Enhanced Parking Lot Simulation

Description

This assessment tests your ability to use the STL `stack` adapter, the STL `vector` container, and the STL `find` algorithm to solve a problem. You are asked to finish the implementation of a program that simulates a multiple-aisle parking lot. When cars are parked bumper-to-bumper, each aisle in this parking lot can hold three cars. There are five aisles in the parking lot.

It is your task to finish the implementation of the simulation that processes the vehicle arrivals and departures. The goal of the simulation is to keep track of and report how many times individual cars are moved while handling the departure of other cars. The simulation also displays an alphabetized list of all the cars that visited the parking lot during the simulation.

Files

Following is a list of files needed to complete this assessment.

- *ex2.rar* contains all of the following necessary files:
  - *Car.h* - This file declares class `Car`. You will not need to modify this file.
  - *Car.cpp* - This file implements class `Car`. You will not need to modify this file.
  - *main.cpp* - This file contains a partial implementation for the simulation. It is your task to finish this implementation.
  - *data.txt* - This file contains arrival and departure data.

Tasks

To complete this assessment, you will need to finish the implementation of the parking-lot simulation.

To begin, verify the files needed for this assessment.

1. **Extract** the archive to retrieve the files needed to complete this assessment.

Following is an ordered list of steps that serves as a guide to completing this assessment. Work and test incrementally. Save often

1. **First**, finish the implementation of function `find_car`. This function returns a reference to the `Car` object stored in the vector `cars` whose license plate equals the parameter `plate`. Use the STL `find` function to perform this task. To use this function correctly, you must supply it with three arguments. The first two arguments specify a range to search. The third argument is the value that the function attempts to find. This argument must be of type `Car`.
2. **Next**, finish the implementation of function `handle_arrival`. This function should iterate through the vector of stacks, looking for the first stack that does not contain three cars. If all five aisles (stacks) are full, output a message indicating such; otherwise place the license plate into the first non-full stack. This is essentially "parking" the car. For this arriving car, also add an entry of type `Car` to the vector `cars`. In this `Car` instance, make sure to record properly the index of the aisle where this car is parked.
3. **Then**, finish the implementation of function `handle_departure`. This function should locate the departing vehicle from the `cars` vector using function `find_car`. Then this function should remove the departing car's license plate from the appropriate aisle. Another stack must be used to move, temporarily, any cars that may be in front of the departing car. Record the number of times a car is moved when accommodating the departure of another car. For the departing car, display the number of times it was moved while it was parked in the lot.
4. **Finally**, write the code that displays an alphabetized list of all the cars the visited the list. To do this you must first sort the vector `cars` using the STL `sort` function. Then, using iterators, traverse the sorted vector and display the license plates of the cars.

Submission

Submit **only** the following.

1. *`main.cpp`* - Your completed parking-lot simulation program