CST8219 – C++ Programming
Lab 7

Introduction:
The goal of this lab is to practice templates, and using the standard template library

Reference:
Week 7 Powerpoint materials on Brightspace. There are many reference websites at the end of the powerpoint slides.

Steps:

1. Take your classes from Lab 6 (HybridVehicle, ElectricVehicle, GasolineVehicle, Vehicle) and make them templated so that the variables engineEfficiency, currentCharge, maxCharge, currentGasoline, and maxGasoline can be whatever data type that passed in.

2. Take your testVehicle function from lab 6 and replace it with this templated function:

```
template<class T >
T testVehicle(T pVehicle, const char *vehicleName)
{
        cout << vehicleName << "'s range is: " << pVehicle->calculateRange() << endl;
        pVehicle->drive(150); //drive 150 km
        cout << vehicleName << "'s energy left is: " << pVehicle->percentEnergyRemaining() << endl;
        cout << vehicleName << "'s range is now: " << pVehicle->calculateRange() << endl;

        return pVehicle;
}
```

Now when your main function calls:

delete testVehicle(new GasolineVehicle(50, 7.1), "Corolla");

it is computing that class T is GasolineVehicle*, so it creates a version of testVehicle for that class. This means that you don't have to use inheritance and a virtual function pointer table to write a common algorithm, so go to the Vehicle class and ***erase*** the abstract functions:

```
virtual float calculateRange() = 0;
virtual float percentEnergyRemaining() = 0;
virtual void drive(float km) = 0;
```

also remove the virtual keyword in front of the ~Vehicle() destructor.

The compiler will also detect that calling:

delete testVehicle(new HybridVehicle(42, 4.3, 8.8, 22.0), "Prius");

passes in a HybridVehicle* as the parameter T, so it will also call the right functions for the HybridVehicle class.

3. Remove all virtual keywords in your code and verify that the code still outputs the same result as lab 6. However, now that you are not using virtual functions, the functions can all be inlined. Take all the function bodies out of the .cpp files and put them in the .h files. Put the inline keyword in front of every function in the classes.

4. Write a templated min(T a, T b) and max(T a, T b) function that takes two parameters and returns the smaller one for min, and the larger one for the max function. Put these functions in a namespace called Helper. Use step #2 as an example of how to write a templated function.

5. Add a function to the week2.cpp file called testTemplateLibrary(). It should declare a vector<> of type float, and initialize it to an initializer list of { 5.0f, 4.0f, 3.0f, 2.0f, 1.0f }. In this function, write a loop that iterates over all the elements to print them out to cout. After that, write a while loop:

```
while(!vec.empty())
{
        cout<< “Last Element:” << vec.back(); //print the last element
        vec.pop_back(); // remove the last element
}
//End of step 5
```

6. Replace your current main() function with this new main() function:

```
int main()
{                                                   //50L of gas, 7.1 L/100km
        delete  testVehicle(new GasolineVehicle<float>(50, 7.1), "Corolla");

                                                    //42 L of gas, 4.3 L/100km, 8.8kWh, 22 kWh/100km
        delete testVehicle(  new HybridVehicle<double>(42, 4.3, 8.8, 22.0), "Prius" );

                                                    //75 kWh, 16 kWh/100km
        delete testVehicle( new ElectricVehicle<int>(75, 16), "Tesla 3");

        cout << "min of 5 and 7 is:" << Helper::min(5, 7) << endl;
        cout << "max of 5 and 7 is:" << Helper::max(5, 7) << endl;

        cout << "min of A and B is:" << Helper::min('A', 'B') << endl;
        cout << "max of A and B is:" << Helper::max('A', 'B') << endl;

cout <<"min of string(Hello) and string(world) is:" << Helper::min(string("Hello"), string("World")) << endl;
cout <<"max of string(Hello) and string(world) is:"<< Helper::max(string("Hello"), string("World")) << endl;
        testTemplateLibrary();
        return 0;
}
```

7. Once you are finished, use git bash to create a new branch called “Week7”. Then commit your work to the branch: **git commit -am “Finished Week 7”**
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, GasolineVehicle.h, GasolineVehicle.cpp, ElectricVehicle.h, ElectricVehicle.cpp, HybridVehicle.h, HybridVehicle.cpp CMakeLists.txt, the .git folder.

Marks: (total of 10)

- The testVehicle function has proper template declaration +1
- The class hierarchy of HybridVehicle, GasolineVehicle, ElectricVehicle are all templated +3
- All function are inlined in the .h files +1
- The min( ) and max( ) functions are templated and return the correct value +2
- The min() and max() functions declared in a namespace called Helper +1
- The testTemplateLibrary() function prints out the elements in a for loop +1
- The testTemplateLibrary() function prints the elements in reverse order and removes the last element until empty. +1