CS 2500/Accelerated Exam 1—Fall 2017

Matthias Felleisen

October 18, 2017

- The exam is a **one-hour** exam.

- We will not answer any questions during the exam. If you believe a problem statement is ambiguous, choose *any* non-trivial interpretation.

- Write down the answers in the space provided, including the back of the given spaces.

- You may use the paper copy of the book or your notes.

- You may *not* use any electronic gadgets (for example, watches, google glasses, phones, tablets, laptops). Any use of an electronic gadget will lead to immediate expulsion from the exam and class.

- You may use all the definitions, expressions, and functions found ISL, especially those suggested in hints. Define everything else.

- Unless a problem requests a solution that does not use the abstractions of ISL, you may use these abstractions. Similarly, unless a problem demands a solution that uses the abstractions of ISL, you do not have to use these abstractions.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Max. Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>15</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>37</strong></td>
</tr>
</tbody>
</table>


Problem 1 Use local to eliminate all nested expressions in the following function definition so that the order of evaluation remains the same:

```
(define UFO ...)   
(define TANK ...)   
(define MISSILE ...) 

(define-struct game [ufo tank mis])
; Game is (make-game Posn Posn Posn).

; Game Image -> image
; add the images of the tank and MISSILE to img
(define (render w img)
  (place-image TANK
    (posn-x (game-tank w))
    (posn-y (game-tank w))
  (place-image MISSILE
    (posn-x (game-mis w))
    (posn-y (game-mis w))
  img)))
```
Problem 2 Design the function `adder`, which consumes a list of `Posn` and computes the sum of all fields. Show the template for the structural design recipe. Assume a template for `Posn` is available.
Problem 3 Take a look at the following structure type and data definitions:

```scheme
(define-struct world [time-left scores])
(define-struct score [name value])
; A World is (make-world N List-of-scores).
; A List-of-scores is one of:
;   -- '()
;   -- (cons Score List-of-scores)
; A Score is (make-score String N).
; N: recall that N represents natural numbers.
```

Design the function `update-score`. It consumes a World and a String and increases the value field in the Scores whose name field is the same as the given String.
intentionally left blank
Problem 4 Design the common abstraction, including signature for these two function definitions: 10pts.

**swap**

; [Listof Posn] Number -> [Listof Posn]
; swap the content of the first posn struct
; on lop whose x coordinate is x
(check-expect (swap (list (make-posn 3 4)) 5) (list (make-posn 3 4)))
(check-expect (swap (list (make-posn 3 4)) 3) (list (make-posn 4 3)))
(define (swap lop x)
  (cond
    [(empty? lop) '()]
    [else (if (= (posn-x (first lop)) x)
      (cons (swap-posn (first lop)) (rest lop))
      (cons (first lop) (swap (rest lop) x)))]))

; Posn -> Posn
; reflex the Posn along the diagonal
(define (swap-posn p)
  (make-posn (posn-y p) (posn-x p)))

**reset**

; [Listof [list Symbol Number]] Symbol -> [Listof [list Symbol Number]]
; reset the number of the first pair
; whose symbol is s to 0
(check-expect (reset '((a 2) (b 3)) 'b) '((a 2) (b 0)))
(check-expect (reset '((a 2) (b 3)) 'c) '((a 2) (b 3)))
(define (reset losn s)
  (cond
    [(empty? losn) '()]
    [else (if (symbol=?= (first (first losn)) s)
      (cons (reset-pair (first losn)) (rest losn))
      (cons (first losn) (reset (rest losn) s))])))

; [list Symbol Number] -> [list Symbol Number]
; reset the number of the pair to 0
(define (reset-pair p)
  (list (first p) 0))