**CARS (contributed by F. R. Salvador)**

**1. INTRODUCTION**

The table below shows survey results of new car owners in the USA. The 1st column corresponds to the Brand, the 2nd column is for the Quality Rating (QR) and the 3rd is the Customer Satisfaction Rating (CSR). The QR value is the number of defects per 100 cars (so lower numbers indicate higher quality). *Thus, we can say that KIA (with 126 defects) has a higher quality compared with JAGUAR (with 130 defects).* The CSR is a score between 0 to 1000 with higher values indicating greater satisfaction. *Thus, we can say that KIA (with a score of 761) has a lower Customer Satisfaction Rating compared with JAGUAR (with a score of 854).*

**Car Brands with their Corresponding Quality Rating (QR) and Customer Satisfaction Rating (CSR)**

| BRAND | QR | CSR |
|---|---|---|
| AUDI | 111 | 832 |
| BMW | 113 | 845 |
| CHEVROLET | 111 | 789 |
| CHRYSLER | 122 | 748 |
| DODGE | 130 | 751 |
| FORD | 93 | 794 |
| HONDA | 95 | 766 |
| HYUNDAI | 102 | 760 |
| JAGUAR | 130 | 854 |
| KIA | 126 | 761 |
| LEXUS | 88 | 822 |
| LINCOLN | 106 | 820 |
| MAZDA | 114 | 774 |
| MERCEDES-BENZ | 87 | 842 |
| MINI-COOPER | 133 | 815 |
| NISSAN | 111 | 763 |
| PORSCHE | 83 | 877 |
| SUZUKI | 122 | 750 |
| TOYOTA | 117 | 745 |
| VOLKSWAGON | 135 | 797 |

*/* Source: USA Today, June 16 and July 17, 2010 as cited on page 124 in the book by Peck, R., Olsen, C. & Devore, J. (2016). "Introduction to Statistics & Data Analysis", 5th Edition. CENGAGE Learning. */*

**2. SKELETON FILE AND OTHER FILES**

a. **`CARS-LASTNAME.c`** - this is the skeleton file that you'll need to edit as base code. Make sure to read and understand the contents and instructions in the skeleton file. Don't forget to rename this file with your own last name. For example, if your last name is SANTOS, then the file should be renamed as CARS-SANTOS.c.

b. **`main`**.c - this is the file that contains the main() function. Edit Line 18: #include "CARS-LASTNAME.c" to replace LASTNAME with your own file.

c. **`cars.h`** - this is a header file that contains the structure type declaration for the car data and function prototypes.

d. **`CARS-RATING.TXT`** - this text file contains the car data encoded in three columns as explained and visualized above. Open it using any text editor (for example, Notepad) and study its contents. Use it to test your solution; see Section 4. Feel free to edit the file contents to accommodate your own tests.

**e.** **`EXPECTED.TXT`** - this text file contains expected output of a logically correct solution to the problem requirements.

**3. REQUIRED TASKS**

Edit **`CARS-LASTNAME.c`** and encode your solutions to the five tasks described below.

**Task #1:** Define the **`Print_Cars_List()`** function which will **`printf()`** the values of all data in the list of car structures. Each line of output should display the brand, the QR value and the CSR value separated by at least one space. **DO NOT** print any extraneous or unnecessary character or string.

Example of Expected Output:

```
AUDI  111   832
BMW   113   845
   :
   :   // colon means other data values which are not shown here due to limited space
   :
VOLKSWAGON  135  797
```

**Task #2:** Define **`Append_Cars_List()`** function that will append new data in the list of car structures. Append means to add new elements (car data) at the end of the list. Assume that there is memory space still available, and that the new data, i.e., car brand, is different from those already stored in the list.

Example: Appending new data for example: **MITSUBISHI 146 767** into the list, and then printing the updated list will result into:

```
AUDI  111   832
BMW   113   845
    :
    :   // colon means other data values which are not shown here due to limited space
    :
VOLKSWAGON  135  797
MITSUBISHI  146  767
```

**Task #3:** Define a C function that will sort the list of structures by brand (name) in alphabetical order. You are required to use **selection sort algorithm**.

*Note: the original data set in the table above are in alphabetical order. However, since we appended new data with brand MITSUBISHI, the list of structures is therefore no longer sorted. Thus, there is a need for a sorting function.*

**Task #4.** Define a C function that will compute the answer to the following question:

***Q:** Is there a **<param_brand>** in the list of cars?*

**For this task, it is assumed that the array is already sorted alphabetically by brand name. There is NO need to call the sorting function inside the function definition.**

The function should perform a **BINARY SEARCH** to determine if **<param_brand>** is in the list of structures or not. For simplicity, assume that all the letters in search string **<param_brand>** are in upper case (just like in the Table above). If it is found, the function should return the **index** corresponding to where it was found in the array; otherwise, it should return -1.

Example #1. Q: *Is there a FORD in the list of cars?*
A: 5 /* a matching FORD brand is in the list, found in index 5 */

Example #2: Q: *Is there a DYIPNI in the list of cars?*
A: -1 /* DYIPNI brand is not in the list */

**Task #5.** Define a C function that will compute the answer to the following question (query):

***Q:** Which brands have better Customer Satisfaction Rating compared with **<param_brand>**?*

The function should build a NEW list of car structures (i.e., an array) where all structure data have a higher CSR value compared with that of **<param_brand>**. Moreover, the function should return the number of brands (an integer) value that satisfied the query.

Example: *Which brands have better Customer Satisfaction Rating compared with **MINI-COOPER**?*

The new list of structures will contain the following data. NOTE: the function should NOT print the data!!!

```
ACURA  86  822
AUDI  111  832
BMW  113  845
JAGUAR  130  854
LEXUS  88  822
LINCOLN  106  820
MERCEDES-BENZ  87  842
PORSCHE  83  877
```

The function returns 8 (which means that there 8 brands have better CSR compared with that of MINI-COOPER.

**4. HOW TO COMPILE, RUN AND TEST YOUR PROGRAM**

Compile your C program, either inside the IDE or in the command line interface. Make sure that there are no syntax/compilation errors. Let's assume that the source file is named as **`CARS-SANTOS.c`**, and that the executable file is named as **`CARS-SANTOS.exe`**.

Run the exe file in the command line **with I/O redirection** as shown in the example below.

```
C:\CCPROG2> CARS-SANTOS < CARS-RATING.TXT > OUTPUT-SANTOS.TXT
```

The input redirection will enable data to be read via **`scanf()`** from the text file. Make sure that **`CARS-RATING.TXT`** is in the same folder as your C source file. The program should produce the desired output following the format described in Section 3.

The accompanying **`EXPECTED.TXT`** file contains the expected output based on the original values in **`CARS-RATING.TXT`** and new data as appended in the **`main()`** function.

**5. SUBMIT YOUR FILES VIA CANVAS**

Submit/upload two files via Canvas before the Canvas deadline:

a. **`CARS-LASTNAME.c`** -- your C source file solution. Don't forget to rename your file with your own last name.
b. **`OUTPUT-LASTNAME.txt`** -- your program's output as described in the I/O redirection example above.

Back-up your solution (files) by sending it as an email attachment to your DLSU email account. Do not delete that email until you have completed CCPROG2.

**6. TESTING & SCORING:**

- Your program will be black box tested with a different set of test data, and/or different main() function.
- Each correct function definition will be given **10 points** each. Thus, the maximum total score will be 50/50.
- A program that has a syntax/compilation error will be given a score of 0 out 50.
- The score for an incorrect implementation of a required function is 0. For example, if the only correct solution is for Tasks 1 and 2, then the score will be 20/50.

**-- The End --**