Homework #4
Assigned: Thurs., Nov. 14, 2024. Due Fri.., Nov. 22 at 11:59pm on Canvas

Programming: Follow proper coding standards as noted in class. *Submit a zipped folder with: a) a single PDF with all your programs and output copied and pasted in to the file, and b) your C++ program file.*

1. (50 points) Using the "CarRecords.txt" input file from Canvas, write an object-oriented C++ program in a single file called *CarRecords.cpp*. In the file, define two independent classes; class Car, class CarRecords, and the main function.

   Implement class Car with the following description:

Car	
private:	int year string color string make string model
public:	Car() setFields(string mk, string md, int yr,string cl) int getYear() string getColor() string getMake() string getModel()

Variables:
year, color, make, model– the four data fields from the file

Methods:
Car() – default constructor sets the data fields to their default values.
setFields(int yr, string cl, string mk, string md**)** – sets the class member fields to the values passed.
int getYear() – returns year
string getColor() – returns color
string getMake() – returns make
string getModel() – returns model

Implement class CarRecords with the following description:

CarRecords	
private:	int arraySize // keep track of number of records ifsteam infile Car *cars

public:	CarRecords(int size) // Reads file records to array
	~CarRecords()
	void printCarRecords()
	void sort_cars_by_make()
	void sort_cars_by_year()
	void print_duplicates()

Variables:

arraySize – size of the array set to the number of records read from the file
infile – input file stream used to read information from a file
cars – a pointer object (to a dynamically allocated array of objects)

Methods:

CarRecords(int size) – constructor sets the value passed to the arraySize member, creates a dynamic array enough to hold arraySize objects of type Car. In addition, the constructor uses the infile file stream and the setFields() method to initialize all the cars array elements with the car records read from the file.
~CarRecords() – Destructor frees the memory allocated with new, and closes the file handler.
void printCarRecords() – prints out the car records from the array of objects (see sample output)
void sort_cars_by_make() – sorts the records in ascending order based on the **make** field.
void sort_cars_by_year() – sorts the records in descending order based on the **year** field.
void print_duplicates() – identifies any repeated records, and prints them out when found. Repeated records means that all the fields are the same.

Test your program with the main function below. If the user enters a value equal to or less than 10 for records to read, the program should create an arrays with that many elements using dynamically memory allocation and should only read that many records from the file, e.g. if the user enters 5, the program should create an array of 5 elements and only read 5 car records. If the user enters a number greater than 10, the program should create an array of only 10 elements and read all 10 records from the file. This checking is done in the CarRecords(int size) constructor when it reads the file.

Note:

- For modularity, or if needed, your classes and the main function can have other helper functions. Your code should be well commented.
- Create your own getInt() function that validates and returns only if the correct int data type is entered. Refer to Homework-1. The getInt() function should not be part of any class.

```
int main() {
        // Ask the user for the number of records to read
        cout << "Number of Records to read? " ;
        int numRecs = getInt(); // getInt() returns in integer

        CarRecords *cr = new CarRecords(numRecs);

        // Print car records
        cr->printCarRecords();
        // Sort by Year
        cr->sort_cars_by_year();
        // Print car records
        cr->printCarRecords();
        // Sort by Make
        cr->sort_cars_by_make();
        // Print car records
        cr->printCarRecords();
        // Check for Duplicates
        cr->print_duplicates();

        delete cr;
} // end main
```