Lab 7

Each lab will begin with a recap of last lab and a brief demonstration by the TAs for the core concepts examined in this lab. As such, this document will not serve to tell you everything the TAs will in the demo. It is highly encouraged that you ask questions and take notes. In order to get credit for the lab, you need to be checked off by the end of lab. For non-zero labs, 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 circumstance, contact your lab TAs and Instructor.

(4 pts) Operator Overload

Make an operator overload member for comparing properties based on their property value and number of rent paying tenants. For example, you want to have the ability to compare two properties using the > and < operators, such as if (p > p1). In order to do this, you need to make an operator overload for > and <.

bool operator>(const Property &, const Property &);
bool operator<(const Property &, const Property &);

Convince your TA that you can create different types of properties and the inheritance and operator overload is working.

(6 pts) Basic Polymorphism Practice

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 accessors/mutators for their local members not inherited from shape.

Use these classes to create polymorphism. Answer the questions below and implement the polymorphism.

- Which function 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?

Modify your driver.cpp file to show polymorphism by making a void print_shape_info() function. 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 &);
```