Lab 07: Lists of structures

Create a separate file for each question. Keep them in your "Labs" folder, with the name `liiqj` for Lab ii, Question j.

Download the headers for each function from the file `labinterface07.rkt` linked off the "Labs" page on the course Web site.

After you have completed a question (except class exercises), including creating tests for it, you can obtain feedback by submitting it and requesting a public test. Follow the instructions given in the Style Guide.

This lab makes use of the following structure and data definitions:

(*define-struct event* (*type dur*))
;; An Event is a structure (make-event Sym Int), where
;; type is the type of event, and
;; dur is the duration of the event and is a positive integer in minutes.

(*define-struct card* (*value suit*))
;; A Card is a structure (make-card Nat Sym), where
;; value is the card value in the range 1 - 10 and
;; suit is the card suit in the set 'hearts, 'diamonds, 'spades, and 'clubs.

(*define-struct clock* (*hours mins*))
;; A Clock is a structure (make-clock Nat Nat), where
;; hour is the number of hours in the range 0 - 23 and
;; mins is the number of minutes in the range 0 - 59.

Language level: Beginning Student.

1. *[Class exercise with lab instructor assistance]* Create a function *total-dur* that consumes a list of *event*s, *eventlist*, and produces the total duration of all *event*s in *eventlist* in minutes.

2. Create a function *max-card* that consumes a nonempty list of *card*s, *loc*, and produces the *card* with the maximum value of any *card* in *loc*. If the maximum value is the value of more than one *card* in *loc*, produce the *card* closest to the end of *loc*.

3. Create a function *values* that consumes a symbol, *asuit*, and a list of distinct *card*s, *cardlist*, and produces a list of the values of the *card*s that have suit *asuit*.

4. Create a function *update-times* that consumes a list of *clock*s, *clist*, and a natural number, *m* (a number of minutes), and produces a list of updated *clock*s, each *m* minutes later on the same day (you can assume none of the new times go past 11:59 pm to the next day).

5. Create a function *average-length* that consumes a list of *event*s, *alist*, and a symbol, *t* (the type of the event), and produces the average duration of *event*s of that type. If a list is empty or there are no *event*s of a particular type, the average will be zero. Be careful not to divide by zero!

6. *Optional open-ended questions* You now have enough tools to be able to use the teachpack `world.ss` to create animations. Create a structure for a shape (type, size1, size2, mode, colour) and a structure for a piece (shape, posn). You will make an animation in which a world is a list of pieces, using *place-image* to place the pieces, in order from last in the list to first, on a scene (initially empty, 400×400). Here are some ideas to get you started.

   (a) Create a function *piece-list-to-scene* to consume a list of pieces and produce a scene.

   (b) Create a function *advance-rainbow* to consume a list of pieces and produce a list of pieces in the next colour of the rainbow.

   (c) Use your functions to create animations.