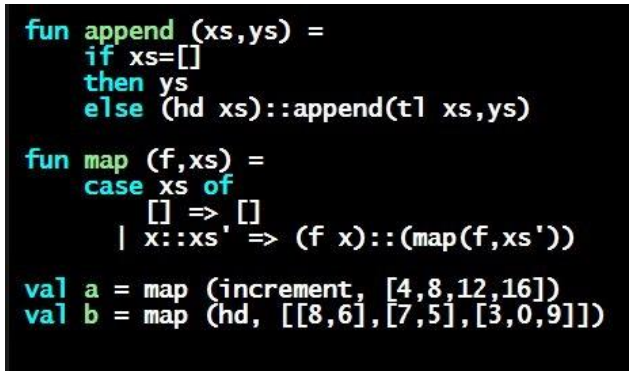

```
fun append (xs,ys) =
    if xs=[]
    then ys
    else (hd xs)::append(tl xs,ys)

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

Optional: Abstract Data Types with Closures

Implementing an ADT

As our last idiom, closures can implement abstract data types

- Can put multiple functions in a record
- The functions can share the same private data
- Private data can be mutable or immutable
- Feels a lot like objects, emphasizing that OOP and functional programming have some deep similarities

See code for an implementation of immutable integer sets with operations *insert*, *member*, and *size*

The actual code is advanced/clever/tricky, but has no new features

- Combines lexical scope, datatypes, records, closures, etc.
- Client use is not so tricky