**THERE WILL BE ON LATE SUBMISSIONS FOR THIS ASSIGNMENT.**

Please read this document in its entirety. There will be requirements scattered throughout the document.

**LEARNING OBJECTIVES:**

This assignment will give you practice working with the following concepts:

* Multiple files
* Command-line arguments
* Inheritance
* Reading in and printing out information
* Vectors
* Creating a working menu
* Many other concepts

**ASSIGNMENT OVERVIEW:**

Consider walking in a car dealership/rental office to either purchase a car/truck or rent a car. Suppose you could walk up to a kiosk sign in and view the available inventory for sale or rent, then choose the car/truck you want.

For this assignment you are going to write a system that will simulate this. Your system will read in a text file that will populate a vector of cars, trucks, and people. As stated in class you have the option to determine how much time you want to put in this assignment. Below I will explain each option. There will also be several options for extra credit.

**ACADEMIC INTEGRITY:**

This is an individual assignment. You may not receive help from anyone other than myself and the lab TA. Please review the academic integrity policy provided in the syllabus.

**Requirements:**

For this assignment, you must have at a minimal the following files:

1. driver.cpp – Your driver should have minimal amount of code in it.
2. Inventory.h and .cpp
3. Vehicle.h and .cpp
4. Car.h and .cpp
5. Truck.h and .cpp
6. Person.h and .cpp
7. Address.h and .cpp
8. Date.h and .cpp

I will share my header (.h) files with you, however, you are not required to do your assignment exactly like mine. Keep in mind there are some functionality that I am requiring you to do, that I have not implemented in my program. With that said I **AM** requiring you to use the same structure I used. If you choose to create another class you may do so, but the following is the minimal structure you will use for this assignment.

Inventory

Has-a

Person

Has-a

Has-a

Vehicle

Date

Address

Is-a

Truck

Car

**Composition (has-a relationship between classes)**

Person has an instance of Address and Date. This you have already done in lab. Also, Inventory has an instance of Person, Car, and Truck. --Depending on how you do your program, this could be Vehicle rather than Car, Truck.

**Inheritance (is-a relationship between classes)**

Car and Truck is a Vehicle

**Levels of Required Functionality:**

As stated above, you get to choose the amount of time and effort you want put in this assignment. Below are listed the levels you may choose from.

**OPTION 1**

85% (This means the highest grade you can receive on this assignment will be 85/100)

For this level you are required to read, from a file, the data I provide for the car and truck inventory. It must also be able to read, from a file, the name, address, and birthday of a person, aka, customer. This you have already done so this part should be easy. You are to store the information in vectors. The Inventory class must have a vector for cars that are for sale, cars for rent, trucks for sale, and for the customer information. When you read in the data for cars you must determine if the car is a rental. A car is a rental if it is no more than 2 years old, has no more than 20,000 miles on the odometer, and is in Excellent condition. To determine if the car is 2 years old or not you must get the current year from your computer. To get the time you should google “how to get the current year in C++”. You will get several examples that you can adapt to your program.

Lastly, your program should print the data for the cars and truck to a file. Your output should first print the size of each vector. It should then print each car for rent, each car for sale, each truck for sale. Each section must be separated by a row of “\*” or some other character. Your program must also output, to a separate file, the names, address, and birthday of the customers that were read in from the customer file. The input files will be determined by the command line arguments. The output file names should be hard coded when opening the file in your program. The names of the output files **MUST** be “vehicleOut85.txt” and “customerOut85.txt”.

**OPTION 2**

95% (This means the highest grade you can receive on this assignment will be 95/100)

You are to complete everything for the 85% plus you are to add a menu that has the functionality similar to what I demonstrated in class. If you choose this option the output file names **MUST** be “vehicleOut95.txt” and “customerOut95.txt”.

Your menu should do the following:

1. Print a warm greeting to the customer. This should be catchy and neat. Be creative. I love creativity.
2. It should ask the user for their first and last name, email address, and birthday. You should let the user know you do not plan to sell their personal data. Thank the user for sharing the information after you have confirmed the birthday data is within the appropriate parameters. As an example, the birth month must be between 1 – 12, etc. If the birthday is wrong, ask them to reenter the birthday information.
3. You should then ask the user to choose what they want to do. See a list of cars, or trucks, or quit. If they want to see cars then ask if they want to see rental cars or cars for sale. You get to choose what type of response the user types (characters or numbers), but you must check to determine if their response is within the appropriate parameters. If they choose to quit then thank the customer for visiting your store and anything else you want them to know and end the program.
4. Once you know what the user wants to see, display the information about the cars or trucks for the user.
5. Once you have displayed the inventory, ask the user to choose the id of the vehicle they want to test drive or rent. You should check that they chose an appropriate id number. If they did not tell them the number was incorrect, display the inventory again and ask them to choose again.
6. Once the user has appropriately chosen a vehicle. If they are renting then tell the rental car is being prepared for them and they should go to the customer service counter to complete the remaining paper work. If they are looking at a truck or car for sale, thank them for their business and that a sales representative will be with them momentarily.

You should test this rigorously, because I will.

**OPTION 3**

100%

In addition to option 2 above, you are to add the functionality that will remove the cars that are rented and/or sold from the appropriate inventory. This means if the user chose sale, you will need to add something to the menu that ask the user if they liked the car or truck, they test drove. (We will pretend enough time for them to do a test drive went by.) If they did like the car or truck and want to purchase it you will need to remove the vehicle from the inventory. If they did not, ask the user if they are interested in another vehicle. Repeat the process of showing the available inventory. After the user has finished you should print to an output file the inventory, as described in option 1. This output should reflect the removed car or truck. The output file name for this portion will be hard coded when opening the output file. You should name the file “vehicleOut100.txt” and the customer output name should be “customerOut100.txt”.

**Other requirements all options must do:**

1. Use initialization with as many constructors as possible.
2. Comments must be place in all header files.
3. Must use header guards for all header files.
4. Must provide a makefile that has a make run that will compile and run the program.
5. Makefile must also have a make clean that removes all .o files as well as the executable.
6. When compiling you should have no errors nor warnings. A warning is just as bad as an error.
7. You must have at least one static variable in your program. (hint: each vehicle must have an identification number)
8. The structure of your program must use inheritance and composition
9. When printing information, each class should be responsible for printing its own data. You should have do this in the person, address, and date assignment so this should not be new.

**EC**

There are three options for extra credit.

**EC1 (5 points) (only available for option 3)**

Add the functionality that will read, from a file, the names of customers that want their names to be removed from the customer list. This file name will be provided by command line arguments. You will also need to print, to an output file, the new customer names. You should first print how many names are in the customer list, then the name, address, email address, birth date. The name of this output file will be “customerOutEC1.txt”

**EC2(5 points) (only available for option 3)**

In the menu, when the system ask for the users first and last names, email, and date of birth, check the inventory to determine if the user is already in the system. If they are, confirm their address information is correct. If not ask the user for the correct information. After you have successfully changed the customer information print the customer information to the following file: “customerOutEC2.txt”

**EC3 (5 Points) (available for all options)**

Use Polymorphism. We will go over this in class. This will require you to us a pointer of type Vehicle to represent the instances of Car and Truck. I will determine this has been done by code review.

**Testing your program:**

You should test your program on the SOC servers prior to submitting. I will grade the assignments on the SOC server. Remember compilers on different computer architectures have slightly different compile warnings/errors. Some compiles will ignore minor problems while others will not. In order for me to grade fairly, I will grade your assignments on the SOC servers.

**FORMATTING:**

You will need to add a header to each of your files similar to the following:

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*Your name

\*CPSC 1020 Sm19

\*Your email

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Your program should compile with no warnings and no errors. If your program does not compile the highest grade you can get for the assignment will be 20. If your program compiles but has warnings, there will be a deduction up to 20 points.

* Your code should be well documented. (comments)
  + Detailed comments should be in the header files above each function
  + Less detailed comments should be in the .cpp files
* There should be no lines of code longer than 80 characters.
* You should use proper and consistent indention.

**HANDIN:**

Use handin.cs.clemson.edu to submit your files. I have created the following buckets:

PA2\_85 – for option1

PA2\_95 – for option2

PA2\_100 – for option3

PA2EC1 – for Extra Credit 1

PA2EC2 – for Extra Credit 2

PA2EC3 – for Extra Credit 3

**\*\*\*\*\*\*Please only submit to one handin bucket\*\*\*\*\*\***

Things to do prior to handing in your files:

1. **Test your program on the SoC servers**. I will not accept the excuse “It compiled on my computer.” I test programming assignments on the SoC servers.
2. Tar zip your files naming the tarred file PA1.tar.gz. Using a terminal type:

tar –cvzf **PA2.tar.gz** \*

The (\*) is a wildcard and tells the tar utility to tar everything in the folder.

1. You should also provide a **README** that consist of the following.

* A short description of any problems you encountered when writing this program.
* How you solved the problems you encountered.
* Your thoughts on the assignment. This is your opportunity to tell me if you like the assignment or not. What you did or did not like about the assignment. Anything you want to tell me.

1. It is your responsibility to make sure you submit all of the appropriate files and that the files are in working order. Hand-in allows you to check your files after submission. You must check your files. If the files are corrupt you should resubmit. This could take time so be sure not to wait to the minute to submit. Any submissions with corrupt or missing files will result in a 0 on the assignment.
2. Your code should be well documented. Below are some guidelines on how to comment your code. When implementing the programs functions you should also have comments within the code to explain what your algorithm is doing. You do not need to comment each line nor code that is obvious. Code that is not obvious you will need to explain. (See comment note above also)

Here are some guide lines for documenting the code in your assignment.

Before each function, in the header file, you should have a detailed description of what the overall function does. To borrow from another student’s code, here is an example of overall function description.

/\* Parameters: img - image\_t pointer array holding the image data for  
 \*                   each of the input files  
 \* Return:     output - image\_t struct containing output image data  
 \* This function averages every pixels rbg values from each of the   
 \* input images and puts those averages into a single output image  
 \*/

You are not required to do yours exactly like this one. However you must provide this type of information.

Also, if you include comments in the body of the function (and you should) they must be placed above the line of code not beside the code.

Example:

Bad

if(something) //This is a comment

{

do something;

}

Good

//This is a comment

if(something)

{

do something;

}