LAB #6
Debugging and Inheritance

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 the lab before the next lab. For extenuating circumstances, contact your lab TAs and Jennifer Parham-Mocello.

Reminder: All of our labs involve paired programming. You do not have to keep the same partner for each lab, but you MUST work with someone in each lab!!! First, find a partner for this lab. It can be the same partner from the previous lab or a different partner.

(2 pts) Conditional Compilation: One of the useful features of it is the ability to conditionally include code, based on macro definitions. For instance, this is often referred to as a DEBUG macro:

```c
#ifdef DEBUG
    /* your debug code here */
#endif
    /* your non-debug code here*/
```

For this task, ensure that all of your print statements are wrapped in DEBUG macros. Now, you can #define DEBUG to turn it on, and comment this statement out using // to turn it off, but you can also compile with a -D DEBUG to define it or leave it out.

```
g++ prog.cpp -D DEBUG
```

(4 pts) More Debugging and Errors. The roles are reversed and you are now the teacher!!! Think of an example for the following errors, create the buggy example yourself, and explain how you would find this logic or runtime error/bug in the future:

- Off by one error in a dynamic array
- Initialize a different object than the one you are using
- Forget “Big Three” with dynamic member variable
- Access memory that hasn’t been allocated

(4 pts) Inheritance - “Is A” relationship

Create classes for a Shape, Circle, and Rectangle. All shapes have a name and color and they take up some area. However, a circle’s area is dependent upon its radius, while a rectangle’s area is dependent on a length and width. You are to create these classes with the correct attributes in the correct class, with mutators and accessors for all the attributes.
class shape {
    private:
        string name;
        string color;
    public:

};

Your circle and rectangle classes will inherit attributes and behaviors from the shape class, but each class will have its own calculate_area() function and mutators and accessors for their local members not inherited from shape. In order to inherit from the shape class, your class declaration must look like:

class circle : public shape {
    private:
        double radius;
    public:

};

class rectangle : public shape {
    private:
        double length;
        double width;
    public:

};

Finish writing the shape, circle, and rectangle classes, and convince yourself that these classes work the way you intended. Would it have helped in any way to have the members in shape be protected rather than private?

Remember, you and your partner will not receive lab credit if you do not get checked off before leaving each lab. Once you have a zero on a lab, then it cannot be changed because we have no way of knowing if you were there or not!!!