CT60A0203 – Introduction to Programming with Python

Project work: Vehicle Rent System: Part ONE

Due date and time: **2nd of November 2021 11.59 PM**

Submission: Submit your code file and supporting text files in the Moodle- project work page.

Course Plagiarism Statement: Plagiarism in oral, written, or visual presentations is the presentation of the work, idea, or creation of another person, without appropriate referencing, as though it's one's own. Plagiarism is not acceptable and may result in charges of academic misconduct which carries ZERO marks in the assessment task(s). It is also a disciplinary offence for students to allow their work to be plagiarized by another student.

A Car Rental company rent vehicles with a wide range of prices. The rental company rents two types of vehicles namely ordinary (O) and premium (P) for its customers with varied range of prices and discounts. The ordinary cars which are usually normal cars with basic facilities such as A/C, heater, and additional tires (including winter tires). The car rental company also supplies premium cars, which are usually luxury cars (will be dealt in part II). The premium cars are given a free mileage allowance for each day of rent. The free mileage allowance varies from one premium car to another. After renting a premium car, renters may also choose to rent additional accessories (such as built in mini-fridge, GPS navigator and more). For example, a BMW vehicle renter can rent GPS navigator and a car seat at additional 20.00 Euros for each day of rent. The customers can view the car details including status (A - available for rent / R - rented or not available).

The text file "**Vehicle.txt**" contain details of all vehicles that owned by rental company for rental or rented at present. The table below shows contents of text file as sample.

Vehicle ID	Description	Type	free_mileage	Daily rate	Status
XJY-604	Hyundai Getz	O	0	45.00	A
AJB-123	BMW mini	P	200	65.00	A
WYN-482	Ford Fiesta	O	0	40	A
BKV-943	Ford Ikon	P	150	60	A
JMB	Ford Flex	O	0	50	A
---	--	--	--	--	--
--	--	--	--	---	--

*Add at least 10 more vehicle information manually at "**Vehicle.txt**" before proceeding to **Part I.** But "*rentVehicles.txt*" and "*Transactions.txt*" file contents should be appended via code only. Read subsequent pages to proceed further.

Part I: You are required to write an interactive menu-based program to manage the renting of vehicles and related transactions. The requirements are specified below.

(i) Write a menu driven program that performs the common operations

Vehicle Menu

Display Available Cars	1
Rent Vehicle	2
Complete Rent	3
Exit	4

Option 1: Display available Cars

Write a procedure **displayCars()** that displays vehicle details that are available for rent. That is, print the details of all vehicles with status **'A'** from the text file "**Vehicle.txt"**. [Refer the table given above]

Option 2: Rent Vehicle

Prompt the user to enter the *Vehicle ID*. Then invoke the procedure **rentVehicle(VehicleID)** that takes Vehicle ID as parameter to check whether the vehicle is available for rent. Search the vehicle "**Vehicle.txt**". If no such vehicle exists or the vehicle is on rent, print an appropriate error message and return to main menu. [Hint: creating a procedure **mainMenu ()** to display menu options may be handy]

If the vehicle is available for rent then it should ask for *renter ID* (usually it is a social security number /national id), odometer reading at the time of renting the vehicle and display the details for renter to proceed further.

Car XJY-604 is rented to 250385-231C

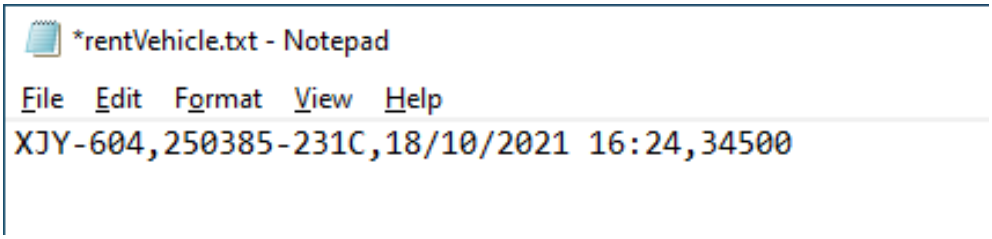
********** Vehicle Details **********

vehicle ID=XJY-604 Description= Hyundai Getz Daily Rate=45.0 Status=R

Renter ID=250385-231C Date/time of rent=18/10/2021 16:24 rent starting odometer=34500

Then the status of the vehicle that was rented should be updated as **"R"** at "**Vehicle.txt".** In addition, **rentVehicle()** should call another procedure **rentDetails()** which should perform the following operations.

rentDetails(): It opens a file **"rentVehicle.txt"** and append the rental details once the **rentVehicle()** execution is successfully completed. Each successful rent vehicle process must be written in a separate row. The details that should be written are, *Vehicle ID, Renter ID, date, and time the vehicle was rented* (should be current date you are not allowed enter manually), and *initial odometer* (int) of the vehicle. After successful completion of text file update, it should display the message "Renting vehicle is successful" then should return to main menu.



Option 3: Complete Rent

Selecting this option should prompts the user to enter *Vehicle ID* as input. Then invoke the procedure **RentComplete()** which takes Vehicle ID as parameter to check whether the vehicle is actually rented. Search status of the vehicle at "**Vehicle.txt**".

If no such vehicle exists or the vehicle is not on rent, print an appropriate error message and return to main menu.

If rented, then rental charges must be calculated based on number of days the vehicle was rented (return date which should be a current date – start date + 1). The status of the returned vehicle must be changed as **"A"** at **Vehicle.txt**. Then the results must be displayed as given below.

Car XJY-604 is returned from 250385-231C

********** Vehicle Details **********

vehicle ID=XJY-604 Description= Hyundai Getz Daily Rate=45.0

Renter ID=250385-231C Date/time of return =20/10/2021 12:14 rent starting odometer=34500 rent end odometer=35500 Rental charges=135.00 Euros

In addition, the afore displayed details must be appended into a file called **"Transactions.txt".** Once the appending is successful it should display the message "Car XJY-604 is returned" and return to main menu. The sample screenshot of the file is given here.

Option 4: Exit

Exit from the menu by displaying "Thanks for using Car Rental System. Bye! Bye!".

Additional requirements:

User input must be checked, and program should not be crashed due to invalid input given by the user. Wrong input should be taken care by displaying appropriate error messages and return to main menu. You can define your own sub programs for better coding style. You can also initially change the rent dates in the file to check your rent complete calculations.

Items	Check if completed	Possible points	Actual points
Main menu display – using procedure [for example Displaymenu()]		2	
Option 1: displayCars() – Available for rent only		4	
Option 2: Rent Vehicle *(i)* ask for input: Vehicle ID and calling *rentVehicle()* (ii) search not successful with message and return to main menu (iii) search is successful, renter ID input, change in the status of "*Vehicle.txt*" and display rent details (iv) Invoking *rentDetails()* and writing/appending rent information into a file "*rentVehicle.txt*" and return to main menu		2 3 6 4	
Option 3: Complete Rent (i) ask for input: Vehicle ID and calling *RentComplete()* (ii) search not successful with message and return to main menu (iii) search is successful, renter ID input, change in the status of "*Vehicle.txt*" (iv) Computation of rental charges + display rent complete details (v) writing rent completion into "*Transactions.txt*" and return to main menu		2 2 3 4 4	
Option 4: Exit		2	
(i) Input check – displaying appropriate message and return to main menu if the input is mistyped. (ii) used simplified coding: for example, defined some more subprograms for file handling to avoid repetition.		3 4	
Total		45/100	

Part II will be released in Week 9-10 that carries 55 marks [updating current menu with some more options and calculations + Exception handling + Visualizing the results in graphics form (self-study)