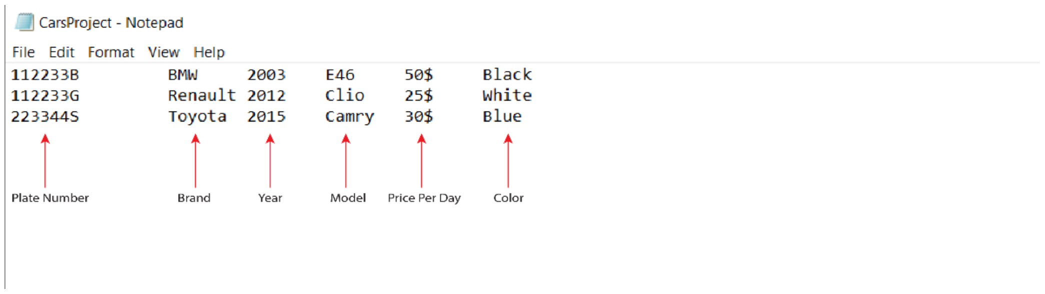
Description

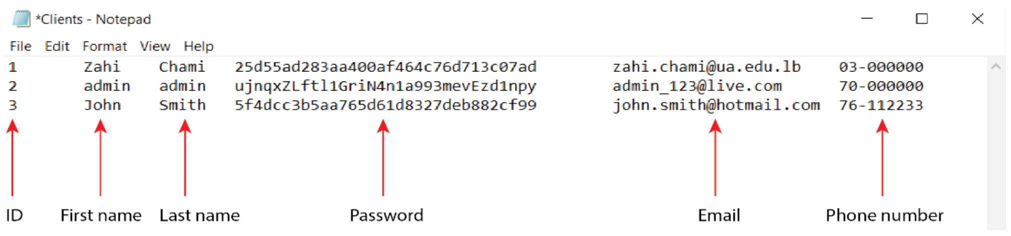
The goal of this project is to create a program in C++ language that represents the functionalities of a « car rental » application. This project consists of developing a tool to help the users to rent cars through the use of the structures and files. The cars and their characteristics are stored in a « **cars.txt** » file (Figure below).



```
112233B         BMW     2003    E46     50$     Black
112233G         Renault 2012    Clio    25$     White
223344S         Toyota  2015    Camry   30$     Blue
```

Figure 1 – cars file

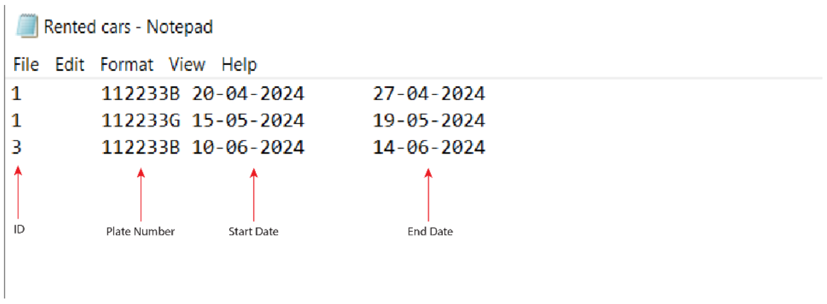
In addition, the users' informations are stored in a « **users.txt** » file, as shown in Figure 2. This file contains: the client ID, first name, last name, password (hashed[1]), email, and phone.



```
1       Zahi    Chami   25d55ad283aa400af464c76d713c07ad        zahi.chami@ua.edu.lb    03-000000
2       admin   admin   ujnqxZLftl1GriN4n1a993mevEzd1npy        admin_123@live.com      70-000000
3       John    Smith   5f4dcc3b5aa765d61d8327deb882cf99        john.smith@hotmail.com  76-112233
```

Figure 2 – users file

The application offers three main services: 1) rent a car, 2) cancel a car rental reservation, and 3) modify a rental car reservation. After each action, the selected car must be added/removed from the « **rental cars.txt** » (Figure 3) as well as from the « **client** » structure. At the end of the program, the rental cars must be saved in a PDF file containing the user's information, <u>sorted by the cars' prices in ascending order</u>.



```
1       112233B 20-04-2024      27-04-2024
1       112233G 15-05-2024      19-05-2024
3       112233B 10-06-2024      14-06-2024
```

Figure 3 – rental cars file

[1] **a hash function is a function that will calculate a unique signature from the data provided; in our case, the provided data represents the passwords to be hashed, and the result is shown in Figure 2 (Hashed Password field). Some hash functions: MD5, SHA1, SHA2, etc.**

The program workflow

The program must, at the beginning, ask the user if he has an account or not. If he is a new user, then he must create an account (step 1 below) and store the data in the « **user.txt** » file (Figure 2), then continue the following steps. Otherwise, he must be authenticated by entering his userID and the password before continuing to step 2. The user data must also be stored in the « **client** » structure in both cases.

1- Ask the user to enter the following information through the console:
- Last name, first name, and password (its length must be at least 8, while containing numbers, letters, and special characters)
- Email address (check and verify the address format)
- Phone number (verify the number format)

P.S: The identifier (userID) for each user must be unique and automatically generated by the program.

2- After completing the authentication process, the user could be either an administrator (he also has an account stored in the **user.txt** file) or a client.
   a. In the first case, he will then be responsible for the following tasks:
      i. Add cars to the « **cars.txt** » file by entering the data through the console. You must first check if the car exists in the file to avoid data redundancy.
      ii. Remove a car from the « **cars.txt »** file through the console.
      iii. Modify the data of a car that exists in the « **car.txt** » file.
   b. If the user is a client, he can then access the following options:
      i. Rent a car by choosing from a displayed list the plate number of the car and the rented date. The program should notify the client if there is an availability at this date for the chosen car. After finishing the rental process, the information must be stored in the « **car** » structure that exists in the « **client** » structure.
      ii. Cancel a rental car reservation through the console. The program must display the rented cars of the client to be able to choose the reservation that will be canceled based on the car plate number. The canceled reservation must be removed from the **car** structure that exists in the **client** structure.
      iii. Modify or change the rented car date. The program must check if there is an availability in the new one and if the entered date is valid (it should be greater than the current date).

3- All the existing and newly added/removed data to/from the structures must be written to the existing files. Note that the data should not be duplicated in the files.

4- Store the rented cars in a PDF file containing the clients' information, sorted by the cars' prices in ascending order.

Development

In this project, it is necessary to create:

- A **client** structure that contains: int usertID, string firstName, string lastName, string password, string phone, string email, int nbReservation, **car** *c.
- A **car** structure that contains: string plateNumber, string brand, string model, int year, string color, double pricePerDay, **date** d.
- A **date** structure: string startDate, string endDate.
- Files to read the data and write the result.
- Use functions, procedures, dynamic arrays, and pointers as much as possible

Deliverable

The source code and the input files

Evaluation Rubric

Criteria	Beginning - 1	Progressing - 2	Developing - 3	Exemplary - 4	Coef	Grade
Understands the project requirements	Student's work shows incomplete understanding of project requirements	Student's work shows slight understanding of project requirements	Student's work shows understanding of most requirements	Student's work shows complete understanding of all requirements	5	4
Uses appropriate algorithms by managing and allocating the memory dynamically	Student hacks out program with no thought to dynamic allocation	student chooses algorithms (for example: vectors) that are incorrect	Student chooses algorithms that are correct but somewhat inefficient (for example: the student allocates more space in the memory)	Student chooses or designs efficient algorithms by using the dynamic allocation concept correctly (without wasting the space memory)	15	4
Implements data structures along with their CRUD operations	no use of Data Structures	use of Data Structures but the CRUD operations are not showing	use of Data Structures while some of the CRUD operations are implemented	use of Data Structures and all CRUD operations are featured	20	4
Performs read and write operations to files	no use of Files	use of Files but the operations are not correct	use of Files while some of the operations are correct	use of Files and all the operations are correct	15	4
Tests Program for correctness	No evidence of any testing by student	Evidence of only one case tested	Evidence of a few cases tested	Evidence of all cases tested	25	4
Project explanation capabilities	Student reads all the code without explanation	Student occasionally reads the code with a partial explanation	Student does not read the code and have good explanation	Student does not read the code and have a very good explanation	10	4
Delivery	The code was more than 2 weeks overdue	The code was within 2 weeks of the due date	The program was delivered within a week of the due date	The program was delivered on time	10	4
				Final Grade	**100**	**100**

Bonus question

You can work on this project using **csv** files as a database (where the data are stored) instead of **txt** files. In addition, you can use the library « **ctime** » instead of creating the **date** structure.

Guidelines and information

1- The project must be worked individually or a group of two students.
2- The defense date will take place in the last week.
3- Beware of plagiarism!!!

Good Luck !!!