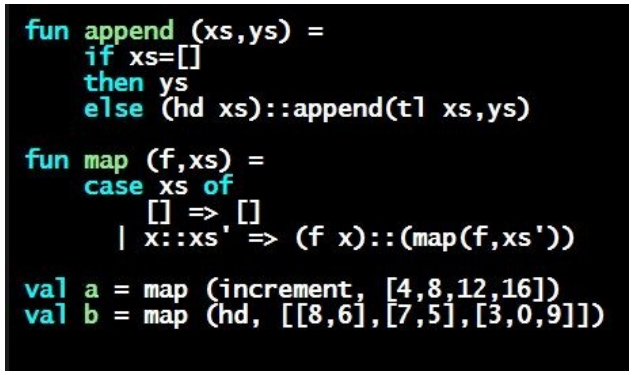

```
fun append (xs,ys) =
    if xs=[]
    then ys
    else (hd xs)::append(tl xs,ys)

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

Nested Patterns

Nested patterns

- We can nest patterns as deep as we want
  - Just like we can nest expressions as deep as we want
  - Often avoids hard-to-read, wordy nested case expressions

- So the full meaning of pattern-matching is to compare a pattern against a value for the "same shape" and bind variables to the "right parts"
  - More precise recursive definition coming after examples

Useful example: zip/unzip 3 lists

```
fun zip3 lists =
    case lists of
          ([],[],[]) => []
        | (hd1::tl1,hd2::tl2,hd3::tl3) =>
              (hd1,hd2,hd3)::zip3(tl1,tl2,tl3)
        | _ => raise ListLengthMismatch

fun unzip3 triples =
    case triples of
          [] => ([],[],[])
        | (a,b,c)::tl =>
            let val (l1, l2, l3) = unzip3 tl
            in
                (a::l1,b::l2,c::l3)
            end
```

More examples to come (see code files)