QUESTIONs: What is printed at each line?

1. > (define x (+ 2 3))
   > x => 5
   > (define y ((lambda (a) a) (* 3 4)))
   > y => 12
   > (define z ((lambda (b) (+ b 10)) y))
   > z => 22

2. > (define count 0)
   > (define (foo x y) (x y))
   > (define z (foo (lambda (a) (set! count a) (* a a))
                 (begin (set! count (+ 1 count)) count)))
   infinite loop
   > count => 0
   > z => ??

3. > (define count 0)
   > (define (incr!) (set! count (+ count 1)))
   > (define (foo x)
       (let ((y (begin (incr!) count)))
         (if (<= count 1)
           (foo y)
           x)))
   > (foo 10) => infinite loop

Paradigm Shift Again (Why Not?)

Lists Again (and again, and again, and again, and again...)

QUESTIONS

1. Write a rule for car of list. For example, (car (1 2 3 4) ?x) would have ?x bound to 1.
   (rule (car (?car . ?cdr) ?car))

2. Write a rule for cdr of list. For example, (cdr (1 2 3) ?y) would have ?y bound to (2 3).
   (rule (cdr (?car . ?cdr) ?cdr))

3. Define our old friend, member, so that (member 4 (1 2 3 4 5)) would be satisfied, and (member 3 (4 5 6)) would not, and (member 3 (1 2 (3 4) 5)) would not.
   (rule (member ?item (?item . ?cdr)))
   (rule (member ?item (?car . ?cdr)) (member ?item ?cdr))

4. Define its cousin, deep-member, so that (deep-member 3 (1 2 (3 4) 5)) would be satisfied as well.
   (rule (deep-member ?item (?item . ?cdr)))
   (rule (deep-member ?item (?car . ?cdr)) (deep-member ?item ?car))
   (rule (deep-member ?item (?car . ?cdr)) (deep-member ?item ?cdr))

Note how ?item can either be in ?car or ?cdr, so we need three rules.