## CSE305, Spring 2023 Assignment 3 Due Tue. Mar. 14, 11:59pm

**Reading:**

For Tuesday's lecture, read Sebesta Chapter 6 and also Chapter 7. Some parts of these chapters are new while other parts will give some pre-exam review. A reminder that the **First Prelim Exam** will be given in class period on **Thursday, March 16**.

—-**Assignment 3**, due **Tue. 3/14** "midnight stretchy" on *TopHat* and *CSE Autograder*—-

**(1)** The *TopHat portion*: This time it is a larger set totaling **40 pts.** Problems in the middle are best treated as if they were a programming assignment to try writing your own code, then check it against the alternative code bodies given. Scores from it will eventually be ported to *Autograder* and later everything to *UBLearns*, which is being used only as a gradebook.

The rest is to be submitted as **a single PDF file** via *CSE Autograder*, plus `submit_cse305 CSE305ps3NN.ml` with your answers to problem (2), where you substitute your initials for the NN. Problem (2) will thus be submitted both ways.

**(2)** (This is the exercise that was intended to be done in this week's recitations. We ask you to submit them both "on paper" in a (short!—handwritable too) PDF document and by online submission using `submit_cse305` on `timberlake`. Your file for the latter should be named `CSE305ps3NN.ml` where you substitute your initials for the NN. It should also have a `(** ...` "header comment" at the top with the filename, your name, and a brief usage note.)

All code used in the week six recitations, including some answers to problem 3 of HW2, is at https://cse.buffalo.edu/~regan/cse305/Week6RecsCode.txt

Consider lists `ell` of pairs $(a, b)$ of nonnegative floats where always $a < b$. Think of $a$ and $b$ as scheduled start and end times of an appointment or processor job or whatnot. Say that two scheduled items $(a_1, b_1)$ and $(a_2, b_2)$ **conflict** if $a_1 \leq a_2$ but $a_2 < b_1$, so that the latter starts before the former has ended.

(a) Write a function `flagConflicts` that determines whether `ell` has any conflicts. This time the function outputs a list of conflicts, not just giving a Boolean value, such that every conflicted appointment appears in at least one of the pairs in your list of pairs. You may *not* assume that `ell` is sorted by its first component $a$. (9 pts.)

(b) Rework your function to work with appointments (or job schedules or etc.) that mention also the day, according to the type

```
type appointment = Mon of float * float | Tue of float * float
| Wed of float * float | Thu of float * float | Fri of float * float;;
```

Updating your `lessThan` function from part (a) to handle the days may get tedious depending on how you match the days in the pairs of appointments, but longer code may even be more readable than clever shorter code that exploits the linear-order way the match cases are polled. (15 pts., making 24 on the problem and 64 pts. on the whole assignment)