LAB 11

RESOURCES

- Discussion slides (./lab11slides.pdf) - Copy of the presentation given by the TA
- Nemo's Discussions (http://www.cise.ufl.edu/~nemo/cop3503/slides/) - Has a discussion on Polymorphism and Inheritance
- main.cpp (./main.cpp) - The main I will be using to test.
- out.txt (./out.txt) - Example output

IDEA

This lab should get you used to inheritance in C++. We will do this by implementing general shape class and extending it to implement a few specific shapes.

IN-LAB ASSIGNMENT

Description

You will first implement a Shape class that has an area function, a perimeter functio a function that prints out the shape. You will then extend this class to implement a rectangle, a triangle, and a square.

Requirements/Deliverables

You will first need to implement the base class: Shape. Every shape should have a typ This is just a string stating what type of shape it is. A Shape should have its type set "shape". A Rectangle should have its type set to "rectangle". A Tirangle should have i

type set to “triangle”. A Square should have its type set to “square”. Your Shapes mu
implement the following functions:

1.
```
virtual float area()
```

Returns the area of the shape.
- A general shape has zero area (we don’t know what shape it is so we can’t get it
area).
- A rectangle of height h and width w will have area wh
- A triangle with a base of b and height of h will have area bh/2
- A square of size s will have area s*s.

2.
```
virtual float perimeter()
```

Returns the perimeter of the shape.
- A general shape has zero perimeter
- A rectangle has perimeter 2w + 2h
- The triangle you will be implementing will be a right triangle with a base and a
height, then the final edge will have length sqrt(b^2 + h^2). So the whole triang
have a perimeter of b + h + sqrt(b^2 + h^2).
- The square has a perimeter of 4s.

3.
```
virtual string get_type()
```

Returns the string representation of the type as described earlier

4.
```
virtual void print()
```

Print out a nice string representation of your Shape. I will leave the exact
implementation of this up to you. This can be a text based representation such a

```
Square: side length=5.0
```

Or, if you are adventurous, you can print out a pricture of your Shape that scale
based on its parameters. For instance, a rectangle with side lengths 3 and 5 coul

```
+-----+
|     |
|     |
|     |
+-----+
```

Note that all of these functions are virtual functions. This means you are going to ne overload them for each subclass that has different behavior. Perhaps the exceptions the get type and number of sides functions. For instance, there are two ways to go ab the get type function. If you write your classes so that each Shape (and thus its subcl has a string member representing its type (protected members are your friend here), you should be able to write this function only once in your Shape class. Otherwise, yo need to make each subclass return something different (a different string).

The following constructors should also be implemented, though you may want to implement more in your code:

1.
   ```
   Shape()
   ```

   Default Shape constructor, not much to do here

2.
   ```
   Rectangle(float w, float h)
   ```

   Constructs a rectangle of width w and height h

3.
   ```
   Triangle(float b, float h)
   ```

   Constructs a Triangle with base b and height h

4.
   ```
   Square(float s)
   ```

   Constructs a Square with side lengths s

You may want to implement some other functions in your classes such as getters and setters for your shape’s paramters to make sure you code is working.

One more note is the Square class. A Square is a special type of a Rectangle. So, if yo already have a Rectangle class written, then the Square can be implemented very eas can be just a constructor, hint: you can call the Rectangle constructor from the Squa class).

You must write your code in such a way so that a *pointer* to a Shape will call the *mos derived* methods of the Shape it is pointing at.

Submission

Arrange the classes however you want in however many files you need. I will need a makefile with a target called all that produces an executable called shape from your s code files as well as a file that I will use to test your code called main.cpp. So, you sho probably make a main.cpp while you are testing your code so that your makefile will when I grade it on my machine.

Also, include a defs.h file that includes any and all .h files needed by your program to each of the shapes. The idea here is that in my main, I will just have #include "defs.h that should include all the Shape classes.

Hints

- Have fun with the print functions
- It will be much easier to focus on getting one shape working and then repeating you have done for each other shape rather trying to write every shape at once.

Grading Distribution

- 3 points for each function spread accross each class.
- 3 points for correct usage of virtual keywords and polymorphism properties
- 5 points for correct implmentation of inheritance.

Optional Enhancements