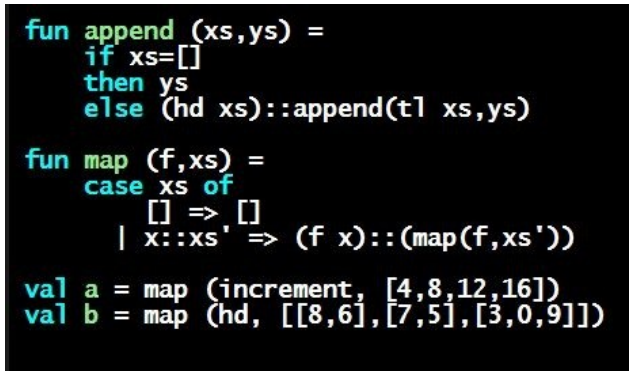

```
fun append (xs,ys) =
    if xs=[]
    then ys
    else (hd xs)::append(tl xs,ys)

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

Signature Matching

Signature matching

Have so far relied on an informal notion of, "does a module type-check given a signature?" As usual, there are precise rules…

`structure Foo :> BAR` is allowed if:

- Every non-abstract type in **`BAR`** is provided in **`Foo`**, as specified
- Every abstract type in **`BAR`** is provided in **`Foo`** in some way
  - Can be a datatype or a type synonym
- Every val-binding in **`BAR`** is provided in **`Foo`**, possibly with a *more general* and/or *less abstract* internal type
  - Discussed "more general types" earlier in course
  - Will see example soon
- Every exception in **`BAR`** is provided in **`Foo`**

Of course **`Foo`** can have more bindings (implicit in above rules)