Practical 1 - Classes. Constant members.

Instructions

- Download the ***aeda2021_fp01.zip*** file from the moodle page and unzip it (contains the ***lib*** folder, the ***Tests*** folder with ***carpark.h***, ***carpark.cpp*** and ***tests.cpp*** files, and the ***CMakeLists*** and ***main.cpp*** files)
- In CLion, open a ***project***, selecting the folder containing the files from the previous point
- Do *"Load CMake Project"* on the *CMakeLists.txt* file
- Run the project (**Run**)
- Note that the ***six unit tests for this project are commented***. Remove comments as you implement the tests.
- You must perform the exercise respecting the order of the questions
- Perform the implementation in the ***carpark.cpp*** file.

Exercise

Implement a program for the management of a car park, which should manage information about customers and the parking of the respective vehicles. Implement the ***CarPark*** class according to the following. The class declaration must be made in the *CarPark.h* file and the definition of its function members in the *CarPark.cpp* file.

```
class InfoCard  {
public:
       string name;
       bool present;
 };
```

```
class CarPark  {
       unsigned freePlaces;
       const unsigned capacity;
        vector<InfoCartao> clients;
       const unsigned numMaxClients;
public:
       CarPark(unsigned cap, unsigned nMaxCli);
       bool addClient(const string & name);
       bool removeClient(const string & name);
       bool enter(const string & name)
       bool leave(const string & name);
       int clientPosition(const string & name) const;
       unsigned getNumPlaces() const;
       unsigned getNumMaxClients() const;
      unsigned getNumOccupiedPlaces() const;
      unsigned getNumClients() const;
};
```

a) Implement the constructor of the ***CarPark*** class, which accepts as parameters, the capacity of the park and the maximum number of clients with access to the park. Consider that the park initially has no clients and is empty. Also implement the following function members:

unsigned CarPark::getNumPlaces() const;

unsigned CarPark::getNumMaxClients() const;

These functions return the capacity of the park and the maximum number of customers, respectively.

b) Implement the following function members:

int CarPark::clientPosition(const string & name) const;

bool CarPark::addClient(const string & name);

These functions return, respectively:

- the index in the *clients* vector, of the client whose name is *name*, returning -1 if it does not exist
- success (true) or failure (false) in adding/registering a new client whose name is *name* to the car park. Consider that the client is initially out of the park.

c) Implement the function member:

bool CarPark::enter(const string & name);

This function records a customer's entry into the car park. Returns false if the customer cannot enter (he is not registered, his vehicle is already in the park, or the park is complete).

d) Implement the function member:

bool CarPark::removeClient(const string & name);

This function removes the registration of the client whose name is *name* of the car park. Removal is only possible if the client is currently out of the park

e) Implement the function member:

bool CarPark::leave(const string & name);

This function records the departure of the client whose name is *name* from the park. Returns *false* if the client cannot leave (he is not registered or his vehicle is not in the park).

f) Implement the following function members:

unsigned CarPark::getNumOccupiedPlaces() const;

unsigned CarPark::getNumClients() const;

These functions return the number of places occupied in the park and the number of registered clients, respectively.