Overall, my project was extremely simple. Almost all of the program was similar to the last projects function and almost had the same functionality. I did struggle with what the program should have printed again, I wondered whether or not the move and insertion operators should be printing only from the car class and never even touch the vehicle class. Or whether or not when the constructors are called, it only calls the car class objects because there are no vehicle class objects. The biggest problem was definitely the virtuals and understanding the reason and purpose behind them. Conceptually, I understand what they do, but I do not understand what implementations there would be if I were to work on a side project myself.

////////////////////////////////

///// Constructor Tests /////

////////////////////////////////

**//1. Printing the beginning cout of the constructor tests**

Testing Derived Default ctor

Vehicle:Default-ctor

Car: Default-ctor

**//2. Calls the default constructors in the classes and sets it to the class object Car c1**

Testing Derived Parametrized ctor

Vehicle:Default-ctor

Car: Parametized-ctor

**//3. Testing Parametized constructor of only the Derived class, which sets the float array (*lla\_rno*) as parameters inside the Car Parametized constructor object (*Car c\_rno(lla\_rno))***

Testing Derived Copy ctor

Vehicle:Default-ctor

Car: Copy-ctor

**//4. Testing copy constructor of only the derived class, which calls the copy constructor (*c\_cpy*) and sets the parametized constructor object as its parameters (*c\_cpy(c\_rno))***

Testing Derived Assignment operator

Car: Assignment

**//5. Testing Assignment operator of only the derived class, which assigns the default constructor class object equal to the copy constructor object**

**/\* From my understanding, all of the Vehicle prints should not print any of their constructors except for the default, the tests is only for the parametized constructor, as there are no Vehicle class objects in the main. If I were to make them call the constructors I would add an option inside my class constructors such as :Vehicle(*parameter)* after the declaration of the constructors in the cpp**

**\*/**

/////////////////////////////////

///// Polymorphism Tests /////

/////////////////////////////////

**//6. Printing cout which declares the tests of polymorphism**

Testing VIRTUAL Move Function for DERIVED Class Objects

Car: DRIVE to destination [37.77, 122.42, 52] , with throttle @ 75

**//7. Testing Virtual move function from the derived class, which is calling the move function. Move function takes in the new location (*lla\_new*) and assigns it to the m\_lla of the first class objected called(c1)**

Testing Insertion operator<< Overload for BASE Class Objects

Car: @39.54 119.82 4500

**//8. Testing insertion operator, which prints out the class object (c\_rno)**

///////////////////////////////////////////////////

///// Polymorphic Base Class Pointer Tests /////

///////////////////////////////////////////////////

**//9. Printing cout, which signifies beginning of polymorphic base class pointers**

Testing VIRTUAL Move Function on Base Class Pointers

Car: DRIVE to destination [37.77, 122.42, 52] , with throttle @ 75

Car: DRIVE to destination [37.77, 122.42, 52] , with throttle @ 75

Car: DRIVE to destination [37.77, 122.42, 52] , with throttle @ 75

**//10. Testing virtual move function from base class, which takes a Vehicle class array pointers and calls the move function, which sets the values of lla\_new to the Vehicle array, which calls upon the Car virtual move.**

Testing Insertion operator<< Overload for Base Class Pointers

Car: @37.77 122.42 52

Car: @37.77 122.42 52

Car: @37.77 122.42 52

**//11. Testing insertion operator overload, which prints the Vehicle array 3 times with the values of lla\_new**

////////////////////////////

///// Tests Done /////

////////////////////////////

**//12. Printing cout, which signifies that all testing is done**

Car: Dtor

Vehicle: Dtor

Car: Dtor

Vehicle: Dtor

Car: Dtor

Vehicle: Dtor

**//13. All destructors are called, which removes all class objects inside the main after the program is finished with them.**