6.001 Recitation 12: Mutation!

21 March 2007

RI: Gerald Dalley (dalleyg@mit.edu)
http://people.csail.mit.edu/dalleyg/6.001/SP2007/

Modifying Bindings

`(set! <name> <value>)` → `undefined`

- Looks for the binding of **`<name>`** and changes the binding to the value of the **`<value>`** expression
  Example:

```
(define x 10)
x →
(set! x (* 10 20))
x →
```

- Is **`set!`** a special form?

Modifying Pairs

`(set-car! <pair> <val>)` → `undefined`
`(set-cdr! <pair> <val>)` → `undefined`

- Change the **`car`**/**`cdr`** part of the **`cons`** cell **`<pair>`** to **`<val>`**. Example:

```
(define x (list 1 2))
x →
(set-car! x 3)
x →
```

- Are **`set-car!`** and **`set-cdr!`** special forms?

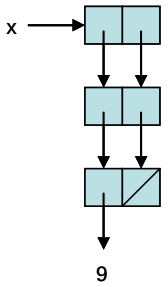
Thought Question

- What is the difference between changiing a binding (using **`set!`**) and changing an object (using **`set-car!`** or **`set-cdr!`**)?

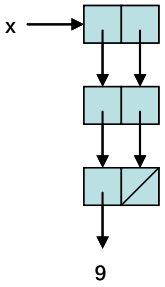
Warmup & Subtle Points

```
(define x '(a b c))
(set-cdr! x x)
(car x)    =>
(length x) =>
```

1. Write a Scheme expression that makes the structure (without using mutation!).
2. Write what Scheme prints for the structure (if you can).

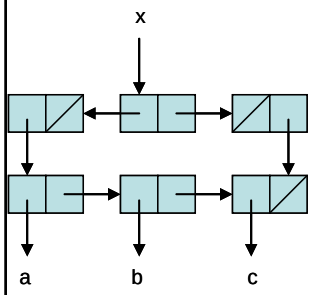


3. Show how the mutation affects the box-and-pointer diagram and the printed representation, assuming the structure is named **x**.

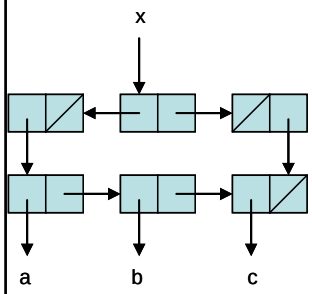



```
(set-cdr! (car x) '(8))
```

1. Write a Scheme expression that makes the structure (without using mutation!).
2. Write what Scheme prints for the structure (if you can).

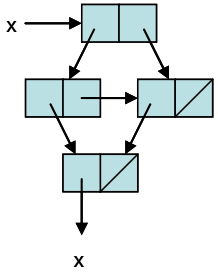


3. Show how the mutation affects the box-and-pointer diagram and the printed representation, assuming the structure is named **x**.

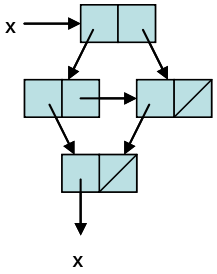



```
(set-car! (cddr x) (caaar x))
```

1. Write a Scheme expression that makes the structure (without using mutation!).
2. Write what Scheme prints for the structure (if you can).

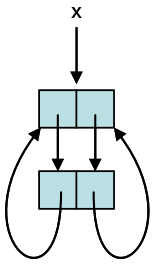


3. Show how the mutation affects the box-and-pointer diagram and the printed representation, assuming the structure is named **x**.

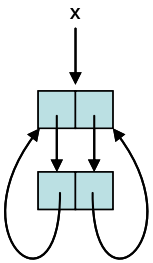



```
(set-cdr! (first x) (second x))
```

1. Write a Scheme expression that makes the structure.
2. Write what Scheme prints for the structure (if you can).



3. Show how the mutation affects the box-and-pointer diagram and the printed representation, assuming the structure is named **x**.




```
(set-car! (cdr x) '())
(set-cdr! (car x) '())
```

1. Draw a box-and-pointer representation of the expression's value.

```
(let ((w (list 6 7 8)))
  (set-car! w w)
  (set! w (list w w))
  w))
```

2. Show how the mutation affects the box-and-pointer diagram and the printed representation, assuming the value of the expression is named **x**.

```
(set-car! (car x) (cddr x))
```

1. Draw a box-and-pointer representation of the expression's value.

```
(let ((y '((a) (b))))
  (set-cdr! (first y) y)
  (set-car! (second y) (cdr y))
  (set! y (car y))
  y))
```

2. Show how the mutation affects the box-and-pointer diagram and the printed representation, assuming the value of the expression is named **x**.

```
(set-cdr! x (third x))
(set-cdr! (cdr x) nil)
```

What does mystery do?

```
(define (mystery x)
  (define (loop x y)
    (if (null? x)
        y
        (let ((temp (cdr x)))
          (set-cdr! x y)
          (loop temp x))))
  (loop x '()))
```

```
(define a (list 1 2 3 4))
a ==>
(define b (mystery a))
a ==>
b ==>
```