

Exam 2 Solutions

Problem 1

(1 2 3 4)

(5 1 2 3 4)

(1 2)

(6 1 2)

Problem 2

```
(tree-manip test-tree
  0
  (lambda (x) x)
  car
  cdr
  +)
```

```
(tree-manip test-tree
  nil
  (lambda (x) (* x x))
  car
  cdr
  cons)
```

```
(tree-manip test-tree
  nil
  list
  car
  cdr
  append)
```

```
(tree-manip test-tree
  nil
  list
  cdr
  car
  append)
```

Problem 3

<1>: P1
<2>: P2
<3>: GE
<4>: E2
<5>: GE
<6>: E1
<7>: 0
<8>: 1

Problem 4:

```
(define (insert-at-end! lst obj)
  (cond ((null? lst) nil)
        ((null? (cdr lst)) (set-cdr! lst (cons obj nil)))
        (else (insert-at-end! (cdr lst) obj))))
```

Problem 5

```
(define counter-starting-at-5 (make-counter 5))

((counter-starting-at-5 'add-to-counter) 3)

(define (make-counter init)
  (let ((value init))
    (define (add-to-counter x)
      (set! value (+ value x))
      value)
    (define (dispatch m)
      (cond ((eq? m 'add-to-counter) add-to-counter)
            ((eq? m 'get-value) value)
            (else (error "Invalid message - MAKE-COUNTER" m))))
    dispatch))

(define (make-counter init)
  (let ((value init))
    (define (add-to-counter x)
      (set! value (+ value x))
      value)
    (define (reset-value)
      (set! value 0)
      value)
    (define (dispatch m)
      (cond ((eq? m 'add-to-counter) add-to-counter)
            ((eq? m 'get-value) value)
            ((eq? m 'reset-value) (reset-value))
            (else (error "Invalid message - MAKE-COUNTER" m))))
    dispatch))
```