Lab 05: 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 `labinterface05.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 game* (*winner loser high low*))
;; A Game is a structure (make-game Str Str Nat Nat), where
;;   winner is the name of the winner of the game,
;;   loser is the name of the loser of the game,
;;   high is the winner's score
;;   low is the loser's score (high is greater than low).

(*define-struct timer* (*hours mins secs*))
;; A Timer is a structure (make-timer Nat Nat Nat), where
;;   hours is the number of hours,
;;   mins is the number of minutes in the range 0 - 59, and
;;   secs is the number of seconds in the range 0 - 59.

(*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
;;   hours 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 *fixed-game* that consumes a game, *agame*, and produces the game formed by giving all of the loser's points to the winner.

2. Create a function *convert-time* that consumes a *timer*, *t*, and produces the equivalent time in seconds.

3. Create a function *bigger-card* that consumes two *card*s, *card1* and *card2*, and produces the *card* with the higher value, or *card2* if they have the same value. Note: the suit of the *card* has no impact on its value.

4. Create a function *big-card-small-suit* that consumes two *card*s, *card1* and *card2*, and produces a *card* as follows: the *card* produced will have the value of the *card* with higher value and the suit of the *card* with lower value. If the two *card*s consumed have the same value, then *card1* should be produced.

5. Create a function *dur* that consumes two *clock* structures, *time1* and *time2*, and produces an integer indicating the number of minutes elapsed between two times. If *time2* is later than *time1*, you can assume that both times are on the same day. If *time2* is earlier than *time1*, you can assume that *time1* is on one day and *time2* is on the next day. For example:

   (*dur* (*make-clock* 16 10) (*make-clock* 1 40)) => 570

   (*dur* (*make-clock* 1 40) (*make-clock* 5 0)) => 200

   One way to approach this problem is to write a helper function that determines if two times are on the same day.

6. *Optional open-ended question* Choose a simple game or puzzle and devise a structure to represent it. Write one or more functions that consume a structure and produce the structure representing how it would change after a single move.