CST8219 – C++ Programming
Lab 4

Introduction:
The goal of this lab is to practice the copy constructor, pass by reference, default parameters, and separating classes into header and implementation files.

Reference:
Week 4 Powerpoint materials on Brightspace. There are many reference websites at the end of the powerpoint slides.

Steps:

1. Use your project from lab 2. Open git bash and navigate to your project on your hard drive. In git bash, type **git branch** to see that you are on the **week3** branch. This means that your current project is the result of applying all the commits in time. Type the command: **git branch Week4**

2. This makes an alternate path for building your files. Type git branch and you should now see **master**, and **Week3, and Week4** in the list. There is a star next to master meaning that the next commit gets added to that path. Type **git checkout Week4** so that git will now add commits to the Week4 path instead.

3. Create a Vehicle.h header file, and move your class Vehicle { }; declaration into the header file. Remove the class declaration from week2.cpp. You should add "Vehicle.h" to the list of files in your CMakeLists.txt in the line:
   add_executable(… "week2.cpp" "Vehicle.h")
   You should also add Vehicle.h to git tracking with the command: **git add Vehicle.h**
   Also, add a compiler guard at the top of the file: ***#pragma once***

4. Create a Vehicle.cpp file to put your function body declarations. Remember to add it to CMakeLists.txt and add it to git. Copy the function bodies from the Vehicle.h to Vehicle.cpp, and add the Vehicle:: scope specifier to all of the function names. That means:

   Vehicle () : Vehicle (0, 0) { }

   becomes:

   Vehicle () ; in the header file

   Vehicle:: Vehicle ():Vehicle (0, 0) { } in the .cpp file.

5. Declare a function void printVehicle (void); in the header file, and declare the function body in the .cpp file:

void* Vehicle*::print*Vehicle*() { }

It should work like the printFraction() in the examples in Powerpoint and print out its variables. It's the equivalent of a toString() in Java.

6. Create 2 copy constructors in the Vehicle class. One should copy by reference:

   Vehicle (Vehicle &) in header file

   Vehicle:: Vehicle (Vehicle &copy) in the .cpp file

   The other should copy by pointer:

   Vehicle (Vehicle *); in the header file

   Hint: try to chain the copy constructor to the reference & copy constructor above in the .cpp file :

   Vehicle:: Vehicle (Vehicle *copy) : Vehicle (???) { }

   How do you get the contents of the pointer?

7. In the week2.cpp file, create a function called

   void Create*Vehicle*(*Vehicle *&v, int w, int d) ;

   w has default value of 4,

   d has default value of 2.

   The function should modify the parameter **v** so that its variables are set to the parameters passed in. You will have to modify the Vehicle class to add getters and setters for the number of wheels and doors.

8. Change your main function so that it looks like this:

```
int main(int argc, char **argv)
{
        Vehicle original ;    //empty constructor no ( )

        Vehicle copy(original); // copy constructor by reference

        Vehicle secondCopy( &original ); //copy constructor by pointer
        copy.printVehicle ();
        CreateVehicle(copy, 2); //wheels is 2, everything else is default value
        copy.printVehicle();
        CreateVehicle(copy, 2, 3); //wheels is 2, doors is 3
        copy.printVehicle ();
        copy = secondCopy;
        copy.printVehicle();    // copy is same as second copy
        return 0;
}
```

9. Once you are finished, use git bash to commit your work to the Week4 branch:
   git commit -am "Finished Week 4"
   Create a zip file containing everything in your week2 directory and submit it on Brightspace. Make sure it includes week2.cpp, Vehicle.h, Vehicle.cpp, CMakeLists.txt, the .git folder.

Marks: (total of 10)

Your header file has the Vehicle class definition and no function bodies +1

Your header file has a compiler guard: #pragma once, or #ifndef, #define, #endif +1

Your implementation file (.cpp) has the function bodies with the Vehicle:: scope specifier in front of all of the function names +2

You have 2 copy constructors, one by reference, one by pointer +2

The CreateVehicle( ) function uses default parameters correctly +1

The main function is the one supplied in this lab document. It correctly prints out the values of the vehicle. The values of the vehicle change after each call to CreateVehicle() +3