EECE 1080C/CS1021C - Laboratory 9 - Summer 2019

Shapes and File I/O

Topics covered: Classes & File I/O

Due:

Sunday July 21, 2019 by 11:59PM uploaded on Blackboard

Objective:

This assignment is an introduction to the C++ classes and optionally File I/O for bonus points. We will learn the basics of class design, and we will put it to use creating a couple of more advanced classes.

Collaboration:

As with most laboratory assignments in this course this laboratory assignment is to 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 get help from tutors, teaching assistants, and instructors. Having 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 other students, 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 review the chapter on classes (including operator overload).
- Optionally review the chapter on File I/O.
- Review the **grading Rubric** at the end of specification.
- **Lab Attendance Milestone: Completion of Part A (all classes)**
- Please ask appropriate questions when necessary.
- This assignment may require completed Point class and/or Line class in Lab8. Wherever required, you can copy in the class definitions of Point and/or Line class or “include”. **Also, make sure to edit these class definitions to have double instead of int.**
- Examine the breakup_string.cpp helper file provided with the assignment.
- Take a few minutes to read this entire document, at least once, before beginning to program it.

EECE 1080C/CS1021C - Laboratory 9 - Summer 2019

Shapes and File I/O

Topics covered: Classes & File I/O

Specification:

Part A: More Classes

Task 1: Circle Class:

Submit as File: circle.cpp

Notes:

You will want to add an overloaded == operator to the **public section** of the Point class using the following:

```
bool operator==(Point &rhs) {
      return (rhs.x == x && rhs.y == y);
}
```

Once you have completed that you should be able to do the following with any Point related objects.

```
if(P1 == P2) cout << "Points are equal" << endl;
else cout << "Points are not equal" << endl;
```

You may want to copy the line.cpp file from lab8 to circle.cpp. The circle class is quite similar to the line class. It uses two point objects just like the line class.

Create an object to implement a "circle" class which allows the programming to store a circle object. The object should use the "point" class developed previously. You will be given the center point and one point on a circle. The object should have at least two constructors, appropriate set/get functions, and overloaded I/O functions. It should also include functions that return the proper value for the following:

1. Determine if the object is a circle.
   Point test: If both points are the same, a circle cannot be formed.
   This should return a bool.

2. Calculates the radius using the line distance formula
3. Calculates the diameter.
4. Calculates the area of a circle.
5. Calculates the circumference of a circle.
6. Overload >> (cin) operator that lets the user provide inputs to construct a Circle. Refer Laboratory 8 for this purpose.
7. Overload << (cout) operator that lets the user display the Circle and its properties (1-5 specifications above).Refer Laboratory 8 for this purpose.
8. Overload == operator (**public**) that compares two Circle objects and returns either **true** or **false.** IT returns true if both circles are the same (same

center and radius) and false otherwise. To compare them, both circles have to be valid (Use Step 1). **If using "point" class to build circles, assume P1 is the center.**

Task 2: Triangle class:

Submit as File: triangle.cpp

Note:

You will want to copy the Circle class implementation to triangle.cpp, rename the class to Triangle, and add another Point object to store the third point of a triangle. After that is complete add a third point to both cin/cout overloaded operators. Remove or modify any remaining circle related functions/methods.

Design a C++ class to implement a "triangle" - which allows the program to store a triangle object. The object should use the "point" class developed previously. The object should have at least two constructors, appropriate set/get functions. It should also include functions that return the proper value for the following:

1. Determines if your object is a triangle. Use the **collinearity test** described at the end of this document.

   This should return a bool

2. Calculates the area of a triangle.
3. Calculates the perimeter of the triangle.
4. Determines if the triangle is an equilateral triangle.
5. Overload >> (cin) operator that lets the user provide inputs to construct a Triangle. Refer Laboratory 8 for this purpose.
6. Overload << (cout) operator that lets the user display the Triangle and its properties (1-4 specifications above). Refer Laboratory 8 for this purpose.
7. Overload == operator (**public**) that compares two triangle objects and returns either **true** or **false.** IT returns true if both triangles are the same (same points) and false otherwise. To compare them, both triangles have to be valid (Use Step 1).

EECE 1080C/CS1021C - Laboratory 9 - Summer 2019

Shapes and File I/O

Topics covered: Classes & File I/O

Task 3: Quadrilateral Class:

Submit as File: quad.cpp

Create an object to implement a Quadrilateral class which allows the programming to store a quadrilateral object. The object can use either the “point” or “line” class developed previously. The object should have at least two constructors, appropriate set/get functions. It should also include functions that return the proper value for the following:

1. Determine if your object can be a quadrilateral. 4 points can form a quadrilateral, when no three points lie on the same line. Use the **collinearity test** described at the end of this document.

   This should return a bool.

2. Calculate the area of the Quadrilateral. A shoelace formula to compute this is posted below.
3. Overload >> (cin) operator that lets the user provide inputs to construct a Quadrilateral. Refer Laboratory 8 for this purpose.
4. Overload << (cout) operator that lets the user display the Quadrilateral and its properties (1-3 above specifications). Refer Laboratory 8 for this purpose.

Submit **separate** source files for all the class implementations above. You can test your classes and functionality in main() whenever required.

EECE 1080C/CS1021C - Laboratory 9 - Summer 2019

Shapes and File I/O

Topics covered: Classes & File I/O

Part B (Optional) File I/O:

Submit as File: shape-info.cpp

You are also provided text file - shapes.txt which contains coordinate(s) information about the shapes. For example, the shape information in the text file can look like below (find the meaning in comments - only in this document):

```
0 0 0 1 1 0   //Three (x,y) pairs - x1 y1 x2 y2 x3 y3

0 0 2.5 0 2.5 2.5 0 2.5 // Four (x,y) pairs - x1 y1 x2 y2 x3 y3 x4 y4

0 0 2 2 //Two (x,y) pairs - x1 y1 x2 y2
```

……..and many more

Assume that, 3 (x,y) pairs (or 6 numbers) **MAY** form a Triangle, 4 pairs **MAY** form a Quadrilateral and 2 pairs **MAY** form a circle (one being the center and the other will be a point on the circumference). Any line with different number of coordinates is invalid input.

In this task you will be reading text from shapes.txt file and writing into another file shapes-info.txt.

Steps to implement file I/O:

1. Create an **ifstream** object and **open** the input text file **shapes.txt**. Also create an **ofstream** object and open the text file **shapes-info.txt**.
2. Read a single line from input file using input stream object into a temporary string. Taking hint from breakup_string.cpp (requires sstream header file), break up the temporary string into a numbers (coordinates).
3. Determine if number of coordinates are sufficient to form any of the shapes. If not, print out the invalidity in the output file and repeat step 2 (one example is shown below).
4. If from step 3 the number of coordinates are valid, **construct** the specific shape.
5. If the shape can indeed be created (collinearity test for Triangle or Quadrilateral and point test for circle), using ofstream object, print all the calculations into the output text file (output examples are shown below).
6. Repeat steps 2-5 until end-of-file (eof()) is encountered in the input stream.

Submit this File I/O task as a separate file.

EECE 1080C/CS1021C - Laboratory 9 - Summer 2019

Shapes and File I/O

Topics covered: Classes & File I/O

Note:

Most editors may not find the shapes.txt file even if it is in the same folder as the source code. That can be fixed with:

infile.open("c:\\very\\specific\\spot\\shapes.txt")

Considering infile is your ifstream object.
You can copy the path of the file and put it with in " ".

EECE 1080C/CS1021C - Laboratory 9 - Summer 2019

Shapes and File I/O

Topics covered: Classes & File I/O

Input & Output eg:

Input in shapes.txt:

```
0 0 1 0 0 1
0 0 1 0 1 1 0 1
0 0 1 1
1 1 2 2 3
0 0 2.5 0 2.5 2.5 0 2.5
1 1 1 1
```

Corresponding Output in shapes-info.txt:

```
Sufficient coordinates input.
The object is a Triangle.
Area of the triangle: 0.5 sq. units
Perimeter of the triangle 3.414 units
The triangle is not an equilateral triangle


Sufficient coordinates input.
The object is a Quadrilateral.
Area of the Quadrilateral: 1 sq. units


Sufficient coordinates input.
The object is a Circle.
Radius of the Circle: 1 unit
Diameter of the Circle: 2.82 units
Area of the Circle: 6.28 sq units
Circumference of the Circle: 8.88 units

Sufficient coordinates NOT input.

Sufficient coordinates input.
The object is a Quadrilateral.
Area of the Quadrilateral: 6.25 sq. units

Sufficient coordinates input.
The object is not a Circle.
```

EECE 1080C/CS1021C - Laboratory 9 - Summer 2019

Shapes and File I/O

Topics covered: Classes & File I/O

Grading Rubric:

- Part A - Triangle Class: **30 points**
- Part A - Quadrilateral Class: **30 points**
- Part A - Circle Class: **30 points**
- Attendance/Check Point: **10 points**
- Part B - File I/O Bonus: **20 points**

EECE 1080C/CS1021C - Laboratory 9 - Summer 2019

Shapes and File I/O

Topics covered: Classes & File I/O

Notes:

Collinearity test - Triangle:

Assuming you are constructing a triangle object with 3 points - P1(x1,y1), P2(x2,y2) and P3(x3,y3), a collinearity test can determine if the triangle is valid.

Three points cannot form a triangle if they all lie on the same line. Which means, if the slopes of the lines formed by each pair of points are equal the points are collinear. Hence, a triangle **CANNOT** be constructed.

Slope of P1P2 = Slope of P2P3

=> (y2-y1)/(x2-x1) = (y3-y2)/(x3-x2)

=> x1(y2-y3)+x2(y3-y1)+x3(y1-y2) = 0

If the above equation is true, triangle cannot be formed

Collinearity test - Quadrilateral:

Assuming you are constructing a quadrilateral object with 4 points - P1(x1,y1), P2(x2,y2), P3(x3,y3) and P(x4,y4), a collinearity test can determine if the quadrilateral is valid.

Four points cannot form a quadrilateral if any three points among them lie on the same line (or are collinear). We can perform a collinearity test on combinations.

If triads (P1,P2,P3), (P2,P3,P4), (P1,P3,P4), (P1,P2,P4) have collinear points, equations obtained are:

x1(y2-y3)+x2(y3-y1)+x3(y1-y2) = 0

x2(y3-y4)+x3(y4-y2)+x4(y2-y3) = 0

x1(y3-y4)+x3(y4-y1)+x4(y3-y1) = 0

x1(y2-y4)+x2(y4-y1)+x4(y1-y2) = 0

If any of the equations are true, a quadrilateral cannot be formed.

EECE 1080C/CS1021C - Laboratory 9 - Summer 2019

Shapes and File I/O

Topics covered: Classes & File I/O

Heron's formula for area of a triangle:

Area = sqrt(p*(p-a)*(p-b)*(p-c)

Where: sqrt - square root
a,b,c are lengths of the sides of the triangle
p = perimeter of the triangle/2 = (a+b+c)/2

Area of a Quadrilateral:

Area of a Quadrilateral formed by 4 points P1(x1,y1), P2(x2,y2), P3(x3,y3), P4(x4,y4) can be calculated using the shoelace formula:

$$\mathbf{A}_{\text{quad.}} = \frac{1}{2}|x_1y_2 + x_2y_3 + x_3y_4 + x_4y_1 - x_2y_1 - x_3y_2 - x_4y_3 - x_1y_4|$$

Note:
This assumes that the points are ordered (clockwise or anticlockwise) on a 2-D plane (50% chance that points you input may not be ordered). Go ahead with the computation, irrespective.