```
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

Another Closure Idiom: Currying

Currying

- Recall every ML function takes exactly one argument
- Previously encoded *n* arguments via one *n*-tuple
- Another way: Take one argument and return a function that takes another argument and…
  - Called “currying” after famous logician Haskell Curry

Example

```
val sorted3 = fn x => fn y => fn z =>
                  z >= y andalso y >= x

val t1 = ((sorted3 7) 9) 11
```

- Calling `(sorted3 7)` returns a closure with:
  - Code `fn y => fn z => z >= y andalso y >= x`
  - Environment maps `x` to `7`
- Calling *that* closure with `9` returns a closure with:
  - Code `fn z => z >= y andalso y >= x`
  - Environment maps `x` to `7`, `y` to `9`
- Calling *that* closure with `11` returns `true`

Syntactic sugar, part 1

```
val sorted3 = fn x => fn y => fn z =>
                  z >= y andalso y >= x

val t1 = ((sorted3 7) 9) 11
```

- In general, **`e1 e2 e3 e4 …`**,
  means **`(…((e1 e2) e3) e4)`**
- So instead of **`((sorted3 7) 9) 11`**,
  can just write **`sorted3 7 9 11`**
- Callers can just think "multi-argument function with spaces instead of a tuple expression"
  - Different than tupling; caller and callee must use same technique

Syntactic sugar, part 2

```
val sorted3 = fn x => fn y => fn z =>
                  z >= y andalso y >= x

val t1 = ((sorted3 7) 9) 11
```

- In general, **`fun f p1 p2 p3 … = e`**,
  means **`fun f p1 = fn p2 => fn p3 => … => e`**
- So instead of **`val sorted3 = fn x => fn y => fn z => …`**
  or **`fun sorted3 x = fn y => fn z => …`**,
  can just write **`fun sorted3 x y z = x >=y andalso y >= x`**
- Callees can just think "multi-argument function with spaces instead of a tuple pattern"
  - Different than tupling; caller and callee must use same technique

Final version

```
fun sorted3 x y z = z >= y andalso y >= x

val t1 = sorted3 7 9 11
```

As elegant syntactic sugar (even fewer characters than tupling) for:

```
val sorted3 = fn x => fn y => fn z =>
                  z >= y andalso y >= x

val t1 = ((sorted3 7) 9) 11
```

Curried fold

A more useful example and a call to it

- Will improve call next

```
fun fold f acc xs =
    case xs of
      []      => acc
    | x::xs' => fold f (f(acc,x)) xs'

fun sum xs = fold (fn (x,y) => x+y) 0 xs
```

Note: **`foldl`** in ML standard-library has **`f`** take arguments in opposite order