MASSACHVSETTS INSTITVTE OF TECHNOLOGY
Department of Electrical Engineering and Computer Science
6.001—Structure and Interpretation of Computer Programs
Spring Semester, 2005

Quiz II

NAME: **Sample Solutions**

Part 1: (25 points)

Question 1:

```
(define (rotate-left cycle)
  (cdr cycle))
```

Question 2:

```
(define (rotate-right cycle)
  (define (aux where start)
    (if (eq? (cdr where) start)
       where
       (aux (cdr where) start)))
  (aux cycle cycle))
```

Question 3:

```
(define (insert-cycle! new cycle)
  (let ((new-cell (list new)))
    (set-cdr! new-cell cycle)
    (set-cdr! (rotate-right cycle) new-cell)
    'done))
```

Question 4:

```
(define (delete-cycle! cycle)
   (set-cdr! (rotate-right cycle) (rotate-left cycle))
   (set-cdr! cycle '())
   'done)
```

Part 2: (30 points)

Question 5:

	Enclosing environment
E1	**E2**
E2	**GE**
E3	**GE**
E4	**E1**

Question 6:

Variable	Environment	Value to which bound
set!-start	GE	**P1**
set!-careful	GE	**P3**
val	E2	**5**
foo	GE	**P4**
new	E4	**(10)**
action	E4	**new**
var	E3	**P4**
val	E3	**10**
current	E1	**10**

Question 7:

location: **E2**

value: **(5)**

Part 3 (15 points)

Question 8: A

Question 9: K

Question 10: H

Question 11: G

Question 12: J

Part 4 (30 points)

Question 13.

2

Question 14.

no method

Question 15.

```
       'SHEETS (lambda ()
                    (fold-right + 0
                       (map (lambda (thing) (ask thing 'SHEETS))
                             contents)))
```

Question 16.

110

Question 17.

0

Question 18.

110

Question 19:

```
(define (aged-cabinet self name)
        (let ((cabinet-part (cabinet self name))
              (age 0))
             (make-handler
              'aged-cabinet
              (make-methods
               'ADD-THING (lambda (thing)
                             (if (<= age 4)
                                 (begin
                                    (ask cabinet-part 'ADDTHING thing)
                                    (set! age (+ age 1)))
                                 'broken)))
              cabinet-part)))
```

Question 20:

```
(define (located-cabinet self name x y)
     (let ((cabinet-part (cabinet self name))
           (located-object-part (located-object self x y)))
          (make-handler
             'located-cabinet
             (make-methods)
             cabinet-part located-object-part)))
```

Question 21:

```
(lambda (new) (display ‘‘My location has changed.’’)
              (ask located-object-part ’set-x! new))
```

Question 22:

NO

```
’SET-X-Y! (lambda (newx newy) (ask self ’set-x! newx)
                              (ask self ’set-y! newy))
```