MASSACHVSETTS INSTITVTE OF TECHNOLOGY
Department of Electrical Engineering and Computer Science
6.001—Structure and Interpretation of Computer Programs
Spring Semester, 1998

Recitation – Friday, February 18

1. Student Presentation

Explain to the class how Scheme prints compound data structures. Use as examples the way the following three things print:

```
(define s1 (cons 7 14))          s1 -> ???
(define s2 (list 7 14))          s2 -> ???
(define s3 (cons 7 nil))         s3 -> ???
```

2. Cons Pairs and Lists

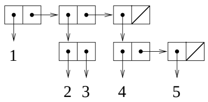
Recall the contract for `cons`, `car`, `cdr`, and `null`.

```
(car (cons a b)) == a
(cdr (cons a b)) == b
(pair? (cons a b)) == #t
```

There are three main methods of representing cons and list structures. You should be able to convert between them.

Scheme Expression: `(list 1 (cons 2 3) (list 4 5))`

Box and Pointer Diagram:



Scheme Printout: (1 (2 . 3) (4 5))

3. Practice

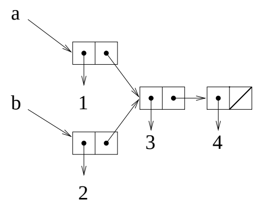
Draw box and pointer diagrams and write what will Scheme print for the following expressions.

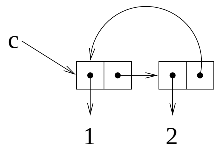
```
(1)==> (define x (cons 5 2))

(2)==> (car x)

(3)==> (cdr x)

(4)==> (car (cdr x))

(5)==> (define y (cons sqrt x))

(6)==> (car (cdr y))

(7)==> (car y)

(8)==> (define z (cons ((car y) 49) x))
```

Write a Scheme expression that will print each of the following. Also draw box and pointer diagrams.

```
(1)==> (1 2 3)
(2)==> (1 2 . 3)
(3)==> ((1 (2)) . 3)
```

Write Scheme expressions that correspond to the following.





4. Functions on Lists

We saw that we have the primitive function `pair?` to see if an object is a pair. What if we wanted to write the function `list?` to see if an object is a list?

What is the contract for `list?` $\forall x_1, x_2, \ldots, x_n$ `(list? (list` x_1 x_2 ... x_n`)) == #t`

What's another way to write it? `(list? nil) == #t`
`(list? (cons` x l`)) == #t <==> (list?` l`)`

Now, how can we write `list?` in scheme?

```
(define (list? x)
  (or (null? x)
      (and (pair? x)
           (list? (cdr x)))))
```

What is the Order of Growth of `pair?` and `list?` ?

5. More Functions on Lists

We know we have `car`, `cdr`, `cadr`, etc, as accessors for list structures. But what if we wanted to reference the n^{th} element of a list? Write the function `list-ref` that takes a list `x` and an integer `n` and returns the n^{th} element of the list `x`.

```
(define (list-ref x n)
  (if (= n 0)
      (car x)
      (list-ref (cdr x) (- n 1))))
```

Write the function `length` that takes a list `x` and returns the length of the list.

```
(define (length x)
  (if (null? x)
      0
      (+ 1 (length (cdr x)))))
```

6. Kons/Kar/Kdr as Higher-Order Procedures

In lecture we saw an example of writing `cons`, `car`, and `cdr` as procedures. Here's an even more elegant (albeit more obscure) way of doing it. Can you figure out how this is working?

```
(define kons (lambda (x y) (lambda (f) (f x y))))
(define kar (lambda (p) (p (lambda (a b) a))))
(define kdr (lambda (p) (p (lambda (a b) b))))
```

What would this do?

```
(define x (kons 1 2))
(x +)
```