CS 201, Summer 2016
Homework Assignment 1

Due: 23:59, June 27, 2016

In this homework, you will implement a reservation system for a hotel chain. This hotel chain operates different hotels each of which has a unique name. Each hotel can have different number of floors and different number of rooms in each floor. The system should support making and cancelling reservations for different hotels.

The reservation system will have the following functionalities. The details of these functionalities are given below:

1. Add a hotel
2. Delete a hotel
3. Make a reservation
4. Cancel a reservation
5. Show the list of hotels
6. Show the list of rooms in a particular hotel
7. Show the list of reservations in a particular hotel

Add a hotel: The reservation system will allow the user to add a new hotel indicating its name, the number of floors, and a list of the number of rooms for each floor. Since the hotel names must be unique, the system should check whether or not the specified hotel already exists, and if it exists, it should not allow the operation and display a warning message. Note also that each floor may have different number of rooms. At the time of adding a new hotel, all rooms in this hotel become available.

Delete a hotel: The reservation system will allow the user to delete an existing hotel indicated by its name. If the hotel does not exist, the system should display a warning message. Note that this operation will also cancel all reservations for this hotel.

Make a reservation: The reservation system will allow the user to make a reservation for a particular room in a hotel. The user will specify the name of the hotel and the floor number and room number for the specific room requested. Both the floor numbers and the room numbers start from 1. A reservation is possible if the hotel exists, if the particular room exists, and if it is available (i.e., it was not reserved earlier). Otherwise, the system should not make any reservation and should display a warning message.

Cancel a reservation: The reservation system will allow the user to cancel a reservation for a particular room in a hotel. The user will specify the name of the hotel and the floor number and room number for the specific room to be cancelled. A reservation can be cancelled if the hotel exists, if the particular room exists, and if it was reserved earlier. Otherwise, the system should not cancel any reservation and should display a warning message.

Show the list of hotels: The reservation system will allow the user to see the names of the hotels in the system together with the number of floors and the total number of rooms in each hotel.

Show the list of rooms in a particular hotel: The reservation system will allow the user to see the list of all rooms in a selected hotel. If no such hotel exists, the system should display a warning message. If the hotel exists, the availability of each room should be displayed where "`o`" represents available rooms and "`x`" represents reserved rooms (see the example output below).

Show the list of reservations in a particular hotel: The reservation system will allow the user to see the list of all reserved rooms in a selected hotel. If no such hotel exists, the system should display a warning message.

Below is the required `public` part of the `HotelReservationSystem` class that you must write in this assignment. The name of the class must be `HotelReservationSystem`, and must include these public member functions. The interface for the class must be written in the file called `HotelReservationSystem.h` and its implementation must be written in the file called `HotelReservationSystem.cpp`. You can define additional public and private member functions and data members in this class. You can also define additional classes in your solution. You must add the interfaces for these classes to the `HotelReservationSystem.h` file, and must provide their implementations in the `HotelReservationSystem.cpp` file.

```
class HotelReservationSystem {

    public:
        HotelReservationSystem();
        ~HotelReservationSystem();

        void addHotel( const string name, const int numFloors, const int *numRooms );
        void deleteHotel( const string name );
        void makeReservation( const string name, const int floor, const int room );
        void cancelReservation( const string name, const int floor, const int room );

        void showHotels();
        void showRooms( const string name );
        void showReservations( const string name );
};
```

Here is a test program that uses this class. The corresponding output is also given below.

Example test code:

```
#include <iostream>
using namespace std;

#include "HotelReservationSystem.h"

int main() {

    HotelReservationSystem hrs;

    // hotel addition, deletion

    int roomsH1[4] = {5, 4, 4, 3};
    hrs.addHotel("H1", 4, roomsH1);

    int roomsH2[7] = {3, 5, 4, 6, 8, 5, 6};
    hrs.addHotel("H2", 7, roomsH2);

    int roomsH3[6] = {2, 3, 2, 5, 3, 3};
    hrs.addHotel("H3", 6, roomsH3);

    hrs.showHotels(); // H1, H2, H3

    hrs.deleteHotel("H3");
    hrs.showHotels(); // H1, H2

    hrs.deleteHotel("H4"); // warning: no such hotel

    int roomsH4[3] = {3, 2, 5};
    hrs.addHotel("H4", 3, roomsH4);
    hrs.addHotel("H2", 3, roomsH4); // warning: existing hotel with the same name

    hrs.showHotels(); // H1, H2, H4

    // room reservations

    hrs.showRooms("H2"); // ()

    hrs.makeReservation("H2", 2, 2);
    hrs.makeReservation("H2", 3, 5); // warning: no such room
    hrs.makeReservation("H2", 5, 1);
    hrs.makeReservation("H2", 6, 5);
    hrs.makeReservation("H5", 2, 3); // warning: no such hotel
    hrs.makeReservation("H2", 5, 1); // warning: room is already reserved

    hrs.showRooms("H2"); // (2,2), (5,1), (6,5)

    hrs.cancelReservation("H2", 2, 2);
    hrs.cancelReservation("H4", 4, 4); // warning: no such room
    hrs.makeReservation("H4", 1, 3);
    hrs.cancelReservation("H2", 2, 2); // warning: no such reservation
    hrs.makeReservation("H4", 3, 4);
    hrs.makeReservation("H2", 2, 4);

    hrs.showReservations("H2"); // (5,1), (6,5), (2,4)
    hrs.showReservations("H3"); // warning: no such hotel
    hrs.showReservations("H4"); // (1,3), (3,4)

    // back to hotel deletion and then some reservations

    hrs.deleteHotel("H4");

    hrs.makeReservation("H1", 2, 4);
    hrs.makeReservation("H1", 1, 5);
    hrs.cancelReservation("H2", 6, 5);

    hrs.showRooms("H1"); // (2,4),(1,5)
    hrs.showRooms("H2"); // (5,1),(2,4)

    int roomsH5[10] = {5, 4, 3, 2, 1, 1, 2, 3, 4, 5};
    hrs.addHotel("H5", 10, roomsH5);

    hrs.showHotels(); // H1, H2, H5

    return 0;
}
```

Output of the example test code:

```
Hotel H1 is added
Hotel H2 is added
Hotel H3 is added

Name   #Floors #Rooms
H1      4      16
H2      7      37
H3      6      18

Hotel H3 is deleted

Name   #Floors #Rooms
H1      4      16
H2      7      37

Hotel H4 does not exist for deletion
Hotel H4 is added
Hotel H2 already exists

Name   #Floors #Rooms
H1      4      16
H2      7      37
H4      3      10

Hotel H2 rooms:
1: ooo
2: ooooo
3: oooo
4: oooooo
5: oooooooo
6: ooooo
7: oooooo

Room 2 on floor 2 is reserved at hotel H2
Room 5 on floor 3 does not exist at hotel H2
Room 1 on floor 5 is reserved at hotel H2
Room 5 on floor 6 is reserved at hotel H2
Hotel H5 does not exist for reservation
Room 1 on floor 5 is not available at hotel H2

Hotel H2 rooms:
1: ooo
2: oxooo
3: oooo
4: oooooo
5: xooooooo
6: oooox
7: oooooo

Reservation for room 2 on floor 2 is cancelled at hotel H2
Room 4 on floor 4 does not exist at hotel H4
Room 3 on floor 1 is reserved at hotel H4
Reservation for room 2 on floor 2 does not exist at hotel H2
Room 4 on floor 3 is reserved at hotel H4
Room 4 on floor 2 is reserved at hotel H2

Hotel H2 reservations:
Floor  Room
2      4
5      1
6      5

Hotel H3 does not exist for showing reservations

Hotel H4 reservations:
Floor  Room
1      3
3      4

Hotel H4 is deleted
Room 4 on floor 2 is reserved at hotel H1
Room 5 on floor 1 is reserved at hotel H1
Reservation for room 5 on floor 6 is cancelled at hotel H2

Hotel H1 rooms:
1: oooox
2: ooox
3: oooo
4: ooo

Hotel H2 rooms:
1: ooo
2: oooxo
3: oooo
4: oooooo
5: xoooooooo
6: ooooo
7: oooooo

Name  #Floors #Rooms
H1     4      16
H2     7      37
H5     10     30
```

IMPORTANT NOTES:
Do not start your homework before reading these notes!!!
Output message for each operation should match the format shown in the output of the example code.

1. This assignment is due by 23:59 on Monday, June 27, 2016. You should upload your homework using the online submission form on the course home page before the deadline. No hardcopy submission is needed. The standard rules about late homework submissions apply. Please see the course home page for further discussion of the late homework policy as well as academic integrity.

2. You ARE NOT ALLOWED to modify the given parts of the header file. You MUST use dynamically allocated arrays in your implementation. You will get no points if you use fixed-sized arrays or data structures such as vector from the standard library. Similarly, you will get no points if you implement your homework using linked lists. Moreover, you ARE NOT ALLOWED to use any global variables.

3. However, if necessary, you may define additional public and private data members and member functions in your class. You can also define additional classes in your solution.

4. Your code must not have any memory leaks. You will lose points if you have memory leaks in your program even though the outputs of the operations are correct.

5. In this assignment, you must have separate interface and implementation files (i.e., separate `.h` and `.cpp` files) for your class. We will test your implementation by writing our own driver `.cpp` file which will include your header file. For this reason, your class' name MUST BE `HotelReservationSystem` and your files' name MUST BE `HotelReservationSystem.h` and `HotelReservationSystem.cpp`. Note that you may write additional class(es) in your solution. You should put these two files (and any additional files if you wrote additional classes in your solution) in a single archive file (e.g., zip, tar, rar). The submissions that do not obey these rules will not be graded.

6. We also recommend you to write your own driver file to test each of your functions. However, you MUST NOT submit this test code (we will use our own test code). In other words, do not submit a file that contains a function called `main`.

7. Make sure that each file that you submit (each and every file in the archive) contains your name, section, and student number at the top as comments.

8. You are free to write your programs in any environment (you may use either Linux or Windows). On the other hand, we will test your programs on "dijkstra.ug.bcc.bilkent.edu.tr" and we will expect your programs to compile and run on the dijkstra machine. If we could not get your program properly work on the dijkstra machine, you would lose a considerable amount of points. Thus, we recommend you to make sure that your program compiles and properly works on dijkstra.ug.bcc.bilkent.edu.tr before submitting your assignment.

9. This homework will be graded by your TA Nour Madhoun (nour.madhoun at gmail dot com). Thus, you may ask your homework related questions directly to her.