**CSCI 1411 – Lab 11– PA3 Prep and CLion**

**Goals:**

* Use Eclipse to compile a program with multiple files
* Provide an outline for CSCI 1410 PA3

**Development Environment:** all students must use Clion

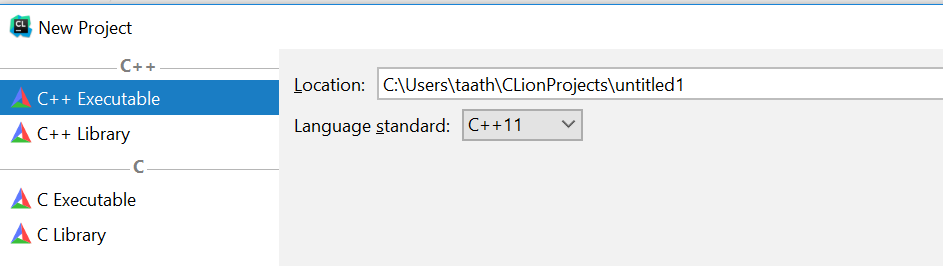
**Skills:** Dynamic Arrays of Characters forming CStrings, definition of a namespace to allow use of a std::string versus an Example::string.

**Reading**: CSCI 1410 PA3 Prep/CSCI 1410 PA3

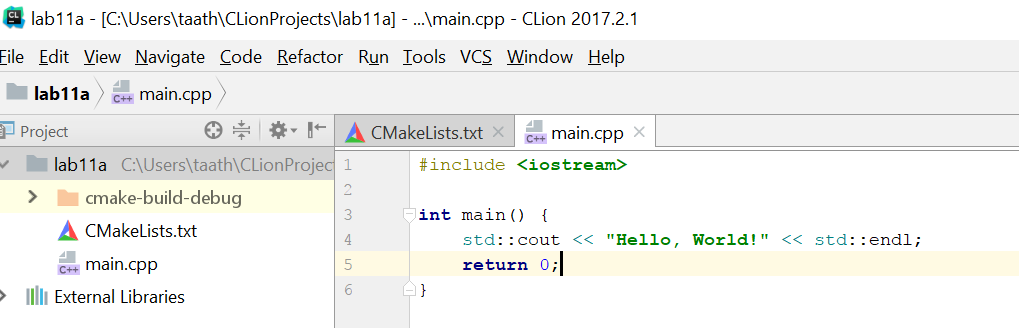
**Deliverables:** 1) This lab with two screen shots 2) lastnameFirstLab10.cpp

**Part I – Skills Practice (10 points)**

* Unlike Visual Studios which has its own built in C++ compiler as part of the program,. C-Lion is an application that is just a front-end editor and environment. Clion is a professional Individual Development Environment with an annual license fee, but is free to students. The company that makes CLion also makes a number of other IDEs for other languages. It is important that you have used many different IDEs so when you apply to jobs, you can easily configure and use whichever IDE your company requires. CLion requires an updated Java Run Time Environment and will run on Windows, Mac and many linux platforms.
* If loading on your own PC system, you would first load mingw, which is a minimum linux emulator. It takes the g++ compiler and makes a Windows based executable out of it.
* Then you would download Jetbrains CLion. That will require you to set up an account using your .edu account. You get one year free access, renewable each year as long as you are a student.
* This has already been done on the Windows machines in the lab.
* Go to the Window->Jetbrains->Clion
* Go to File->New Project
* Change the untitled name to Lab11a



* You will get a screen like this:



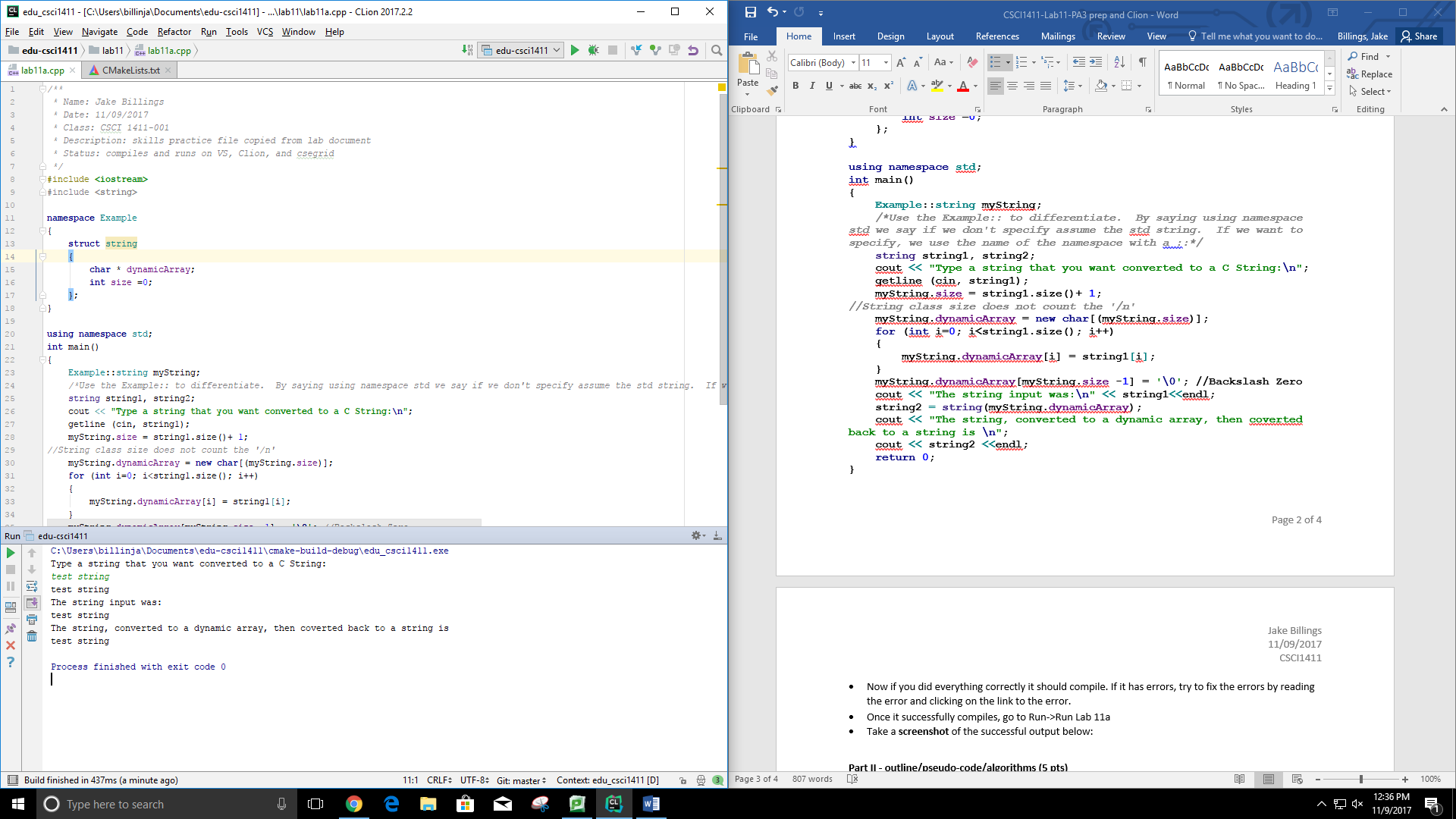
* Replace the main.cpp text with this

**#include <iostream>  
#include <string>  
  
namespace Example  
{  
 struct string**

**{  
 char \* dynamicArray;  
 int size =0;  
 };  
}  
  
using namespace std;  
int main()  
{  
 Example::string myString;  
 */\*Use the Example:: to differentiate. By saying using namespace std we say if we don't specify assume the std string. If we want to specify, we use the name of the namespace with a ::\*/* string string1, string2;  
 cout << "Type a string that you want converted to a C String:\n";  
 getline (cin, string1);  
 myString.size = string1.size()+ 1;**

***//String class size does not count the '/n'* myString.dynamicArray = new char[(myString.size)];  
 for (int i=0; i<string1.size(); i++)  
 {  
 myString.dynamicArray[i] = string1[i];  
 }  
 myString.dynamicArray[myString.size -1] = '\0'; //Backslash Zero  
 cout << "The string input was:\n" << string1<<endl;  
 string2 = string(myString.dynamicArray);  
 cout << "The string, converted to a dynamic array, then coverted back to a string is \n";  
 cout << string2 <<endl;  
 return 0;  
}**

* Now if you did everything correctly it should compile. If it has errors, try to fix the errors by reading the error and clicking on the link to the error.
* Once it successfully compiles, go to Run->Run Lab 11a
* Take a **screenshot** of the successful output below:



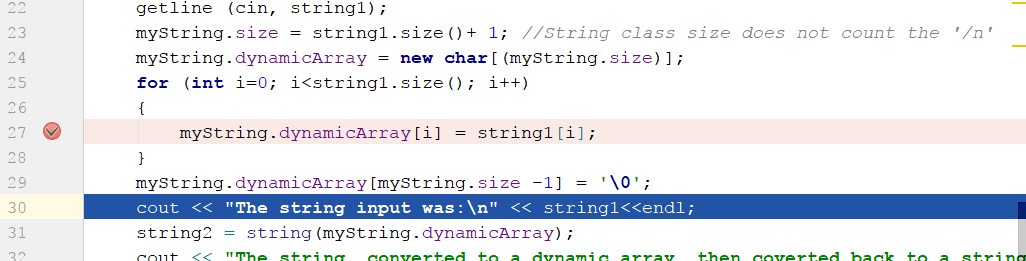
**Part II - outline/pseudo-code/algorithms (5 pts)**

Using the debugger is a key skill that you will use often, especially when dealing with pointers. Every time you use a new Individual Development Environment (IDE), you should learn how to use the debugger, and what kind of information this debugger will give you.

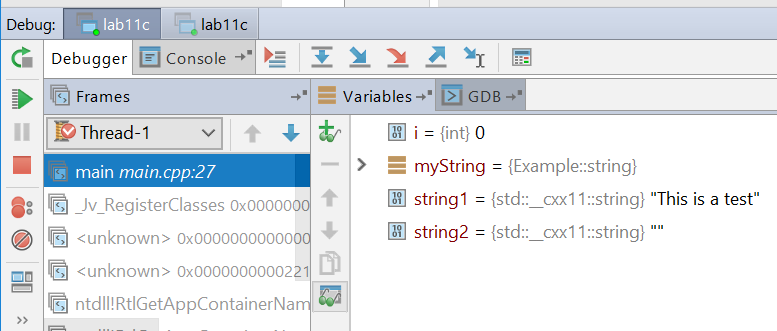
* In the main.cpp, click just to the right of the number on the line

myString.dynamicArray[i] = string1[i];

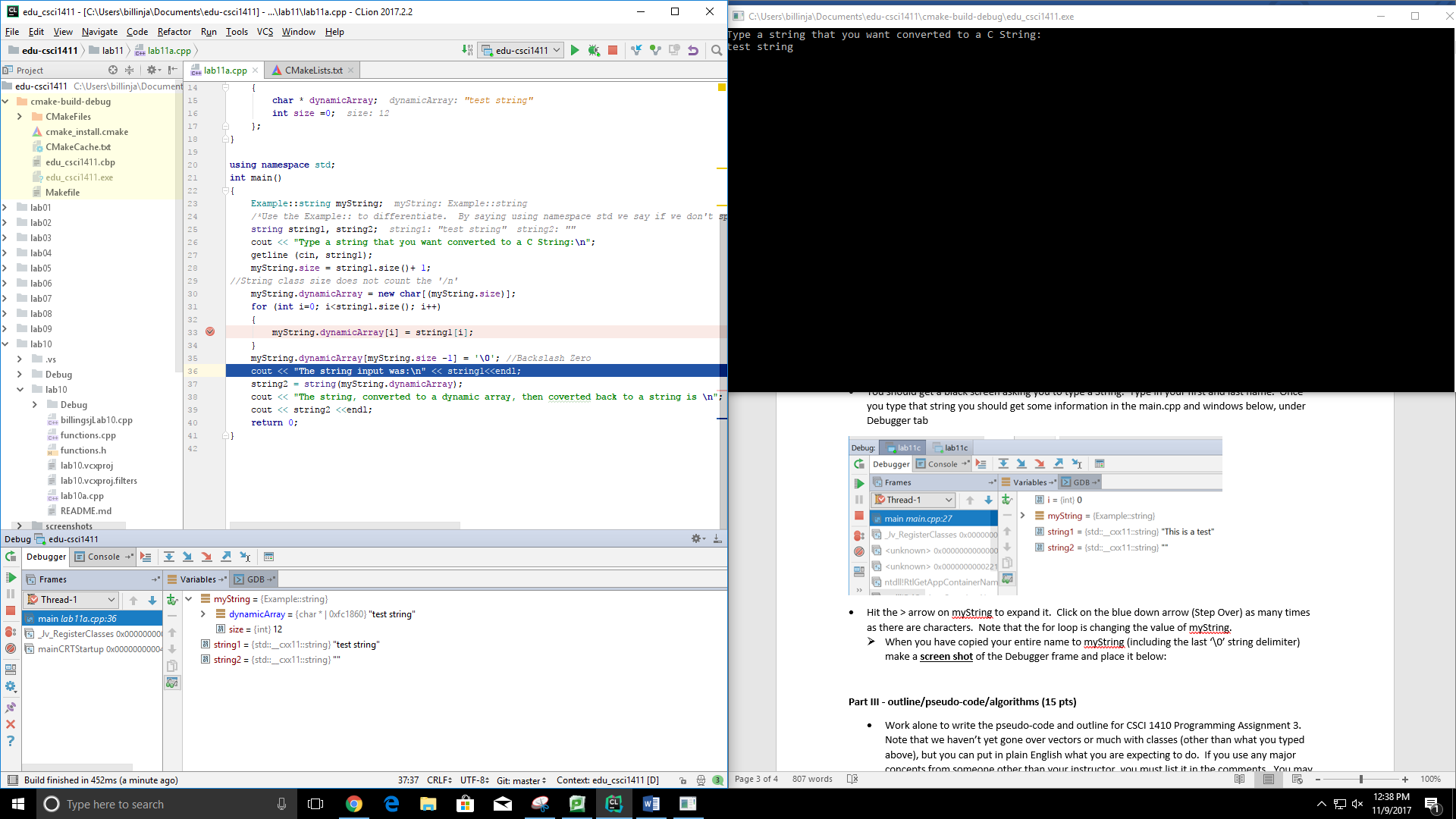
* and you will get a red circle



* Now Run->Build
* Run Debug Lab 11a
* You should get a black screen asking you to type a String. Type in your first and last name. Once you type that string you should get some information in the main.cpp and windows below, under Debugger tab



* Hit the > arrow on myString to expand it. Click on the blue down arrow (Step Over) as many times as there are characters. Note that the for loop is changing the value of myString.
* When you have copied your entire name to myString (including the last ‘\0’ string delimiter) make a **screen shot** of the Debugger frame and place it below:



**Part III - outline/pseudo-code/algorithms (15 pts)**

* Work alone to write the pseudo-code and outline for CSCI 1410 Programming Assignment 3. Note that we haven’t yet gone over vectors or much with classes (other than what you typed above), but you can put in plain English what you are expecting to do. If you use any major concepts from someone other than your instructor, you must list it in the comments. You may create this outline in any IDE.

/\* Name:  
Class: CSCI 1411-00X  
Description: [fill in description]

Lab Partner:  
Status: successfully compiled and run on csegrid [Since this is an outline, it is not expected to compile, but the status should say so!/\*

* If you put a full effort on this lab, you will be completing a requirement that will also be graded in CSCI 1410. Upload only the .cpp and .h files to Canvas.

My complete version of PA3 is available at: <https://github.com/jake-billings/edu-c-demo>

Plan:

*/\*\*  
 \* Name: Jake Billings  
 \* Date: 08/02/2017  
 \* Class: CSCI 1411-001  
 \* Description: Implementation of getters/setters for the dealer class from PA3  
 \* Status: compiles and runs on VS, Clion, and csegrid  
 \*/  
//Include the dealer header file  
/\*\*  
 \* setName() sets the DealerName attribute of a given Dealer object.  
 \*  
 \* @param input std::string value to set the DealerName to  
 \*/  
/\*\*  
 \* setAddress() sets the DealerAddress attribute of a given Dealer object.  
 \*  
 \* @param input std::string value to set the DealerAddress to  
 \*/  
/\*\*  
 \* getName() returns the DealerName attribute of a given Dealer object.  
 \*/  
/\*\*  
 \* getAddress() returns the DealerName attribute of a given Dealer object.  
 \*/  
/\*\*  
 \* Define a default constructor to init an empty Dealer object  
 \*/  
/\*\*  
 \* Define a constructor that sets the name of a Dealer using the setName() method  
 \* @param iName The name to set the dealer to  
 \*/*

*/\*\*  
 \* Name: Jake Billings  
 \* Date: 08/02/2017  
 \* Class: CSCI 1411-001  
 \* Description: Function implementations for the car dealership functions  
 \* Status: compiles and runs on VS, Clion, and csegrid  
 \*/  
//Include cout, cin, fileout, filein, and function headers  
/\*\*  
 \* Implement the user interface loop for the menu system. Called my main() in main.cpp  
 \*  
 \* @param inventory - A vector reference to the inventory we are managing  
 \* @return bool - False if the program should exit. True for another loop.  
 \*/  
 //Print menu  
 //Read the user's choice  
 //Select the correct function for a given choice  
/\*\*  
 \* Implement the display inventory function  
 \*  
 \* @param inventory The inventory we are managing  
 \*/  
 //If there aren't any vehicles, tell the user.  
 //Print vehicle information for each vehicle  
/\*\*  
 \* Implement the add inventory function  
 \*  
 \* @param inventory The inventory we are managing  
 \*/  
 //Create an empty vehicle  
 // Use the >> operator to read the vehicle from cin  
 //Append the vehicle to the inventory  
 //Notify the user of our success  
/\*\*  
 \* Implement the update inventory function  
 \*  
 \* @param inventory The inventory we are managing  
 \*/  
 //First, display the inventory  
 //Ask the user which vehicle they want to edit  
 //Tell the user we're starting  
 //Take the new info  
 //Tell the user we updated  
/\*\*  
 \* Implement the delete inventory function  
 \*  
 \* @param inventory The inventory we are managing  
 \*/  
 //First, display the inventory  
 //Have the user pick a vehicle  
 //Erase the vehicle by id  
 //Tell the user what we did  
/\*\*  
 \* Implement the sort inventory function  
 \*  
 \* @param inventory The inventory we are managing  
 \*/  
 //Using bubble sort, arrange Vehicle from tallest to shortest in the inventory vector.  
 //Tell the user we're done  
/\*\*  
 \* Implement the search inventory function  
 \*  
 \* @param inventory The inventory we are managing  
 \*/  
 //Ask the user what model to search for  
 //Iterate through the inventory and add matches to a new vector  
 //If no results are found, tell the user. If they are, print them out.  
/\*\*  
 \* Implement the read inventory from file function  
 \*  
 \* @param inventory The inventory we are managing  
 \*/  
 //Open a file input stream  
 //Keep count  
 //Until we reach the end of the file, deserialize vehicles and add them to inventory  
 //Tell the user we're done  
/\*\*  
 \* Implement the write inventory to file function  
 \*  
 \* @param inventory The inventory we are managing  
 \*/  
 //Open an output stream  
 //For each vehicle, write it to the file  
 //Flush the output  
 //Close the output stream  
 //Tell the user we're done  
/\*\*  
 \* Implement the unknown option function  
 \*  
 \* @param inventory The inventory we are managing  
 \*/  
 //Print a message stating we don't know the option*

*/\*\*  
 \* Name: Jake Billings  
 \* Date: 08/02/2017  
 \* Class: CSCI 1411-001  
 \* Description: Data system for a car dealership  
 \* Status: compiles and runs on VS, Clion, and csegrid  
 \*/  
//Include cout, cin, and functions  
/\*\*  
 \* Implement main() as a while(true) loop that calls userInterfaceLoop until exit  
 \*  
 \* @return int 0 - Everything executed without error  
 \*/  
 //Declare a vector to hold our inventory  
 //Iterate forever on the user interface loop unless, it sends us an exit signal  
 // by returning false  
 //Return 0 because everything went okay*

*//  
// Created by Jake Billings on 8/2/17.  
//  
/\*\*  
 \* Implement the default vehicle constructor  
 \*/  
/\*\*  
 \* Implement the full vehicle constructor  
 \*/  
//Implement vehicle setters  
//Implement vehicle getters  
/\*\*  
 \* Implement the << operator as a method to serialize this object  
 \*  
 \* Contains cout/cin specific content for the purpose of user input.  
 \*  
 \* @param out The output stream to serialize to  
 \* @param car1 The vehicle to serialize to out  
 \* @return out  
 \*/  
/\*\*  
 \* Implement the >> operator as a method to deserialize this object  
 \*  
 \* Contains cout/cin specific content for the purpose of user input.  
 \*  
 \* @param out The input stream to read from  
 \* @param car1 The vehicle to dump the incoming data to  
 \* @return out (input)  
 \*/  
 //Read strings then call setters  
/\*\*  
 \* Implement a method to serialize this object without console output  
 \*  
 \* @param out The output stream to serialize to  
 \*/  
 //Call getters and output the values to the output stream  
 //Dereference the dealer  
 //Dump the dealer data  
/\*\*  
 \* Implement a method to deserialize this object without console output  
 \*  
 \* @param in The input stream to serialize to  
 \*/  
 //Read the values from the input stream to the properties  
 //Read strings then call setters*