MQF Object Oriented Programming I Fall 2019

Hw2 Due 10/1/2019 before midnight

Specifications

Rutgers parking garage management system is required to take care Rutgers University Paring needs. The system can keep track of all the cars parked in your garage. For security reason, your system shall be able to tell the maker, model, color, and license plate number of the cars parked on this garage. The time a car parked in the garage is a normal distribution with mean at five hours and standard deviation of 2 hours. Assume this parking garage charges a $4.00 minimum fee to park for up to three hours. Your garage charges an additional $1.00 per hour for each hour *or part thereof* in excess of three hours. The maximum charge for any given 24-hour period is $10.00. For example, if a car entered the garage at 10th hour 30th minute, and exited 14th hour 10th minute then the car was parked 3 hours and 40 minutes and can be treated as parked for 4 hours. So it will be charged $5.00 (first three hours $4.00 and plus $1.00 extra hour). If it exited at 20th hour then it parked 9 hours 30 minutes. So it will be treated as 10 hours. The charge will be $4.00 + $1.00 x (10-3) which is $11.00. Because the maximum charge is capped at $10.00, so it will be charged $10.00. If it exited at 12th hour then it would be counted to park for 1 hour and 30 minutes so the minimum charge rule will be applied and will be charged for $4.00.

Your program shall output how much to charge a car each time a car exit the garage. In order for you to facilitate the simulation of time progress, your screen will have an option to allow you to advance time of thirty (3) minutes. You may assume your garage opens at 6am and closes at midnight. Note you can only forward time but not backward. A car enters the garage will use the current time to associate with its ticket. Similar a car exited will use the difference between current time and the time on its ticket to calculate its charge.

At the end of your simulation, your system will output all information about cars as the following

```
Time-Entered   Maker  Model  Color  License-Number Time-Exited Total-Charge
```

At the end of all these listings, the system will output how much earned by summing up all charges.

Requirements

You have to write a **class** call **Car** which contains all the information you need in order to implement the system. The class **Car** contain at least one constructor, more constructors are allowed if you decide to do so. It also contains getters, and setters if you think it is needed, so that you can retrieve (get) and store (set) information such as maker, model, color, license plate number, and ticket into your system.

Create another **class** call **Ticket** which represents a ticket of a the car object. A ticket object will be associated with a car object when a car enters your garage. A ticket can be described by *hour* and *minute* which records the hour and minute when a car enters your garage of a particular day. Since that we assume the parking time of a car is normal distribution, you also need to store this information in the car object or in the ticket object. The time it will stay in the garage will be predetermined using C++11 normal distribution as mentioned in class. Your TA may give you the *seed* to generate random numbers when give test cases. You may use time(0) as seed when you run your planned test cases.

Write a main program call **MyParkingGarage**.cpp. It shall create a data structure (for example, using vector <Car>) which will be used to maintain the information of all the cars in and out of this parking garage.

The system will output a menu screen which allow user to do the following

- Printout all cars' information in the garage
- Allow a car to enter (add a car)
- Advance time (increment time by 30 minutes)
- Search for a particular car(s) in the garage. For example, find all Honda cars, find all silver color cars, fins a car with license plate '123ABC' etc.

Each time when time advance 30 minutes, your system will output all cars and their corresponding information that exit form your garage.

To print information of all cars currently parked in your garage, you will print one car on one row format as described earlier in this handout. Align the output for each column so that it will look neat.

If you use *vector* to implement this system. When a car entered your garage, you can use *puch_back ()* member function of *vector* to add it into the end of vector.

When a car exited your garage, you may use *erase()* member function of *vector* to delete that car object from the vector. (Either use *erase()* or not is depends on your design.)

To simplify the query of any particular car(s) is(are) on your system, you can assume user can only choose one criterial. For example, user can choose one and only one of maker, model, color, or license number. It means that user cannot use logical operator 'AND', 'OR' to form query. So user cannot ask the system to output cars which are 'maker = Honda' AND color = silver' cars. Or ask the system to output 'maker = Ford OR color = red'.

If user select ***maker*** then your system will ask the user to enter a string which represent a maker. If user entered *Honda*, your system will then search the system and output car(s) that match the search criterial. If user select License plate number as search criteria, then your system will ask user to enter license number and output that car's information if it exists in the system, otherwise it will simply output a message indicate there is no such car with that license number. Other searches are similar.

For example, if there are three cars in your system

```
Honda Accord Silver abc123
Toyota Camry Red dfg234
Honda Civic Red aaa123
```

If user wants to find all Silver color car(s) then you system will output

```
        Honda Accord Silver abc123
```
If user wants to find all Red color car(s) then you system will output
```
        Toyota Camry Red dfg234
        Honda Civic Red aaa123
```
If user wants to find how many Honda car(s), your system will output
```
        Honda Accord Silver abc123
        Honda Civic Red aaa123
```
If user wants to check if any car with license plate number dfg234, your system will output
```
        Toyota Camry Red dfg234
```

Testing

To make the simulation of this garage management system easier, you may assume all inputs are legal and valid. So no error input testing required.

You shall come up with your own “testing plan” when you develop this system, Your TA will provide some test cases later.

Grading rubrics

(1) Programming style	4
(2) Use of classes	6
(3) Appropriate testing	3
(4) Correct results	12

Submission:

(1) Run your program and record the output by copy the screen shots
(2) Put all your screen shots into a document and convert it into a pdf file
(3) Put all your source code file or files (.cpp file, .h) into a folder. Use *yourLastName-FirstName-Hw2* as the name of the folder. E.g. if your name is Kathy Smith then *Smith-Kathy-hw2* is the folder name
(4) Put the test run screen shot files into the folder above
(5) Zip the folder and Use *yourLastName-FirstName-Hw2* as the zip file name.
(6) Submit the file to blackboard.

Extra credit 3 points

Allow user to query with AND and OR. For example Honda car with Yellow color.

References of vector

Chapter 7 of our textbook

Example of *push_back* function of vector

https://en.cppreference.com/w/cpp/container/vector

Example of *erase* function

https://stackoverflow.com/questions/875103/how-do-i-erase-an-element-from-stdvector-by-index

http://www.enseignement.polytechnique.fr/informatique/INF478/docs/Cpp/en/cpp/container/vector/erase.html

Example of vector *size* function

http://www.cplusplus.com/reference/vector/vector/size/