91.301, Organization of Programming Languages  
Spring 2010, Prof. Yanco  

Solutions to Sample Exam 2

Problem 1

Set 1: 8  
Set 2: 12  
Set 3: (a b c bar e f)

Problem 2

<table>
<thead>
<tr>
<th>#f</th>
<th>#f</th>
<th>#t</th>
</tr>
</thead>
<tbody>
<tr>
<td>#f</td>
<td>#f</td>
<td>#t</td>
</tr>
<tr>
<td>#f</td>
<td>#f</td>
<td>#t</td>
</tr>
<tr>
<td>#t</td>
<td>#t</td>
<td>#t</td>
</tr>
</tbody>
</table>

Problem 3

Ask in class to see the box and pointer diagrams.

Problem 4

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

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

Problem 5

<1>  P2  
<2>  P1  
<3>  GE  
<4>  3  
<5>  E1  
<6>  9  
<7>  E2  
<8>  E2  
<9>  E2  
<10> (+ a x m)  
<11> GE
**Problem 6**

For part a, your modified code would be as follows:

```scheme
(define (make-inc init)
  (let ((value init))
    (define (inc-val x)
      (set! value (+ value x))
      value) ;this line was missing on the exam
    (define (dispatch m)
      (cond ((eq? m 'inc-val) inc-val)
        ((eq? m 'reset-val) (set! value 0) value)
        (else (error "Invalid message – MAKE-INC" m))))
    dispatch))
```

For part b, your modified code would be as follows:

```scheme
(define (make-inc init)
  (let ((value init))
    (define (inc-val x)
      (set! value (+ value x)) ;this line was missing on the exam
    (define (set-val x)
      (set! value x)
      value)
    (define (dispatch m)
      (cond ((eq? m 'inc-val) inc-val)
        ((eq? m 'set-val) set-val)
        (else (error "Invalid message – MAKE-INC" m))))
    dispatch))
```

**Problem 7**

```scheme
(define (map! op lst)
  (if (null? lst)
    'done
    (begin (set-car! lst (op (car lst)))
      (map! op (cdr lst))))))
```