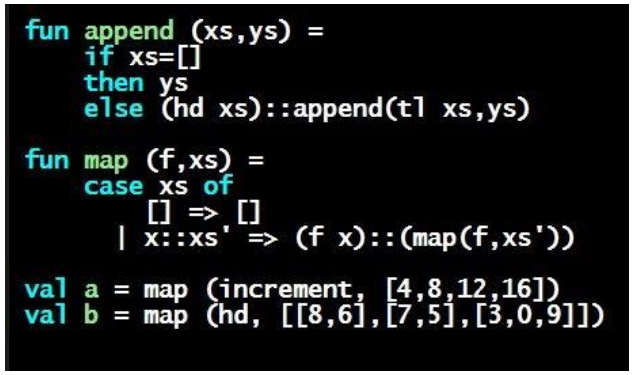

```
fun append (xs,ys) =
    if xs=[]
    then ys
    else (hd xs)::append(tl xs,ys)

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
University of Washington

Part C Overview

Where we've been…

Part A:

1. Basics, functions, recursion, scope, variables, tuples, lists, …
2. Datatypes, pattern-matching, tail recursion
3. First-class functions, closures [and course motivation!]
4. Type inference, modules, equivalence

Part B:

5. Dynamic types, parentheses, delayed evaluation, streams, macros
6. Structs, interpreters, closures
7. Static checking, static vs. dynamic

Overall: Functional programming, types or lack thereof, recursion, interpreters, …

Part C

8. Dynamically-typed Object-Oriented Programming
   Ruby basics, arrays, blocks, classes, methods, much more, …
   - "Even more dynamic" than Racket
   - "Pure" OOP
   - Blocks that are "almost" closures

9. OOP vs. Functional decomposition (aha moment (!))
   Advanced OOP topics (e.g., mixins, double dispatch)

10. Subtyping; Generics vs. Subtyping