Faculty of Information
School of Graduate Studies
University of Toronto – St. George
Semester: Fall 2017
INF1340H – Programming for Information Systems

Assignment 3 – 17%
Due: Friday December 15th, 2017, before midnight

In this assignment, you have to write a program that manages car records and a car inventory. Car records are organized in a *list* containing car data. A car is a *list* containing (in this EXACT order): a model number (`str`), a year (`int`), a colour (`str`), a make (`str`), a model (`str`), a body type (`str`), and a quantity (`int`). Since a car is a list, we can say that the complete set of records is a list of lists. Car data are stored in a Comma Separated Values (CSV) file. The file has the following format:

```
#Comments in the file. To be ignored
Fog Lights,EK13Z,2003,Black,Chevrolet,Silverado,Truck,3
```

The first value is a car accessory (`str`). This value must be stored in a dictionary (inventory), in which the keys are the different accessories and the value is a *list* of car model numbers. This dictionary represents the full inventory of cars, organized by key/value pairs.

The other values in the file represent the seven (7) data elements for a car record. These must be loaded in a separate list of records (similar to assignment #2).

Your program will have two main data structures:

1- A dictionary containing key/value pairs of accessories and car model numbers
2- A list in which each element is a list of car data

The ***main*** portion of your program (after all functions definitions) must do the following:

1. Create an empty dictionary to start with
2. Create an empty car records list to start with
3. Load the data from the file into the dictionary and the car records list
4. Read a menu selection from the user (see example run posted on Blackboard)
5. Call the appropriate function based on the menu selection
6. Read another menu selection and repeat steps 4 and 5 until the user selects Quit
7. When the user selects Quit, print “Goodbye”

Your program must *define* and *use* the following functions:

`load_data(file, inventory, records):`
`(file open for reading, dict of {str: list of str}, list) -> None`
This function takes three arguments: a `file` representing the input file containing the car data, a `dict` representing the car inventory stored in a dictionary of key/value pairs of accessories and model numbers, and a `list` representing the car records (a `list` of `list`). It loads all the data from the input file into the dictionary (`inventory`) and the records list (`records`).

`menu(inventory_size):`
`(int) -> str`
This function takes one argument: the size of the car inventory as an `int`. The function returns a `str` representing the menu selection from the user. It displays the Car Inventory Menu and reads the menu selection from the user, which is returned as a `str`. Note that the function only displays menu options 1 and Quit if the car inventory is empty (i.e., when the car inventory is empty, the only options to the user are to add a car or quit). When the car inventory is not empty, the function displays the full menu.

`find_index(records, model_number):`
`(list, str) -> int`
This function takes two arguments: a `list` representing the car records (a `list` of `list`) and a `str` representing a car model number. The function searches the car records and returns the index of the car with a matching model number, as an `int`. The function returns `-1` if the model number is not found.

`add_car(inventory, records):`
`(dict of {str: list of str}, list) -> None`
This function takes two arguments: a `dict` representing the car inventory stored in a dictionary of key/value pairs of accessories and model numbers, and a `list` representing the car records (a `list` of `list`). The function adds the key/value pair (accessory and model number) to the inventory and the car to the list of records *if and only if* the car is not already part of the inventory. If the car is already part of the inventory, the function asks the user the quantity to be added and increases the current quantity accordingly. Use the `find_index` function to determine if a car record already exists or not.

`remove_car(inventory, records):`
`(dict of {str: list of str}, list) -> None`
This function takes two arguments: a `dict` representing the car inventory stored in a dictionary of key/value pairs of accessories and model numbers, and a `list` representing the car records (a `list` of `list`). The function removes a car from the inventory (i.e., removes the model number for a given accessory) and from the list of records *if and only if* the car quantity is one (1). If the car quantity is greater than one, it decreases the quantity of the car by one.

The function will also remove a key (accessory) from the inventory *if and only if* the list of values (model numbers) is empty.

Use the `find_index` function to determine if a car record already exists or not. If the accessory cannot be found in the inventory, the function prints the message: "No cars for accessory ' + accessory + '. Cannot remove car!". If the car cannot be found in the records, the function prints the message: "No cars with model number ' + model_number + ' for accessory ' + accessory + '. Cannot remove car!".

`find_car(inventory, records):`
`(dict of {str: list of str}, list) -> None`
This function takes two arguments: a `dict` representing the car inventory stored in a dictionary of key/value pairs of accessories and model numbers, and a `list` representing the car records (a `list` of `list`). The function searches for a car model number for a given accessory. If the car is part of the inventory, the function prints the car data, tab-delimited on one line and the car accessory on the next line. Use the `find_index` function to determine if a car record already exists or not. If the accessory cannot be found in the inventory, the function prints the message: "No car for accessory ' + accessory + '.". If the car cannot be found in the records, the function prints the message: "No cars with model number ' + model_number + ' for accessory ' + accessory + '. '".

`show_inventory(inventory, records):`
`(dict of {str: list of str}, list) -> None`
This function takes two arguments: a `dict` representing the car inventory stored in a dictionary of key/value pairs of accessories and model numbers, and a `list` representing the car records (a `list` of `list`). The function prints all the cars for every accessory, tab-delimited, one car per line. See sample execution on Blackboard for output format.

`output_inventory(file, inventory, records):`
`(file open for writing, dict of {str: list of str}, list) -> None`
This function takes three arguments: a `file` representing an output file, a `dict` representing the car inventory stored in a dictionary of key/value pairs of accessories and model numbers, and a `list` representing the car records (a `list` of `list`). The function outputs all the cars for every accessory, tab-delimited, one car per line to the output file. See output file example posted on Blackboard for output format. PS: The name of your output file should be "`output.txt`".

An example of complete execution ***testing all options*** as well as an example of the output file are available on Blackboard under "Assignment #3". Please look and walkthrough these examples VERY carefully. Make sure your program accounts for all possibilities shown in the example.

WHAT & HOW TO SUBMIT

This assignment can be completed in groups of 2 students or individually. The code for your assignment must be in a `.py` file and submitted electronically. Your `.py` file must include the names of all group members as *comments* at the top of the file (DO NOT include STUDENT NUMBERS). Submit your assignment through the UofT portal (Blackboard: Assignment #3). One submission per group.

Suggested header (first lines in `.py` file):

```
# Assignment #3
# INF1340 Section 1
# Fall-2017
# John Doe
# Jane Smiths
```

GRADING

Your assignment will be graded in accordance to the following criteria:

Definition, docstring, and body of `load_data` function:	10 marks
Definition, docstring, and body of `menu` function:	3 marks
Definition, docstring, and body of `find_index` function:	2 marks
Definition, docstring, and body of `add_car` function:	10 marks
Definition, docstring, and body of `remove_car` function:	10 marks
Definition, docstring, and body of `find_car` function:	10 marks
Definition, docstring, and body of `show_inventory` function:	10 marks
Definition, docstring, and body of `output_inventory` function:	15 marks

Code to create empty inventory dictionary, empty records list, open the input file, call `load_data` function, read menu selection in a loop, and call functions:	10 marks
Coding style and comments:	5 marks
Program produces the correct outputs (as per example posted on Blackboard):	15 marks