Extra Practice Problems (Module 4)

1. Write a function \((\text{posn-mult} \ p1 \ p2)\) that produces a Posn value which is the product of the two consumed Posns, \(p1\) and \(p2\) calculated by \((x1 \times x2 - y1 \times y2, x1 \times y2 + x2 \times y1)\)
   
   Example:
   
   \[
   (\text{posn-mult} \ (\text{make-posn} \ 3 \ 4)) \ (\text{make-posn} \ 1 \ 2)) \Rightarrow \ (\text{make-posn} \ -5 \ 10)
   \]

2. Write a function \((\text{posn-div} \ p1 \ p2)\) that produces a Posn value by calculating:

   \[
   \left(\frac{x1 \times x2 + y1 \times y2}{x2^2 + y2^2}, \frac{y1 \times x2 + x1 \times y2}{x2^2 + y2^2}\right)
   \]

3. Using the structure below,

   \[
   \text{(define-struct clicker \ (correct incorrect unanswered))}
   \]
   
   ;; A Clicker is a (make-clicker Nat Nat Nat)
   
   ;; where the total number of questions (correct + incorrect + unanswered) is divisible by 4
   
   write a function \((\text{clicker-grade} \ ck)\) that consumes a clicker structure, \(ck\), and calculates the total clicker grade from the best 75%.

   Example:

   \[
   (\text{clicker-grade} \ (\text{make-clicker} \ 75 \ 12 \ 13)) \Rightarrow 100
   \]

4. Using the structure below,

   \[
   \text{(define-struct student \ (asmt mid final participation))}
   \]
   
   ;; A Student is a (make-student Num Num Num Clicker)
   
   ;; requires:
   
   ;; asmt, mid, final are between 0 and 100,
   
   ;; and participation is of type Clicker.
   
   write a function \((\text{final-grade} \ stud)\) that consumes a Student structure, \(stud\), and produces the final grade of \(stud\). Considering the assignments are worth 20%, midterm 30%, final 45%, and best 75% of clicker marks are worth 5%.

   Example:

   \[
   (\text{final-grade} \ (\text{make-student} \ 100 \ 100 \ 100 \ (\text{make-clicker} \ 75 \ 12 \ 13))) \Rightarrow 100
   \]
5. Answer part a to e using the structures below.

\[
\text{(define-struct name (first last))}
\]
\[
;; \text{A Name is a (make-name Str Str)}
\]
\[
;; \text{requires:}
\]
\[
;; \text{first is the first name of a person}
\]
\[
;; \text{last is the last name}
\]

\[
\text{(define-struct officehour (day start end))}
\]
\[
;; \text{An Officehour is a (make-officehour Str Nat Nat)}
\]
\[
;; \text{requires:}
\]
\[
;; \text{day is the day of the office hour of a person}
\]
\[
;; \text{start is the start time of the office hour}
\]
\[
;; \text{end is the end time of the office hour}
\]
\[
;; \text{start and end are a valid time on the 24-hour clock}
\]

\[
\text{(define-struct personnel (identity availability))}
\]
\[
;; \text{A Personnel is a (make-personnel Name Officehour)}
\]

a. Write a function
\[
\text{(update-info new-ta-first new-ta-last new-time)}
\]
that consumes two strings, new-ta-first & new-ta-last, and an Officehour structure, new-time. The function must produce a Personnel structure containing the new information.
Example:
\[
\text{(update-info "Nisha" "Eappen" (make-officehour "Mon" 15 16))} \Rightarrow
\text{(make-personnel (make-name "Nisha" "Eappen")
\hspace{1cm} (make-officehour "Mon" 15 16))}
\]

b. Write a function \text{(plain-english ta)} that consumes a Personnel structure, ta and produces a string summarizing the details of the ta. The produced string will have the following format:
"First Last has office hours on Day from Start until End."
Example:
\[
\text{(plain-english (make-personnel (make-name "Bettina" "Boucher")
\hspace{1cm} (make-officehour "Monday" 14 15)))} \Rightarrow
"Bettina Boucher has office hours on Monday from 14:00 until 15:00."
\]

c. Write a function \text{(first-longer-than-last? aname)} that consumes a Name structure, and produces true if the first name is longer than the surname (last name), and false otherwise.
Examples:
\[
\text{(first-longer-than-last? (make-name "Bettina" "Boucher"))} \Rightarrow
\text{false}
\]
\[
\text{(first-longer-than-last? (make-name "Mbabi" "Tema"))} \Rightarrow \text{true}
\]
\[
\text{(first-longer-than-last? (make-name "Nisha" "Eappen"))} \Rightarrow \text{false}
\]
d. Write a function `(how-long? oh)` that consumes `oh`, an Officehour structure, and produces the length of the office hours.
   - Example:
     `(how-long? (make-officehour "Monday" 14 18)) => 4`

e. Write a function `(weekly-total inst1 inst2 inst3)` that consumes three Personnel values and produces the total of their office hours. Hint: you may use `how-long?` as a helper function.
   - Example:
     `(weekly-total
      (make-personnel (make-name "Victoria" "Sakhnini")
                      (make-officehour "Monday" 13 14))
      (make-personnel (make-name "Collin" "Roberts")
                      (make-officehour "Tuesday" 13 15))
      (make-personnel (make-name "Joseph" "Istead")
                      (make-officehour "Friday" 8 10)))

     => 5`