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 **lab4.cpp**, and should compile at the command line very simply as follows:

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

Please do not submit labs that don't even compile. We're past that. I'm not going to fix your syntax errors. If you can't make any progress on the lab whatsoever, submit a simple program that prints a message saying as much. For this, I will award you a single mark.

Lab Description

In this week's lab you will implement a greedy algorithm for scheduling hotel bookings. When executed from the command line, your program will accept one argument in the form of a file name. This file will contain data in the following format:

```
3
1 3 5 6 6
2 6 8 7 8
```

The first line of the file indicates the number of rooms your hotel has available. The second line indicates the start of each booking, and the third line indicates the end time of each booking. In the above example then, we have 3 rooms and 5 intervals: { (1, 2), (3, 6), (5, 8), (6, 7), (6, 8) }.

Your program should determine whether the N rooms are enough to accommodate all the requested bookings. Note that this is a slightly modified version of the interval scheduling problem we saw in class.

Testing

Test your program on several different inputs. Include test cases where N is sufficient to accommodate all bookings, but also where N is not sufficient. You need not synthesize any large-scale data sets. Simply hard-code a few files to test on. I.e., you may keep N in the single digits, and the number of intervals should be appropriate for the given vales of N. Try some of the sets we saw in class. Your command line execution should look as follows:

Linux: `./lab3 bookings.txt` **Windows:** `./lab3.exe bookings.txt`

Results

Your results should be written out to a file. For part marks, you may simply print True or False to stdout. For full marks, your file should indicate which intervals were scheduled in which room. A sample output for the previous example is shown below:

```
1 (1, 2) (3, 6) (6, 7)
2 (5, 8)
3 (6, 8)
```

If the bookings cannot all be accommodated, your output file should include an additional line without a room number that indicates which intervals were remaining after your algorithm was run. If our example from before had two rooms instead of three, the output might be:

```
1 (1, 2) (3, 6) (6, 7)
2 (5, 8)
(6, 8)
```

Submission

Lab 4 may be done in **pairs**! Submit your source file (**lab4.cpp**) to D2L. If you worked in pairs, Indicate the names of both partners in a comment at the top of your source code.

Marking Rubric: This lab is out of 5 marks

3 marks – Algorithm implementation correctness/completeness.
2 marks – Results produced correctly. One mark given for a simple True/False output, a full two marks given for printing which intervals are scheduled into which rooms.