Assignment 4
Problem-solving using OOP - *QuickBook*: A Simplified Train Seat Reservation System
Total Points: 200

Due: 10:00 p.m. of December 02, 2010 (Part 1)
10:00 p.m. of December 07, 2010 (Part 2)

In this assignment you are required to design a simplified ticket reservation system for railway trains. The solution will involve many of the *OO* concepts you have learnt in the course.

A *train* consists of *cars* of different types. A new car is added to a train according to some fixed rules which will be described later.

The seating plan for each type of car can be given as a two dimensional array. There are two types of cars.

1. [**Sleeping Car.**] As shown in Figure 1(a), a sleeping car contains four cabins and each cabin has three sleeping berths: the lower L, the middle M, and the upper U. Thus column number 0 corresponds to the lower berths, column number 1 corresponds to the middle berths, and the column number 2 corresponds to the upper berths in a car. Each berth in a sleeping car is identified by a seat number as shown in Figure 1(a).

2. [**Chair Car.**] As shown in Figure 1(b), a chair car contains seven rows and four columns of seats. The four seats in each row are designated as *WL* (window left in column 0), *ML* (middle left in column 1), *MR*

(middle right in column 2), and *WR* (window right in column 3). Each seat in a chair car is identified by a seat number as shown in Figure $1(b)$.

Note that a *seat* in a car can be identified in two different ways e.g., by its seat number or by the index pair (i, j) denoting a seat in row i and column j. For each car it is necessary to keep track of the number of unoccupied seats of each type. Recall that a sleeping car has three types of berths and a chair car has four types of seats. Furthermore, a car is identified with a car number which is assigned to it as a new car is added to the train. Thus for a train with n cars, the cars are numbered $0, 1, \ldots, n-1$.

Criteria for adding a new car to a train are as follows.

1. A new car (chair or sleeping) is added to the train only if the last car of the same type that was added does not have any more unoccupied seats/berths.

2. Whenever the requested seat/berth type cannot be booked (because all of them in the current car are sold out) for a passenger, the booking system should book the passenger a seat/berth type with smallest number of occupied seats/berths.

A train is a linked list of cars. A new train is created with two cars: a sleeping car and a chair car. A train always maintains a *current sleeping car* and a *current chair car* from which reservations are being made. A passenger supplies information on the type of car (sleeping or chair) and the type of seat or berth to make a reservation.

A train car can be implemented as a `C++` class called `Car`. Some of its data members are.

1. [`carNumber`.] Car's serial number;

2. [m.] Number of rows;

3. [n.] Number of columns;

4. [`carPlan`.] a Boolean two dimensional array of size $m \times n$ where a true value indicates that the place is not occupied and a false value indicates that the place is taken;

5. [`numVacant`.] an integer array (vector) of size n to keep track of the number of available places of each type or seats/berths.

	L [0]	M [1]	U [2]
[0]	0	1	2
[1]	3	4	5
[2]	6	7	8
[3]	9	10	11

(a)

	WL [0]	ML [1]	MR [2]	WR [3]
[0]	0	1	2	3
[1]	4	5	6	7
[2]	8	9	10	11
[3]	12	13	14	15
[4]	16	17	18	19
[5]	20	21	22	23
[6]	24	25	26	27

(b)

Figure 1: Diagram for train cars: (a) Sleeping Car (b) Chair Car; the number written in each cell denotes berth/seat number.

1. [Part 1] Define, implement, and test class `Car`. Write a test program that tests each member function. If necessary use additional data members such that the following functionalities are available in your implementation.

   (a) A constructor that takes four parameters (i) `carNumber`, (ii) number of rows, (iii) number of columns, and (iv) car type (sleeping/chair) and creates a `Car` object.

   (b) A member function that takes a row index i and a column index j and marks the seat/berth at location (i, j) as occupied and updates the `numVacant` data member.

   (c) A member function that returns true if there are unoccupied seats/berths of a given type and returns false otherwise.

   (d) A member function that returns true if the car is fully booked and returns false otherwise.

   (e) A member function that returns the type of largest number of unoccupied seats/berths in the car.

   (f) A member function that takes seat/berth type as parameter and returns the row number of the first unoccupied seat/berth of that

type.

(g) A member function that takes the row and column positions of a seat/berth as parameters and returns the corresponding seat/berth number.

(h) A member function that takes a seat/berth number as parameter and returns the corresponding row and column positions of a seat/berth.

(i) A member function that takes a type of seat/berth as parameters and books the first available seat/berth if possible and returns the seat/berth number; otherwise it returns -1.

(j) A member function that takes a seat/berth number as parameter and displays on computer screen an overview of the current booking status of the car where the seat/berth number passed as parameter is marked with the symbol `*`, an occupied seat/berth is marked with the symbol `+`, and an available seat/berth is marked with the symbol `-`. Given below is an example output from this function:

`C`hair Car Number 18
Seat Number: 8

	WL	ML	MR	WR
0	+	-	+	+
1	+	-	-	+
2	*	-	-	+
3	+	-	+	+
4	-	-	-	+
5	+	+	+	+
6	-	-	-	-

2. [**Part** 2(a) **(30 points)**] In this part of assignment you will implement two subclasses: `class SleepingCar` and `class ChairCar` derived from `class Car`. The main idea is to define common attributes and behaviors of train cars in the base/super class `class Car` and then define attributes and behaviors that are specific to each of sleeping cars and chair cars in their respective subclass while inheriting the common properties from `class Car`. Define and implement the classes `class SleepingCar` and `class ChairCar`. You should explain what

attributes (data members) of `class Car` are specific to the subclasses `class SleepingCar` and `class ChairCar`. Further, you must identify the member functions in the base class that should be made `virtual` or pure `virtual`. Recall that a constructor of the derived class must use an appropriate constructor of the base class to create the base portion of the derived class object. Implement `class SleepingCar` and `class ChairCar` with the following functionalities. You may use public inheritance.

(a) A constructor that takes an `int` value (car serial number) as a parameter and constructs an object of the respective subclass type.

(b) A member function that takes a column number in the car plan as parameter and returns a string representing the type of seat/berth corresponding to that column. For example, if the function is passed 1 as parameter and the car is a chair car, then the function returns the string `Seat: Middle Left`.

(c) A member function that takes no parameter and returns a string indicating the type of car (chair car or sleeping car) and the car serial number. For example, for an object of type sleeping car the string returned from the function may look like `Sleeping Car (Nr. 10)` indicating that the object is a sleeping car and its serial number is 10.

3. **[Part** 2(b) **(50 points)]** In this part of assignment you will design and implement `class RailwayTrain` which is a linked list of `Car` objects. You may use a simplified form of singly-linked list than the one discussed in lectures. More specifically, you need not implement the iterator class in the singly-linked list. Also, just define and implement only those member functions of the linked-list class that are necessary to implement `class RailwayTrain`.

   Recall from Part 1 of the assignment that there is a sleeping car and a chair car in a train from which bookings for seats/berths are being made. Also make sure that the booking rules as described in Part 1 of the assignment are followed in the implementation of `class RailwayTrain`. Define `class RailwayTrain` and implement the following functions.

   (a) A constructor that takes no parameter and constructs an object

of type `RailwayTrain` with the necessary initiliazation of the data members.

(b) Write a member function that displays an overview of the RailwayTrain object onto the computer screen. A sample output from the function may look like:

```
Train Overview.
Total Number of Chair Car: 10
Total Number of Sleeping Car: 3
```

(c) Write a member function that takes as parameter a car (pass-by-reference) and inserts the car in the railway train object and returns a reference to the newly inserted car.

(d) A member function to make a reservation. The function takes two parameters: a car (car currently being booked) and type of seat/berth (the column number) the user wants. The function will display on the computer screen the information about the requested seat/berth and the available seat/berth that can be booked.

(e) [**Part** 2(c)] **(20 points)** Write a test program to fully test the train booking system.