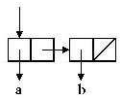
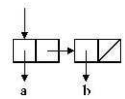
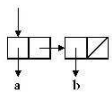
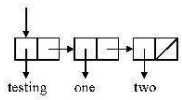
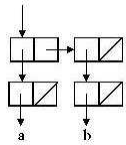
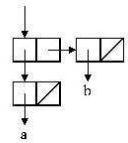
Symbols and Quote

Scheme uses the character ' to distinguish between symbols and their values. The value of 'x is the symbol x and not the value of the thing named x.

The quote syntax 'expr is interpreted by the reader as syntactic sugar for (quote expr).

Practice giving the printed representation and the box-and-pointer diagram for each of the following:

'(a b) ⇒ (a b) 	(cons 'a '(b)) ⇒ (a b) 
(list 'a 'b) ⇒ (a b) 	(quote (testing one two)) ⇒ (testing one two) 
(list '(a) '(b)) ⇒ ((a) (b)) 	(cons '(a) '(b)) ⇒ ((a) b) 

Given the definitions above the line below, what does the Scheme interpreter print for each of the expressions below the line?

```
(define a 3)
(define b 4)
(define c (list 5 6))
```

```
(define p (list cons +))
(define q '(cons +))
(define r (list 'cons '+))
```

```
(define x '(a b))
(define alpha '(a Greek letter))
(define beta '(I don't know))
```

```
(list 'a b)
⇒ (a 4)

'(a b)
⇒ (a b)

(cons b x)
⇒ (4 a b)

(list 'b c)
⇒ (b (5 6))
```

```
(car x)
⇒ a

(cdr alpha)
⇒ (Greek letter)

((car p) 3 4)
⇒ (3 . 4)

((cadr p) 3 4)
⇒ 7
```

```
((car r) 3 4)
⇒ Error -- can't apply a symbol

((cadr q) 3 4)
⇒ Error -- can't apply a symbol

(car ''a)
⇒ quote

beta
⇒ (I don (quote t) know)
```

Symbol Equality

To determine if two numbers are identical, we use the Scheme primitive `=`. To determine if two symbols are identical, we use the Scheme primitive `eq?`. For example,

```
(eq? 'apples 'apples)
⇒ #t
```

```
(eq? 'apples 'oranges)
⇒ #f
```

We will discuss other variations of `eq?` and other equality predicates next week, but remember that the behavior of `eq?` on numerical arguments is **unspecified**:

```
(eq? 1 1)
⇒ unspecified
```

```
(let ((x (+ 1 2)))
  (eq? x x))
⇒ unspecified
```

Building memq

`(memq <item> <list>)` will return return a sublist beginning with the first occurrence of the symbol `<item>`. If the symbol `<item>` is not contained in the `<list>`, then `memq` will return false.

```
(define (memq item lst)
  (cond ((null? lst) false)
        ((eq? item (car lst)) lst)
        (else (memq item (cdr lst)))))
```

What is the result of using `memq` in the following situations:

```
(memq 'robots '(building robots is fun))
⇒ (robots is fun)
(memq 'robot '(building robots is fun))
⇒ false
```

```
(memq 'bolts '(screws (bolts washers)
nails))
⇒ false
(memq 'bolts '(screws (bolts washers)
nails bolts pins))
⇒ (bolts pins)
```

Puzzle

Can you write a scheme expression that prints itself?

(Hint: Consider how a cell reproduces. It contains its own blueprint (the DNA strand in the nucleus); reproduction involves (a) copying the blueprint, (b) implementing the blueprint. That is, it uses the blueprint twice.)

```
((lambda (x)
   (list x (list (quote quote) x)))
 (quote (lambda (x)
          (list x (list (quote quote) x)))))
```

Administrivia

- Quiz #1 Statistics:
  - Average: 76
  - Approximate ranges: A(100-85) B(84-69) C(68-53) D/F(52-0)
- Problem Set #3 : Due next Wednesday