EECE 2160 - Embedded Design: Enabling Robotics

Spring 2017 – Homework #1
Due: Tuesday, January 31st (11:59pm)

Note: *Submit, in a zipped folder, all individual files, and a single PDF file with all the C++ programs, a sample output from each, and the input file. Your code should be well commented.*

1. (20 points) Write a C++ program on COE Linux that prompts the user for two positive integers. Provide individual program functions, called from `main()`, that do the following:
    a. Prints out the number of bytes used to store the following data types: **bool**, **char**, **int**, **float**, and **double**.
    b. Using the math library, returns the first number raised to the **power** of the second.
    c. Using the algorithm library, returns the **maximum** of the two numbers.
    d. Prints out the two numbers in **decimal**, **hexadecimal**, **octal**, and **binary** formats.

2. (20 points) Write a program that finds the transpose of a 3 x 3 matrix of integers using:
    a) A function that uses array indices, and
    b) A second function using only pointers

    Your program should have the following functions: `main()`, `void printMatrix()` called from `main` to print the matrix whenever necessary, `void indexTranspose()`, and `void pointerTranspose()`. The `main` function should declare and read the matrix elements from the keyboard or from a text file. The matrix should not be global! The main function should print out the matrix using `printMatrix()` function before and after transposing with either functions. Your functions should receive the matrix as arguments and can include any other necessary arguments, and can modify the original matrix. Submit your **commented** code and output of both transpose functions.

3. (30 points) Download and use the included "CarRecords.txt" file from Blackboard as your input file. The file has 10 random used car records. Each record has 4 fields with the following format: car make, car model, year, color e.g.

```
Toyota, Matrix, 2006, silver
Volvo, XC70, 2009, blue
              .
              .
             etc
```

```
MENU - Select an option:

1. Print the cars array
2. Insert car records into a sorted array
3. Sort cars by year
4. Print duplicates
5. Exit
```

Implement the tasks below for an **array of structs** of car records in a C++ program. Create a struct with the 4 fields described above. Create an interactive main program that asks the user to select an element from a menu (above) containing a set of actions associated with the tasks below. Your array should be created in the main function and passed to other functions as an argument. It should not be global!

a.) Provide a function `insert_array( )` that reads and stores the 10 records from file into an a **array of structs**.

b.) Provide a function `print_cars_array( )` that prints out the car records in the array list.

c.) Provide a function `sort_cars_by_year( )` that sorts the records in ascending order based on the **year** field.

d.) Provide a function `print_duplicates( )` that identifies any repeated records, and prints them out when found. Repeated records means that all the fields are the same.

Note: The above functions can have arguments and return types as you see fit.

4. (30 points) In a new C++ program, and using the same "CarRecords.txt" input file, implement the tasks below for a **linked list** of car records. Create a linked list node with the car fields and a pointer. Provide a similar menu as the one used above with the necessary modifications to reflect the tasks below. Do not use arrays at any point in this program. Your linked list should be created in the main function and passed to other functions. It should not be global!

   a.) Provide a function `insert_linkedList( )` that reads the 10 records from the file into a **linked list**.

   b.) Provide a function `print_cars_list( )` that prints out the car records in the linked list.

   c.) Provide a function `sort_cars_by_color( )` that sorts the records in ascending order based on the **color** field. The sorting should be done by **pointer manipulation**.

   d.) Provide a function `print_duplicates( )` that identifies any repeated records, and prints them out when found. Repeated records means that all the fields are the same.

Note: The above functions can have arguments and return types as you see fit.