COMP 356 Homework Assignment 4

Acknowledgment. This assignment was written by Prof. Tim Wahls, with minor changes by John MacCormick.

Implement a class `CarDealership` in C++ that keeps track of the automobile inventory of a car dealership. The dealership sells ordinary cars, sports cars and sport-utility vehicles. The information stored about all vehicles includes the VIN (Vehicle Identification Number), make, model, color and year of the vehicle. Note that VINs are not simple integers - they can contain letters. For sports cars, the engine horsepower is also stored. For sport-utility vehicles, the ground clearance and whether or not the vehicle has four wheel drive is also stored.

Although only these vehicle types are currently sold, your design must be flexible enough so that other vehicle types (vans, trucks etc.) can easily be added.

Your `CarDealership` class must include the following:

- a constructor that takes the maximum number of cars that can be in stock as a parameter
- a destructor that deallocates all heap-dynamic variables associated with an instance of the class.
- one or more member functions that add vehicles to the inventory. It must be possible to add a vehicle of each type to the inventory.
- a member function to display all vehicles in inventory. All information about each vehicle type must be displayed – i.e. the display for a sport-utility vehicle must include the ground clearance and whether or not the vehicle has four wheel drive.
- a member function `numSportsCars()` that returns the number of sports cars currently in inventory. You are not allowed to use a data member (or other variable) that keeps track of the number of sports cars added to the inventory.
- a member function `numSUVs()` that returns the number of sport-utility vehicles currently in inventory. Again, you are not allowed to use a data member (or other variable) that keeps track of the number of sport-utility vehicles added to the inventory.
- a member function `sellCar()` that takes a VIN as a parameter, and removes the vehicle with that VIN from the inventory. This member function should have return type `void`, and should not create any garbage.

You must also include a definition of `main()` that creates an instance of your `CarDealership` class, adds at least one vehicle of each type to the inventory, and then invokes each of the remaining member functions listed above.

The class `CarDealership` must be implemented such that cars are stored using an array of pointers to pointers (see the hints below for some more details on how to do this). In addition, the cars in the array must always stored in sorted order by VIN. Also, adding a car must take only linear time. That is, if there are currently n cars in the inventory, adding a single additional car is $O(n)$. Finally, note that your implementation must use only the elements of C++ covered in class. In particular, you may not use the Standard Template Library (STL).

For a small amount of extra credit, improve your implementation so that the time to add a single additional car is $O(\log n)$. If you attempt the extra credit, please submit two versions of your code. The first, $O(n)$ version must use an array of pointers to pointers as described above. The second, $O(\log n)$ version may use any data structures, including those in the STL for C++. In fact, you should definitely use the STL for the extra credit version, since writing your own data structure for this would be a lot of work.

Hints:

- define member functions `isSUV()` and `isSportsCar()` returning `bool` (false) in your car/vehicle class, and then override these member functions as appropriate in subclasses
- a heap dynamic array of pointers can be declared as a pointer to a pointer, as follows:

```
int **iptrArray;
```

and then allocated using `new` with the size. For example,

```
iptrArray = new int *[50];
```

creates an array of 50 pointers to `int`.

- you are allowed to use a data member that keeps track of the total number of all kinds of cars in the inventory
- values and variables of type `std::string` can be compared with the usual relational operators `<`, `==` and so on.

It's easiest to write the entire assignment in a single `.cpp` file. As described in class, real-world C++ projects would separate the code into multiple `.h` and `.cpp` files. If you wish to write and submit your project in that way, feel free to do so, but it will not affect your grade.

Your source file(s) must be submitted to Moodle before class time on the due date, as a single zip file. Your program must compile and run correctly under `g++` on the Tome Macs.

Your program will be graded on correctness, compliance with the above guidelines, use of encapsulation and coding style (including use of comments).