Solutions to Exam 1

Problem 1

17
19
12
#f

Problem 2

(cadadr a) or (car (cdr (car (cdr a))))
(caddr b) or (car (cdr (cdr b)))
(cdr c)

Problem 3

#f  #f  #t
#t  #t  #t
#f  #f  #t
#t  #t  #t

Problem 4

(define (numer r) (car r))
(define (denom r) (cadr r))
or
(define numer car)
(define denom cadr)

(define (numer r) (r 'numer))
(define (denom r) (r 'denom))

(define (numer r) (r (lambda (x y) x)))
(define (denom r) (r (lambda (x y) y)))
or
(define (numer r) (car (r cons)))
(define (denom r) (cdr (r cons)))

Extra Credit

Data abstraction, or just abstraction.
Problem 5

Θ(n)
Θ(n)
recursive

Θ(n)
Θ(1)
iterative

Problem 6

(define (combine-two-lists combiner list1 list2)
  (if (null? list1)
      nil
      (cons (combiner (car list1) (car list2))
            (combine-two-lists combiner
                           (cdr list1)
                           (cdr list2))))

(define (mul-two-lists list1 list2)
  (combine-two-lists * list1 list2))

Problem 7

(define (quad-split painter n)
  (if (= n 0)
      painter
      (let ((smaller (quad-split painter (- n 1))))
        (below (beside smaller painter)
               (beside painter smaller))))