Quiz 2 Solutions

Problem 1

\[(1 \ 2)\]
\[(3 \ 1 \ 2)\]
\[(4 \ 2)\]
\[(3 \ 4 \ 2)\]
\[(5 \ 2)\]
\[(3 \ 5 \ 2)\]

Problem 2

\[7\]
\[2\]
\[27\]
\[27\]

Problem 3

<table>
<thead>
<tr>
<th>false</th>
<th>false</th>
<th>true</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
</tbody>
</table>

Problem 4

\[
\text{Problem 4}
\]

\[
\text{(define (amount s tree)}
\text{ (accumulate-tree (lambda (x) (if (eqv? x s) 1 0)))}
\text{ + 0))}
\]

Problem 5

\[
\text{Problem 5}
\]

\[
\text{(define (make-toggle)}
\text{ (let ((answer 'no))}
\text{ (lambda ()}
\text{ (if (eq? answer 'no)}
\text{ (begin (set! answer 'yes) answer)}
\text{ (begin (set! answer 'no) answer))))}
\]
Problem 6

The missing code:

```
((eq? action 'reset)
  (set! current (car old))
  (set! old (cdr old))
  current)
```

Problem 7

Learned by student:  More fun with scheme: object oriented code
Learned by opl-student:  metacircular evaluator
Learned by opl-student:  More fun with scheme: generic operators
Learned by student:  I know More fun with scheme: streams

Problem 8

```
<1>  P2
<2>  L1
<3>  E1
<4>  GE
<5>  E1
<6>  x
<7>  E3
<8>  0
<9>  10
<10> E1
<11> 10
<12> GE
<13> P1
<14> P3
```

7