Problem 1

Problem 2

Applicative order: Evaluate all subexpressions first, then apply the first to the rest.
(Scheme uses this.)

Normal order: No arguments are evaluated until they are needed. Fully expand, then reduce.

In Scheme (applicative order), the following two items could be printed:

one two plus

two one plus

In normal-order Scheme, the following two items could be printed:

plus one two

plus two one

Problem 3

(caddr first-list)

(caadr second-list)

Problem 4

(define (apply-twice f)
  (lambda (x) (f (f x))))

Problem 5

(define (prod a b)
  (if (> a b)
      1
      (* a (prod (+ a 1) b))))

Time: $\Theta(n)$

Space: $\Theta(n)$

n is dependent upon b-a

prod generates a recursive process

Problem 6

For cons defined as follows,

(define (cons x y)
  (lambda (m)
    (cond ((= m 0) x)
          ((= m 1) y)
          (else (error “Unknown message – CONS” m)))))

the corresponding definitions of car and cdr would be the following.

(define (car z)
  (z 0))

(define (cdr z)
  (z 1))