Logan Leavitt

CS 202

# Project 4 Documentation

The source files for this project are: *Agency.cpp*, *Car.cpp*, *Sensor.cpp*, and *proj4.cpp*. *Agency.cpp* contains the definitions of the class Agency's member functions, and the same goes for *Car.cpp* and *Sensor.cpp*. The file *proj4.cpp* contains the main function, string functions, and other miscellaneous functions. I also wrote header files *Agency.h*, *Car.h*, *Sensor.h*, and *proj4.h* (which contains the string function prototypes and the *MAX_STR_SIZE* constant). I also included the makefile I used to compile the project, so the code can be compiled with just the command "make" and "make clean" can be run to remove the object files. The compiled executable is name *proj4*. When the program is run, the user is prompted with a menu that accomplishes all the functionalities outlined. Data must be read before any of the other options can be picked, so if the user attempts to try any of the other options before a file has been successfully read, they will instead just receive a warning message. Data is read with option 1, where the user is prompted with a file name, and the corresponding file is then passes to the agency object's *readAllData(std::ifstream&)* function. The agencies members and its car members members are then changed from their default values. I did not implement *m_zipcode* as a const int, but if I were to implement it I would most likely use a parameterized constructor to initialize *m_zipcode*. The other menu options work about as expected. Option 2 prints out the agency name, zipcode, and all its cars. Option 3 prints out the total number of sensors and the number of sensors by type. I had an issue early on where the static sensor counts were twice as large as they should have been because *Car::operator+(Sensor&)* member function incremented the sensor type

count a second time (the sensor type count was incremented the first time by the Sensor it took in as a parameter), but it ended up being a quick fix and now works as expected. Option 4 find the most expensive available car (based on m_finalprice) and asks the user if they want to rent it. If there are no available cars, the program says so. If the user does want to rent the car, they will be prompted for a first name and the cars status will be updated. Option 5 will exit the program. One issue I did not think of earlier is that I did not account for the user wanting to read in data a second time. If data from a new file is read in, the sensors in the cars from the old agency will be left over in the *agency* object, as well as most likely prevent some sensors from being added because of the limit of 3 sensors. So, if the user wants to read data from a new file, they will have to exit the program and run it again. Given more time, this is most likely the first thing I would change.