CCPS616 Lab 4 – Greedy Interval Scheduling

Preamble

In this lab you will write a C++ program that determines if your hotel with N rooms can accommodate a series of room requests. Your implementation must be from the ground up without linking any external libraries. Your lab should be completed in a single cpp file called **sched.cpp**, and should compile at the command line very simply as follows:

```
g++ -o lab4 sched.cpp lab4.cpp
```

Lab Description

In this week's lab you will implement a greedy algorithm for scheduling hotel bookings. You will write a function that accepts as input a series of hotel room requests and the number of rooms that are available. Your function will determine a scheduling that maximizes the number of bookings that are accommodated and return those bookings. At the pseudocode level, the input will include an integer representing the number of rooms and an array of integer pairs where each pair represents a start time and end time:

```
3    // Number of rooms
[ (1, 2), (3, 6), (5, 8), (6, 7), (6, 8) ]
```

The output of your function will be similar. It will be a 2D array of interval arrays, one array of intervals for each room. This will represent the scheduling that maximizes the number of intervals that are accommodated for a given number of rooms.

```
Room 1: [ (1, 2), (3, 6), (6, 7) ]
Room 2: [ (5, 8) ]
Room 3: [ (6, 8) ]
```

If the bookings cannot all be accommodated, your output will also include a list of those intervals that were not used in the solution. If the intervals in the above example were the same, but we only have two rooms, the output would look as follows:

```
Room 1:   [ (1, 2), (3, 6), (6, 7) ]
Room 2:   [ (5, 8) ]
No Room:  [ (6, 8) ]
```

A function signature with implementation details of the input and output data are provided in a template sched.cpp file (using C-style arrays and C++ STL containers). Like previous labs, you should build your scheduling function into this template.

Testing & Results

The provided lab4.cpp includes a main() function and several helper functions for generating randomized intervals. Additionally, you will test your scheduling function on a handful of small examples whose solutions are known. Several are provided in lab4.cpp, but you may add your own as well and use them to verify the correctness of your implementation.

Unlike previous labs, we won't bother doing any timing or efficiency analysis in this lab. We are simply concerned with producing the greedy result.

Submission

Submit your source file (**sched.cpp**) on D2L.

Labs may be submitted individually, or in groups of up to **two**. If you submit as a group, be sure to include both names in the D2L submission as well as your source code.