MASSACHVSETTS INSTITVTE OF TECHNOLOGY
Department of Electrical Engineering and Computer Science
6.001—Structure and Interpretation of Computer Programs
Spring Semester, 1999

Recitation – Friday, February 12

1. Cons Pairs and Lists

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

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

```
(list a b c) ==  (cons a
                    (cons b
                       (cons c nil)))
```

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

Scheme Expression	Box & Pointer	Scheme Printout
(cons 1 2)		
(cons 1 (cons 2 nil))		
(cons 1 nil)		
(cons 1 (cons 2 3))		
(cons (cons 1 2) nil)		
(list 1 2 3 4)		
(list 1 (cons 2 3) (list 4 5))		

2. Other Accessors

In scheme, we often want to access elements deep in a cons structure. Therefore, the following accessors have been defined to help us out:

```
(cadr x)  == (car (cdr x))
(caddr x) == (car (cdr (cdr x)))
(cdaar x) == (cdr (car (car x)))
```

```
(cddr x)   == (cdr (cdr x))
(cdadar x) == (cdr (car (cdr (car x))))
etc, etc...
```

For lists, we also often want to easily access the n'th element of a list. The accessors `first`, `second`, `third`, ..., `tenth` are defined to access the corresponding values of a list. For example,

```
(sixth (list 1 2 3 4 5 6 7 8 9))
;Value: 6
```

How could you define `first`, `second`, `third`, and `fourth` using the `c???r` functions?

```
(first x)  ==
(second x) ==
```

```
(third x)  ==
(fourth x) ==
```

3. Practice

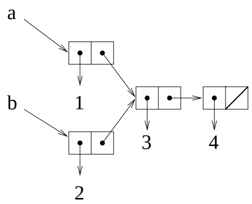
Draw box and pointer diagrams and write what will Scheme print for the following expressions.

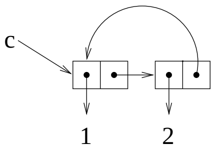
```
⇒ (define x (cons 5 2))
⇒ (car x)

⇒ (cdr x)

⇒ (car (cdr x))

⇒ (define y (cons sqrt x))
⇒ (car (cdr y))

⇒ (car y)

⇒ (define z (cons ((car y) 49) x))
⇒ z
```

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

```
⇒
;Value: (1 2 3)
⇒
;Value: (1 2 . 3)
⇒
;Value: ((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)



)
```

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

`pair?` is $\Theta(\ )$ and `list?` is $\Theta(\ )$