Lab 7

Due No Due Date **Points** 10 **Available** after Feb 17 at 7pm

Polymorphism and Exceptions

In order to get credit for the lab, you need to be checked off by the end of lab. You can earn a maximum of 3 points for lab work completed outside of lab time, but you must finish before the next lab. For extenuating circumstances, contact your lab TAs and the instructor.

For this lab we will continue working with the shapes developed for Lab 6.

(4 pts) Implementing Polymorphism

Use the classes from Lab 6 (Shape, Rectangle, Circle, and Square) to create polymorphism. Answer the questions below and implement the polymorphism in the appropriate location of the interface and implementation files.

- Which function(s) are you going to make polymorphic?
- How will you make it polymorphic?
- Can it be a pure virtual function?
- Which class have you made an abstract base class?

(3 pts) Testing Polymorphism

Modify your **application.cpp** file to show polymorphism by making a function that prints the information of a shape. In this function, you will print the name, color, and area of the shape. You should pass your shape by reference (or address explicitly) to have polymorphism.

```
    void print_shape_info(Shape &);
```

(3 pts) Exceptions

After you've convinced yourself and the TAs that you have successfully implemented polymorphism for your shapes, it is time to add some error handling. Since constructors don't return anything, throwing an exception during object construction is one of the best ways to signal an error with the construction.

Throw an exception in your constructors if the user does not provide a valid argument for their member variables. The constructors should throw an invalid_argument exception, when the user tries to create an object that would result in an area of zero. You'll need to add the statement below to your function

```
throw invalid_argument("Invalid constructor argument");
```

First, run your program with trying to pass an invalid argument to your constructor to see what it does now that your function throws an exception, and you are not catching it.

Now, catch the exception so that it doesn’t have an error! Remember, you can use the what() member function to see your message from the invalid argument exception.