COMP 482 Project 2: Earliest Starting Time on Interval Scheduling???

Due: 2355 Tuesday July 5, 2022

Points: 30 points possible

Overview: Recall the IntervalScheduling problem: you are given a set of intervals and asked to find the largest cardinality subset that has no overlapping intervals. We found that the EarliestStartingTime algorithm was not optimal by finding an instance where it did not get the best possible solution. Occassionally, you will want to test an algorithm on a problem to find an instance where it does not get the optimal solution.

Details: The input will come from a file called input.txt which will be placed in the same directory as your java file. The first line of the file will have a single integer value N which will be the number of intervals. The next N lines will be the intervals represented by s_i and f_i seperated by whitespace. Your program should output a single integer which is the number of non-overlapping intervals found by the EarliestFinishingTime algorithm. It is likely that the number of intervals found will not be the optimal solution. See the sample input below for examples.

You can discuss the algorithm to be used with anyone and consult any source (books, internet, etc). However, for this project, you are expected to write the code on your own with limited or no assistance from the professor, no assistance from others, and limited or no assistance from other sources (books, internet, etc). To clarify, you can seek assistance in understanding the task, but "your code" should be written by you: not written by others, not copied from others, not copied from books/internet.

Picky, but required specifications: Your project must:

- be submitted via canvas.
- consist of 1 or more dot-java files (no class files, zip files, input files or other files should be submitted). Each file must have your name and which project you are submitting as comments on the first 2 lines.
- not be placed into any package (for the java pedants, it must be in the default package).
- be designed and formatted reasonably (correct indentation, no excessively long lines, no excessively long methods, has useful method/variable names, etc)
- have one file called Project2.java.
- compile using the command 'javac Project2.java'.
- run using the command 'java Project2',
- accept input from a file called input.txt in the same directory as the java file(s) formatted precisely as described above.
- accomplishes the goal of the project. In other words, the output should be the correct answer, computed in a valid way, formatted correctly.
- be submitted on time (early and multiple times is fine).

For each listed item that you fail to follow, expect to lose at least 5 points. In particular, submitting via anything other than canvas will result in a 0 and submitting more that a week late will also result in a 0.

Sample execution: If input.txt contains

```
8
1 2
1 3
1 4
1 5
1 6
1 7
1 8
1 9
```

then the output should be

```
1
```

If input.txt contains

```
5
1 10
2 3
4 5
6 7
8 9
```

then the output should be

```
1
```

If input.txt contains

```
8
1 6
2 3
4 5
7 12
8 9
10 11
13 18
14 15
```

then the output should be

```
3
```

Stray Thoughts:
I suggest you finish you finish and submit your project at least a day or 2 days in advance. This way you have time and opportunity to ask any last questions and verify that what you upload satisfies the requirements. There is nothing wrong with working on the project for a day and uploading your code, working for another day and uploading your improved code, ..., working for another day and uploading your final version. In fact there are advantages: you have a fairly reliable place that keeps your versions and even if you get busy at the last moment you have still uploaded your best version. Do not be concerned that canvas will append a -1, -2, -3, ... to your submission.
While I am likely to test your project on the input above, I am likely to test your projects on many other inputs as well.
Your project should be written and understood by you. Helping or receiving help from others to figure out what is allowed/required is fine, but copying code is not. Significant shared source code indicates that you either did not write/understand what you submitted or you assisted another in submitting code they did

not write/understand.