Solutions to Sample Quiz 2

Problem 1

Set 1: 8
Set 2: 12
Set 3: (a b c bar e f)

Problem 2

Ask in class to see the box and pointer diagrams.

Problem 3

<1> P2
<2> P1
<3> GE
<4> 3
<5> E1
<6> 9
<7> E2
<8> E2
<9> E2
<10> (+ a x m)
<11> GE

Problem 4

a) Ask in class.

b) (define (get-first-name record)
    (caar record))

(define (get-last-name record)
    (cadar record))

(define (get-id record)
    (cadr record))

(define (get-major record)
    (caaddr record))

(define (get-year record)
    (cdaddr record))
c) If you followed directions, you could not get the correct solution to this part. The following two solutions were accepted as correct.

The first does not violate the abstraction barriers, but would not work. You can not use \texttt{set!} on a value in a list. It can only be used on a variable. However, the wording of the question would lead you to this solution, so it was accepted as correct.

\begin{verbatim}
(define (change-major! record new-major)
  (set! (get-major record) new-major)
  (get-id record))
\end{verbatim}

This next definition violates the abstraction barriers, but would work.

\begin{verbatim}
(define (change-major! record new-major)
  (set-car! (caddr record) new-major)
  (get-id record))
\end{verbatim}

There is no way to write this procedure without violating the abstraction barriers.

\textbf{Problem 5}

For part a, your modified code would be as follows:

\begin{verbatim}
(define (make-inc init)
  (let ((value init))
    (define (inc-val x)
      (set! value (+ value x))
      value) ;this line was missing on the exam
    (define (dispatch m)
      (cond ((eq? m 'inc-val) inc-val)
            ((eq? m 'reset-val) (set! value 0) value)
            (else (error "Invalid message - MAKE-INC" m)))
      dispatch))
\end{verbatim}

For part b, your modified code would be as follows:

\begin{verbatim}
(define (make-inc init)
  (let ((value init))
    (define (inc-val x)
      (set! value (+ value x))
      value) ;this line was missing on the exam
    (define (set-val x)
      (set! value x)
      value)
    (define (dispatch m)
      (cond ((eq? m 'inc-val) inc-val)
            ((eq? m 'set-val) set-val)
            (else (error "Invalid message - MAKE-INC" m)))
      dispatch))
\end{verbatim}