -1 + 2i
[ -1 + 1i ] - [ 1i ] = -1
[ -1 + 1i ] * [ 1i ] = -1 - 1i
[ -1 + 1i ] / [ 1i ] = 1 + 1i

Polar form for C1: 1.41421 cis ( 2.35619 )
Polar form for C2: 1 cis ( 1.5708 )

Calculations for C1: Exponentiation
[ -1 + 1i ] ^ 7 = 11.3137 cis ( 16.4934 )
11.3137 cis ( 16.4934 ) in Rectangular form is: -8 - 8i

Calculations for C2: Nth Roots
Root 1: [ 1i ] ^ ( 1 / 5 ) = 1 cis ( 0.314159 )
Root 2: [ 1i ] ^ ( 1 / 5 ) = 1 cis ( 1.5708 )
Root 3: [ 1i ] ^ ( 1 / 5 ) = 1 cis ( 2.82743 )
Root 4: [ 1i ] ^ ( 1 / 5 ) = 1 cis ( 4.08407 )
Root 5: [ 1i ] ^ ( 1 / 5 ) = 1 cis ( 5.34071 )

Root 1: 1 cis ( 0.314159 ) in Rectangular form is: 0.951057 + 0.309017i
Root 2: 1 cis ( 1.5708 ) in Rectangular form is: 1i
Root 3: 1 cis ( 2.82743 ) in Rectangular form is: -0.951057 + 0.309017i
Root 4: 1 cis ( 4.08407 ) in Rectangular form is: -0.587785 - 0.809017i
Root 5: 1 cis ( 5.34071 ) in Rectangular form is: 0.587785 - 0.809017i
```

- **Example 2:**

Please enter the real and imaginary part for Complex 1:

0

-2

Please enter the real and imaginary part for Complex 2:

-4

0

You have entered for C1: [-2i] and for C2: [-4]

C1 conjugate: [2i]

C2 conjugate: [-4]

[-2i] + [-4] = -4 - 2i

[-2i] - [-4] = 4 - 2i

[-2i] * [-4] = 8i

[-2i] / [-4] = 0.5i

Polar form for C1: 2 cis (4.71239)

Polar form for C2: 4 cis (3.14159)

Calculations for C1: Exponentiation

[-2i] ^ 7 = 128 cis (32.9867)

128 cis (32.9867) in Rectangular form is: 128i

Calculations for C2: Nth Roots

Root 1: [-4] ^ (1 / 5) = 1.31951 cis (0.628319)

Root 2: [-4] ^ (1 / 5) = 1.31951 cis (1.88496)

Root 3: [-4] ^ (1 / 5) = 1.31951 cis (3.14159)

Root 4: [-4] ^ (1 / 5) = 1.31951 cis (4.39823)

Root 5: [-4] ^ (1 / 5) = 1.31951 cis (5.65487)

Root 1: 1.31951 cis (0.628319) in Rectangular form is: 1.0675 + 0.775587i

Root 2: 1.31951 cis (1.88496) in Rectangular form is: -0.40775 + 1.25493i

Root 3: 1.31951 cis (3.14159) in Rectangular form is: -1.31951

Root 4: 1.31951 cis (4.39823) in Rectangular form is: -0.40775 - 1.25493i

Root 5: 1.31951 cis (5.65487) in Rectangular form is: 1.0675 - 0.775587i

- **Example 3:**

Please enter the real and imaginary part for Complex 1:

4

Please enter the real and imaginary part for Complex 2:

4

-3

You have entered for C1: [3 + 4i] and for C2: [4 - 3i]

C1 conjugate: [3 - 4i]

C2 conjugate: [4 + 3i]

[3 + 4i] + [4 - 3i] = 7 + 1i

[3 + 4i] - [4 - 3i] = -1 + 7i

[3 + 4i] * [4 - 3i] = 24 + 7i

[3 + 4i] / [4 - 3i] = 1i

Polar form for C1: 5 cis (0.927295)

Polar form for C2: 5 cis (5.63968)

Calculations for C1: Exponentiation

[3 + 4i] ^ 7 = 78125 cis (6.49107)

78125 cis (6.49107) in Rectangular form is: 76443 + 16124i

Calculations for C2: Nth Roots

Root 1: [4 - 3i] ^ (1 / 5) = 1.37973 cis (1.12794)

Root 2: [4 - 3i] ^ (1 / 5) = 1.37973 cis (2.38457)

Root 3: [4 - 3i] ^ (1 / 5) = 1.37973 cis (3.64121)

Root 4: [4 - 3i] ^ (1 / 5) = 1.37973 cis (4.89785)

Root 5: [4 - 3i] ^ (1 / 5) = 1.37973 cis (6.15449)

Root 1: 1.37973 cis (1.12794) in Rectangular form is: 0.591248 + 1.24663i

Root 2: 1.37973 cis (2.38457) in Rectangular form is: -1.00291 + 0.94754i

Root 3: 1.37973 cis (3.64121) in Rectangular form is: -1.21108 - 0.661015i

Root 4: 1.37973 cis (4.89785) in Rectangular form is: 0.254419 - 1.35607i

Root 5: 1.37973 cis (6.15449) in Rectangular form is: 1.36832 - 0.177082i