

```
fun append (xs,ys) =  
  if xs=[]  
  then ys  
  else (hd xs)::append(tl xs,ys)  
  
fun map (f,xs) =  
  case xs of  
    [] => []  
  | x::xs' => (f x)::(map(f,xs'))  
  
val a = map (increment, [4,8,12,16])  
val b = map (hd, [[8,6],[7,5],[3,0,9]])
```

Programming Languages

Dan Grossman
2013

mcons For Mutable Pairs

cons cells are immutable

What if you wanted to mutate the *contents* of a cons cell?

- In Racket you cannot (major change from Scheme)
- This is good
 - List-aliasing irrelevant
 - Implementation can make `list?` fast since listness is determined when cons cell is created

Set! does not change list contents

This does *not* mutate the contents of a cons cell:

```
(define x (cons 14 null))  
(define y x)  
(set! x (cons 42 null))  
(define fourteen (car y))
```

- Like Java's `x = new Cons(42,null)`, *not* `x.car = 42`

mcons cells are mutable

Since mutable pairs are sometimes useful (will use them soon), Racket provides them too:

- **mcons**
- **mcar**
- **mcdrr**
- **mpair?**
- **set-mcar!**
- **set-mcdr!**

Run-time error to use **mcar** on a cons cell or **car** on an mcons cell