COM S 213 – Fall 2004

Assignment #3

Honest Used Car Dealership

Due September 16, 2004

The main purpose of this assignment is to experience dynamically allocated arrays along with getting used to pointers.

You will need to create two classes:

1. An Automobile class
2. A Dealership class

We'll start with the `Automobile` class. This class is very simple and contains 5 member variables and two methods. The 5 member variables allow us to store information about the car's:

- Year of manufacture
- Make (Ford, Chevy, Honda, Toyota, etc.)
- Model (Taurus, Malibu, Accord, Camry, etc.)
- Mileage
- Price

It is up to you to pick the right data types for these fields. The class has two member functions which allow us to do the following:

- `setInfo()` which sets the values in the 5 member variables all at once
- `display()` which displays the 5 member variables all on one line

Next, we have the `Dealership` class. This class needs to have three private member variables: One to store a dynamically allocated array of `Automobile`s, a second to store the maximum number of elements in that array, and a third to indicate the index of the last item placed in the array. In addition to the three member variables, the `Dealership` class has 5 public methods:

- `initialize()` which sets all private member variables to "0"
- `setInventoryCount()` which takes a "size" parameter that specifies the number of elements to dynamically allocate to store inventory.
- `addToInventory()` which takes 5 parameters and utilizes `Automobile::setInfo()` to set the attributes of the next car in the inventory.
- `displayInventory()` which prints out a listing of all cars in inventory, utilizing `Automobile::display()` where appropriate.

- `freeInventory()` which frees all dynamically allocated memory and sets private member variables back to 0.

Some hints for the member functions of the `Dealership` class:

- `setInventoryCount()` should check to see if memory has already been allocated to the internal, dynamic array and free that memory after new memory has been successfully allocated.
- Before attempting to utilize the internal, dynamic array you should make sure that it contains a valid pointer.
- `addToInventory()` should make sure there is room left in the dynamic array before allowing an entry to be stored in the array. If no room is left, it should print out a message to that effect.

Finally, write a main function which utilizes the two classes you just wrote to test the dealership code. You should declare an instance of a `Dealership`, initialize it, set an appropriate inventory size, add cars to the inventory, print out the inventory and then free any dynamically allocated memory. You should try to add one more car than your maximum so that you can demonstrate that error condition being "caught".

Please feel free to email me with any questions!