Lab 11: Structures

Create a separate file for each question. Keep them in your “Labs” folder, with the name \(l_{ij}q_k\) for Lab \(ij\), Question \(k\).

Download the headers for each function from the file \(l11\)-interface.rkt.

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 data and structure definitions:

\[
\begin{align*}
\text{(define-struct game (winner loser high low))} \\
\text{;; A Game is a (make-game Str Str Nat Nat)} \\
\text{;; requires:} \\
\text{;; * high (the winner’s score) > low (the loser’s score)} \\
\text{(define-struct card (rank suit))} \\
\text{;; A Card is a (make-card Nat (anyof 'hearts 'diamonds 'spades 'clubs))} \\
\text{;; requires:} \\
\text{;; * rank is in the range 1–13} \\
\text{(define-struct clock (hours mins))} \\
\text{;; A Clock is a (make-clock Nat Nat)} \\
\text{;; requires:} \\
\text{;; * hours is in the range 0–23} \\
\text{;; * mins is in the range 0–59} \\
\text{(define-struct meeting (start end))} \\
\text{;; A Meeting is a (make-meeting Clock Clock)} \\
\text{;; where:} \\
\text{;; * start is a Clock representing the starting time of the meeting} \\
\text{;; * end is a Clock representing the ending time of the meeting} \\
\text{;; requires:} \\
\text{;; * if end is equal to or earlier in the day than start, then end is exactly one day after start}
\end{align*}
\]

Language level: Intermediate Student with lambda

1. [Class exercise with lab instructor assistance]

Create a function \(\text{fixed-game}\) that consumes a Game structure, \(a\)-game, and produces the Game formed by giving all of the loser’s points to the winner.

For example:

\[
\text{(fixed-game (make-game "Jack" "Nick" 100 1))} \\
\Rightarrow (\text{make-game "Jack" "Nick" 101 0})
\]
2. Create a function \textit{big-card-small-suit} that consumes two card structures, \textit{card1} and \textit{card2}, and produces a new Card as follows: the Card produced will have the rank of the card with the higher rank, and the suit of the card with the lower rank. If the two Cards consumed have the same rank, then \textit{card1} should be produced.

For example:
\begin{verbatim}
     (big-card-small-suit (make-card 5 'hearts)
                      (make-card 7 'spades))
=> (make-card 7 'hearts)
\end{verbatim}

3. Create a function \textit{card-value} that consumes a Card, \textit{a-card}, and produces the “value” of a card where the value is the rank plus the points for the suit. Spades are worth 3 points, hearts are worth 5, diamonds are worth 10, and clubs are worth 20.

For example:
\begin{verbatim}
     (card-value (make-card 10 'diamonds)) => 20
\end{verbatim}

4. \textbf{[Part a]} Create a function \textit{games-won} that consumes a list of Games, \textit{results}, and a string, \textit{name}, and produces the number of games in \textit{results} that \textit{name} won. You must use abstract list functions in your solution.

For example:
\begin{verbatim}
     (games-won (list (make-game "Lori" "Troy" 52 34)) "Lori")
=> 1
\end{verbatim}

\textbf{[Part b]} Create a function \textit{high-score} that consumes a list of Games, \textit{results}, and a string, \textit{name}, and produces the highest winning score for player \textit{name}. You may assume \textit{name} won at least one of the games in \textit{results}. You must use abstract list functions in your solution.

For example:
\begin{verbatim}
     (high-score (list (make-game "A" "B" 10 9)
                     (make-game "C" "A" 30 20))
               "A") => 10
\end{verbatim}

5. \textbf{[Part a]} Create a function \textit{time-elapsed} that consumes two Clocks, \textit{start-time} and \textit{end-time}, and produces a number indicating the number of minutes elapsed between two times. If \textit{end-time} is later than \textit{start-time}, you can assume that both times are on the same day. If \textit{end-time} is equal to or earlier than \textit{start-time}, you can assume that \textit{start-time} is on one day and \textit{end-time} is on the next day. See examples. One way to approach this problem is writing a helper to determine if the two times are on the same day.

For example:
\begin{verbatim}
     (time-elapsed (make-clock 16 10) (make-clock 1 40))
=> 570
     (time-elapsed (make-clock 1 40) (make-clock 5 00))
=> 200
\end{verbatim}
Part b] Create a function total-meeting-time that consumes a list of Meetings, meetings, and produces the total time (in minutes) spent in all meetings. Your solution must use abstract list functions.

For example:

(define Davis-account (make-meeting (make-clock 16 00)
          (make-clock 16 30)))
(define quarter-review (make-meeting (make-clock 17 00)
          (make-clock 21 45)))
(define my-Monday (list Davis-account quarter-review))
(totall-meeting-time my-Monday) => 315

Optional open-ended questions

Make your own game!

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.

Try again…

Rewrite time-elapsed and total-meeting-time so that it produces a list or structure that has the number of hours and minutes separately, for easier reading.

Helpful tips

Structure templates

If may be helpful, when creating many functions for a specific structure, to create a template for the structure, or a list containing that structure.

Constants for structures

It is often useful to create named constants for sample structure values so that you can easily use the same value in multiple tests, rather than writing out the long form (make-sname field1 … fieldn) every time. See q5b for an example of this.