Concordia University
Department of Computer Science and Software Engineering
COMP 348: Principles of Programming Languages
Fall 2018
Assignment 3

Evaluation: 100 points
(5% of your final grade)
Due date & time: Monday November 26th,2018 at 23:59

I. Object Oriented Programming with Ruby:

1. (10 pts) Write a Ruby method that takes an array of strings as an argument, sorts it alphabetically, iterates and uses a code block to display each string element along with its characters count. For example, if passing the following argument array ["Adam","Eve","Mark","Franklin","John"], then the output should be:

```
Adam, ch_count= 4

Eve, ch_count= 3

Franklin, ch_count= 8

John, ch_count= 4

Mark, ch_count= 4
```

2. (20 pts) Automated Readability Index (ARI) is used for testing readability in English text. The mathematical formula for calculating ARI in a text document is as follows:

$$ARI = 4.71 \times (characters/word) + 0.5 \times (words/sentence) - 21.43$$

Where:

$$characters = \sum letters, numbers\ and\ punctuation$$

$$words = \sum spaces$$

$$sentences = \sum full\ stops$$

The following table shows the educational grade level that corresponds to each ARI score:

Score	Age	Grade Level
1	5-6	Kindergarten
2	6-7	First/Second Grade
3	7-9	Third Grade
4	9-10	Fourth Grade
5	10-11	Fifth Grade
6	11-12	Sixth Grade
7	12-13	Seventh Grade
8	13-14	Eighth Grade
9	14-15	Ninth Grade
10	15-16	Tenth Grade
11	16-17	Eleventh Grade
12	17-18	Twelfth grade
13	18-24	College student
14	24+	Professor

Write a Ruby method that can read any document file, count number of characters, words, and sentences and then apply the ARI formula to find out the grade level required for a person to read the opened document. For example, for the following "paragraph.txt" file:

> *After the Lord Stanley of Preston was appointed by Queen Victoria as Governor General of Canada on June 11, 1888, he and his family became highly enthusiastic about ice hockey. Stanley was first exposed to the game at Montreal's 1889 Winter Carnival, where he saw the Montreal Victorias play the Montreal Hockey Club. The Montreal Gazette reported that he "expressed his great delight with the game of hockey and the expertise of the players". During that time, organized ice hockey in Canada was still in its infancy and only Montreal and Ottawa had anything resembling leagues.*

A Ruby method call: calcARI("paragraph.txt") should output the following:

```
Total # of characters: 474

Total # of words: 96

Total # of sentences: 4

Automated Readability Index: 13.8

Grade level: 18-24 (college student)
```

3. (30 pts) In this question, you should create an auto showroom inventory catalogue using Ruby. Your program should read a car listings file which contains all available inventory details where each line contains the following 11 listing features:

> #km, type, transmission, stock#, Drivetrain, Status, Fuel Economy, car_maker, model, year, Trim,set_of_features

examples of listing lines as follows:

> 65101km,Sedan,Manual,18131A,FWD,Used,5.5L/100km,Toyota,camry,SE,{AC, Heated Seats, Heated Mirrors, Keyless Entry},2010
>
> coupe,1100km,auto,RWD, Mercedec,CLK,LX ,18FO724A,2017,{AC, Heated Seats, Heated Mirrors, Keyless Entry, Power seats},6L/100km,Used
>
> AWD,SUV,0km,auto,new,Honda,CRV,LE,19BF723A,8L/100km,2018,{Heated Seats, Heated Mirrors, Keyless Entry}

Note: order of listing features varies from one line to another as it was entered by employees not trained for proper data entry. Also, assume the following listing feature possible values (case insensitive):

Listing features	Values
#km	Kilometers followed by "km"
Type	{Sedan, coupe, hatchback, station, SUV}
Transmission	{Auto, manual, steptronic}
stock#	Combination of letters and numbers NOT ending with "km"
Drivetrain	{FWD,RWD,AWD}
Status	{Used, new}
Fuel Economy	Similar to: 5.5L/100km format
car_maker	{Honda, Toyota, Mercedes, BMW, Lexus}
Model	Any text that doesn't match any of the other criteria in this table
Year	Any year
Trim	any two letters acronym
set_of_features	Any set of features inside curly parenthesis

Your program should have four methods as follows:

- *convertListings2Catalougue: a method* that:
  - reads the listings file line by line
  - correctly recognizes and extracts different listing features
  - instantiates appropriate objects of different classes and subclasses that you should define to hold the listing information for different car makers, their models and trim lines available in our listings. Any parent class should have properties that count the number of listings of that maker or that

model along with other common behavior or properties. In the listings examples above, you can create an object of class (car_model) and store all the listing information in that class and also update the parent (car_maker) common class properties accordingly.

Note: The OOP design and choice of classes and subclasses is something that differs from one programmer to another and depends on the way you look at the problem.

- searchInventory: a search method that accepts a hash as its argument and based on the combination of hash key-value pairs will perform an advanced search for all vehicles in stock that matches the criteria. For example, searchInventory({“car_maker” => ”Mercedes”}) should display all Mercedes vehicles available.

- Add2Inventory: a method that accepts a new listing as a single line of unordered listing features, add the line to the original listing file and add an appropriate object to the catalogue based on the listing features.

- saveCatalogue2File: a method that traverses all created catalogue objects and stores them to an output file alphabetically according to their maker name. each listing features will follow a strict fixed order as opposed to the original file random order. For example, the original three listings will look as follows in the output file:

  Toyota,camry,SE,65101km,2010, Sedan, FWD, Manual, 18131A, Used,5.5L/100km, {AC, Heated Seats, Heated Mirrors, Keyless Entry}

  Mercedec,CLK,LX, 1100km,2017,coupe, RWD, auto, 18FO724A, Used,6L/100km,{AC, Heated Seats, Heated Mirrors, Keyless Entry, Power seats}

  Honda,CRV,LE,0km, 2018, SUV, AWD, auto, 19BF723A, new,8L/100km,{Heated Seats, Heated Mirrors, Keyless Entry}

II. Procedural programming with C:

4. (10 pts) write a C function matrixTranspose that takes a two-dimensional array as its input argument then transposes its elements. Develop it in two steps:

a. First, assume the matrix is a square matrix. This will make finding the transpose a simple swapping of the arrays’ elements.

b. Second, generalize your function to work with any NxM matrix where N≠M. You will need to use pointer arithmetic instead inside the function and also properly handle memory allocation for this purpose.

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}^T = \begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix}$$

5. (30 pts) Given that C is NOT OOP, so in this question you will try to mimic OOP state of auto inventory catalogue in Q3 using C structures. Using the ordered output file created using your Ruby program in Q3, write a C program that uses that output file and works according to the following description:

a. Creates different structure types that corresponds to different class types that you created in Q3

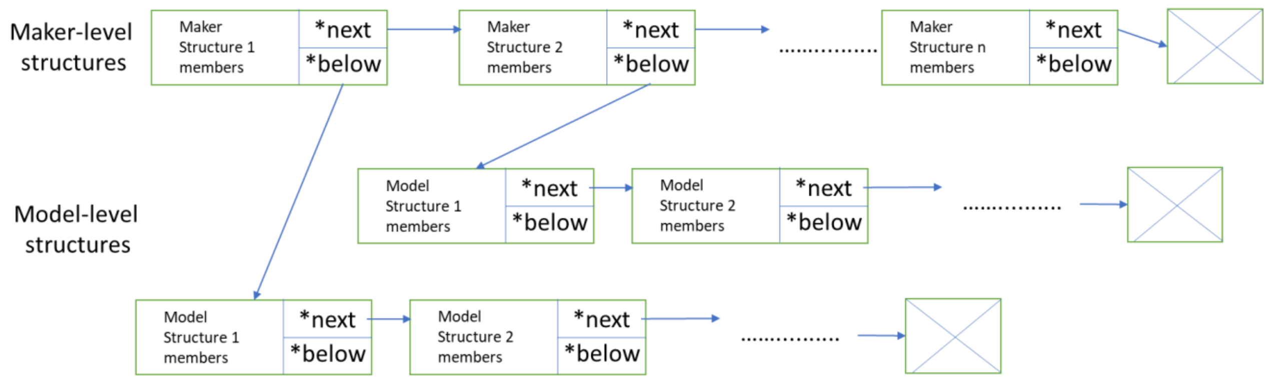
b. Each structure will contain a pointer to the next structure of the same type.

c. The top-level structure will additionally contain a pointer to the first structure in the level below that corresponds to one of the models of that car maker

d. It reads the file and starts populating variables of the corresponding structure type and updates the linked list pointers accordingly. It will also store different vehicle listing features in the structures that it just created.

e. You should also create the three functions explained in Q3: searchInventory, Add2Inventory, saveCatalogue2File. The implementation will be quite different as you will have to traverse the linked list (*multi linked list* in our case here) to find different listing and process them accordingly.

The multi linked list of structures should look like the figure below:



Maker-level structures
Maker Structure 1 members
*next
*below
Maker Structure 2 members
*next
*below
....
Maker Structure n members
*next
*below
Model-level structures
Model Structure 1 members
*next
*below
Model Structure 2 members
*next
*below
....
Model Structure 1 members
*next
*below
Model Structure 2 members
*next
*below
....