LAB #6 – Practice Inheritance/“Is A”

Each lab will begin with 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. 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.

Pair Programming
In this lab, you can choose a partner for pair programming. **You must be checked off together. You only need one computer for pair programming.** One person will be the driver, who controls the computer and codes, while the other is the navigator, who observes and advises. After 20 minutes, your TA will switch driver and navigator, continuing this pattern until the task is complete. Please read more about pair programming and the benefits: [Pair Programming Student Handout](#)

Classes Tip: If you find yourself struggling to understand classes and follow the flow of classes (when you are in which class), then the best way to think about classes is writing/designing each class on a separate piece of paper. Each time you put a dot after an object, then you essentially have taken yourself to a different piece of paper, and you can only access the members and functions on that piece of paper!!!

(8 pts) Inheritance and Static Members

Write a **Vehicle** class which is the base class and the inherited subclasses (**skateboard, motorcycle, race car, bike**). You may design your own member variables as appropriate, but all vehicles are either motorized or non-motorized, have a specific number of wheels, type/description, and speed.

Make the type/description a **static, constant string** because it does not change and is unique to the class, not the object, e.g. “bike”, “skateboard”, etc. I would not make the number of wheels or motorized/non-motorized members be static because these can change based on the object. For example, there are 3 wheel bikes and bikes with motors these days! Whether these values are constant is up to you, but if you make it constant, you can’t change it, such as adding a motor to a bike or skateboard. 😊

Setup your classes to contain inheritance with the following functionality.

- **Accessors/mutators for the private and protected members**
- **static const string** getVehicle(void) // returns a string description of the vehicle
- **int** getDistance(void) // returns an integer based on speed
- // Feel free to get creative.
(2 pts) Operator Overload

Make an operator overload member for comparing vehicles based on their distance. For example, you want to have the ability to compare two vehicles using the > and < operators, such as if (v > v1). In order to do this, you need to make an operator overload for > and < as a member in the Vehicle class, so all vehicles can be compared to one another.

```
bool operator>(const Vehicle &);
bool operator<(const Vehicle &);
```

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

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!!!