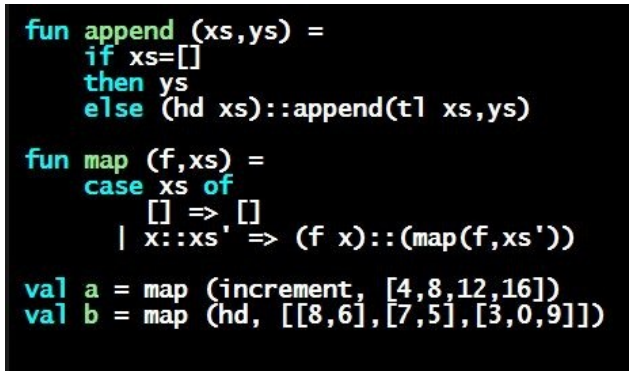

```
fun append (xs,ys) =
    if xs=[]
    then ys
    else (hd xs)::append(tl xs,ys)

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

Functions As Arguments

Functions as arguments

- We can pass one function as an argument to another function
  - Not a new feature, just never thought to do it before

```
fun f (g,…) = … g (…) …
fun h1 … = …
fun h2 … = …
…   f(h1,…) … f(h2,…) …
```

- Elegant strategy for factoring out common code
  - Replace *N* similar functions with calls to 1 function where you pass in *N* different (short) functions as arguments

[See the code file for this segment]

Example

Can reuse **`n_times`** rather than defining many similar functions

- Computes **`f(f(…f(x)))`** where number of calls is **`n`**

```
fun n_times (f,n,x) =
   if n=0
   then x
   else f (n_times(f,n-1,x))

fun double x = x + x
fun increment x = x + 1
val x1 = n_times(double,4,7)
val x2 = n_times(increment,4,7)
val x3 = n_times(tl,2,[4,8,12,16])

fun double_n_times (n,x) = n_times(double,n,x)
fun nth_tail (n,x) = n_times(tl,n,x)
```