Part I: PA3 Outline (10 points). Create an outline in comments/psuedocode for the programming assignment below. Place everything in a single file (for the outline), but use the following comments to show how you will separate the files when you hand in Part II:

//your name

//Dealer.h

//Dealer.cpp

~~//functions.cpp~~ (there is no function.cpp)

// function.h

//main.cpp

Ensure your name is in comments in each file AND each file is fully commented. You should define the two classes in your Dealer.h section (based on the names and types in Part II). Then you should provide pseudocode for what these functions do in your Dealer.cpp and functions.cpp sections. Your main.cpp will just consist of declaring a vector of type Dealer (if you don't know how to do this yet, just place comments in plain English). Then you will have a menu that calls functions.

Part II: PA3: Car Dealership (40 points)

For Programming Assignment 3 you will be creating a program to manage cars in a dealership. This will again be a menu driven system. You will make a vector of Dealers (there can be zero to many dealers). Each dealer can have zero to many cars on the lot, represented not by a vector, but by a dynamic array. The following is your menu:

1. Read Dealers and Cars from file
2. Display Dealers
3. Choose a Dealer Number, Display Cars
4. Choose a Dealer Number, Add Car
5. Choose a Dealer Number, List Cars and Modify a Car
6. Choose a Dealer, Sort cars by VIN (EXTRA CREDIT)
7. Write Dealers and Cars to file (EXTRA CREDIT)
8. Exit

Your program will be object oriented (using classes, not structs) with the following classes defined in Dealer and Car (there is no Car.h)

Dealer.h

```
#ifndef _DEALER
#define _DEALER

class Dealer {

private:

        Dealer ( )

        // Don't forget to add a new car pointer when you use the Dealer default / constructor

        Dealer ( string _dealerName, int _dealerNumber);

    string dealerName;
    int dealerNumber;
    int numberOfCars;

    public:
Car *carArrayPtr;
```

- //pointer to a Dynamic Car Array (Note: make sure this is set to nullptr in the constructors). This would normally be a private, but this takes away a little complexity)

```
void setDealerName (string _dealerName);

void setDealerNumber (string _dealerNumber);

void setNumberOfCars ( int _numberOfCars);

string getDealerName ( );

int getDealerNumber ( );

int getNumberOfCars( );

+ friend ostream & operator << (ostream &out, Dealer _dealer);

            friend ostream & operator << (ostream & out, Dealer & _dealer)

        //Print the Dealer Name and Number and Blank line for a specific dealer.

}

#endif //_DEALER
```

class Car
{
#ifndef _CAR
#define _CAR
private:

string vin;
string make;
string model;
int year;
doubler price;

public:

Car( );

Car(string _vin, string _make, string model, int year, double price);

+ getVIN( ):string

+ getMake( ):string

+ getModel( ):string

+ getYear( ):int

+ getPrice( ):double

+ setVIN(_VIN:string):void

+ setMake(_make:string):void

+ setModel(_model:string):void

+ setYear(_year:int):void

+ setPrice(_price:double):void

+ friend operator << (out: ostream &, Car: _car):ostream &
friend ostream & **operator** << (ostream & out, Car c)
//Print the array index+1, VIN, Make, Model, Year, Price and Blank line for a specific car.

#endif

You will have four files for your program (Use these file names!): main.cpp, functions.h, Dealer.h, Dealer.cpp. Place into a file folder named LastnamePA3, then zip the content and hand in a zip file to canvas

- main.cpp: this will be your driver file.
- functions.h: this will contain your functions for each of the menu items (the prototypes and the function code for the menu options).
- Dealer.h: this will contain the class declarations for car and dealer including the operator <<.
- Dealer.cpp: this will contain the class implementations for car and dealer.

Remember, you do not #include "Dealer.cpp", just #include "functions.h" and #include "Dealer.h"

You will be storing your dealer objects in a **vector**. In the main function, you will create a vector of Dealers. When the menu is called, you will have to check your vector and ensure that it is not empty (size == 0) before referring to the index number of the vector. Your Dealer vector points to a **dynamic array** of cars (remember dynamic arrays use the keyword new). When you access the cars class you should verify that it is not set to nullptr before accessing it.

You may not use any global variables, so must pass vectors and arrays into parameters. If you want to change a vector you must pass by reference (&). If you have arrays, you must pass size and the array, but arrays are already pointers, so do not use & when passing an array.

Each menu item will have a corresponding function, and the definition of the function will be found in the file functions.h. Also, your operator << function for Dealer and Car will be implemented in your Dealer.h.

All the options in main will be done in the functions.h and not in main (except asking for what menu option the user wants).

The following are the details for each of your menu options:

1. Read Dealers and Cars from file
2. Display Dealers
3. Choose a Dealer Number, Display Cars
4. Choose a Dealer Number, Add Car
5. Choose a Dealer Number, List Cars and Modify a Car
6. Choose a Dealer, Sort cars by VIN (EXTRA CREDIT)
7. Write Dealers and Cars to file (EXTRA CREDIT)
8. Exit

1. Pass the vector of dealers (from main) into the function by reference. Each Dealer will have a name and number followed by the number of cars in the dealership (on separate lines). Then you will read in a Vin, Make, Model, Year and Price for each car (on separate lines). You can assume that the number of cars as well as the Vin, Make,

Model, Year and Price will be complete for each car. Each dealer can have more then one car. Your file format will be as follows:

Dealer Name
Dealer Number
2
Car1 Vin
Car1 Make
Car1 Model
Car1 Year
Car1 Price
Car2 Vin
Car2 Make
Car2 Model
Car2 Year
Car2 Price
Dealer Name
Dealer Number
1
Car1 Vin
Car1 Make
Car1 Model
Car1 Year
Car1 Price

Don't forget, between << and a getline you must use a .ignore( ) (e.g. infile.ignore( ))
Here is an example in.txt file (You MUST read in this file name):

```
John Elway Dodge
123
2
VIN123
Dodge
Ram 1500 Quad Cab
2017
45186.00
VIN456
Chevrolet
Traverse Premier SUV
2017
47868.00
Tesla Cherry Creek
234
1
VIN789
Tesla
Model X
2017
105200
```

2. Pass the vector of dealers (from main) into the function by reference. Display the Dealer Name and Number for each dealer (put a blank line between dealers). Use the << operator with cout (cout << dealer[x];)

3. Pass the vector of dealers (from main) into the function by reference. Display the Dealer Names and Numbers (menu #2). Ask the user for the dealer number, display all of the cars by using a for loop (from zero to size of the car array) using the << operator with cout (cout << dealer[x].carArrayPtr[x];)

4. Pass the vector of dealers (from main) into the function by reference. a) Display the Dealer Names and Numbers (menu #2). b) Ask the user for the dealer number then the new car information. Follow the steps found in CSCI 1411 Lab 09 to increase the dynamic array. (Essentially you will make a new array, one bigger than the previous car array. Then you will copy everything over from the old car array to the new one. Then you will point the car pointer to the new array)

You can earn up to 20% **extra credit (48/40)** for completing 5 and 6 as long as you get at least 80% on the required tasks.

5. Pass the vector of dealers (from main) into the function by reference. Display the Dealer Names and Numbers (menu #2). Ask the dealer number, search for that number in the vector, list the cars for that dealer number. Ask which car has to be updated (chose by the index of the car) . Enter all the attributes for the car and update the car attributes.

6. (extra credit) Ask the user for the dealer number. Sort the cars by vin and display them (with blank lines between each car)

7. (extra credit) Pass the vector of dealers (from main) into the function by reference. Also pass in the filestream (from main) by reference. Call the output file out.txt. Write all of the Dealers and Cars to a separate output file in the same format as in item #1. This should include all changes to the vector and dynamic array (including add, delete, modify, sorting etc.)

8. Exit out of the program

What to Hand-In:

Submit to canvas main.cpp, functions.h, Dealer.h and Dealer.cpp and an in.txt files. Each must be complete with your name on it. If you complete the extra credit, you must have 7 menu items and this must be documented in your main.cpp.

Note: You must document ALL code that you took from ANY source (including the internet, your book, someone else). Failure to do so may result in a grade of zero for EACH student with "too similar" code. You can work together with big ideas, but all code MUST be your own.