Nicholas Ang

CS 202 - 1101

Project 4 Documentation

Purpose: The purpose of this project is to design a program that can be able to take information from a file and manipulate it into classes. The program is supposed to have aspects of multiple files, a functional menu, and array manipulation. The program uses classes and pointers to get information from the file and be able to print it to the terminal and filter out the unavailable cars. The project also involves makefiles and organizing files into folders to make compiling easier and neater. The program is also supposed to be able to give an estimate for the price of renting a car and reserve a car. The project also includes overloading operators to be able to use operators for different things.

Design: The project is organized into four folders and a make file. Within the src folder is the proj4.cpp main source file and another folder holding all my other source files. The include file holds all my header files. The bin folder holds the executable and the lib folder holds the libraries and .o files. My main source file proj4.cpp only initializes an agency and prompts the user with the menu. The menu source file holds the menu function which is supposed to function by reading in all the cars, printing all the cars out, printing only the available cars, and being able to reserve a car. When reserving a car, the user has to input an index to the car to select the car. If it was unavailable, an error message would pop up. If it was available, it prompts the user for the number of days and prompts the user for a name. The agency source file holds the implementation of the methods used in the menu. The readAllCars function uses an operator overload to input all the cars and their information into the Car array of size five. The

printAllCars function loops to print each car's information and prints out the owner of the car if the car is unavailable at the agency. The Car Class source file holds the getters and setters of the car information. It also has the function implementation updatePrice to update the final price of a car based on the sensors they have. It also holds the printCar function which prints one car's information. The Sensor Class source file holds getters and setters for the type of sensor the car has. It also has constructors for the type of sensor that is read in. The my_string source file contains the functions that allow for string copying, string comparing, checking string length, and string concatenation.

Problems/Challenges: Some difficulties I had were with the sensors and reading in information from a file to the classes. I had to make another operator for reading into a file in the Car class to be able to read all the cars at once and get their sensors. I also had difficulties stopping at the '}' when reading in sensors. In the end, I looped it in a while loop while checking if the end character of the sensor was a '}'. If it was not a bracket, it would continue scanning in the sensors. I also had difficulties with the makefile not able to find a specific file so I put it into a directory that was easy to access.

Possible Changes: If I had more time to change stuff, I would try to make the program more like the sample. I would get rid of the "none" sensors that show up when printing to the terminal. I would also clean up the code a bit more and find a way to simplify finding a bracket to stop filing in sensor. I would try to organize the folders more thoroughly and create multiple libraries instead of one library.