Exam 1 Solutions

Problem 1
a) 11
b) a procedure
c) 4

Problem 2
\[
\begin{array}{ccc}
#f & #f & #t \\
#f & #t & #t \\
#t & #t & #t \\
\end{array}
\]

Note 1: `(eq? b c)` is not `eq?` because the pair structure is different. However, it
would be best not to use `eq?` with numbers, since the result can be unexpected.
DrScheme would return `#f`, but unspecified or error was also accepted for this
answer.

Note 2: `(eq? (car a) (car b))` will return `#f` in DrScheme, but is
unspecified or an error in other Scheme implementations. Any of these three
answers was accepted.

Problem 3
\[
\begin{align*}
&\text{(caddr first-list)} \\
&\text{(caddr second-list)}
\end{align*}
\]

Problem 4
fun adds a to b by adding 1 to the recursive call of fun with one subtracted from a
Time: Theta(n)
Space: Theta(n)
Variable n is dependent on: a
Recursive process is generated

Problem 5
\[
\text{(define (apply-backwards f)}
\quad \text{lambda (x y) (f y x))})
\]

Problem 6
\[
\text{(define (prod-of-every-third a b)}
\quad \text{(prod (lambda (x) x)}
\quad \quad \quad \quad a
\quad \quad \quad \quad \quad \quad \text{(lambda (x) (+ x 3))}
\quad \quad \quad \quad b))}
\]
Problem 6 continued
(define (fact n)
  (prod (lambda (x) x)
      1
      (lambda (x) (+ x 1))
      n))

Problem 7
(define (combine-two-lists f l1 l2)
  (if (null? l1)
      nil
      (cons (f (car l1) (car l2))
            (combine-two-lists f (cdr l1) (cdr l2)))))

Time: Theta(n), where n is the length of the list (both assumed equal)
Space: Theta(n)
Recursive process

(define (max-elts l1 l2)
  (combine-two-lists max l1 l2))

Problem 8
(define (recipe-name recipe)
  (car recipe))

(define (num-recipe-ingredients recipe)
  (caadr recipe))

(define (num-recipe-servings recipe)
  (cdadr recipe))

(define first-recipe car)

(define rest-recipes cdr)

(define (count-servings recipe-list)
  (if (null? recipe-list)
      0
      (+ (num-recipe-servings (first-recipe recipe-list))
          (count-servings (rest-recipes recipe-list)))))

Problem 9
(define (car obj)
  (obj 1))

(define (cdr obj)
  (obj 2))