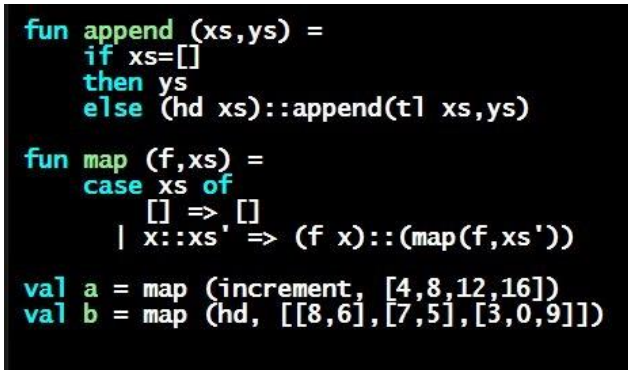

```
fun append (xs,ys) =
    if xs=[]
    then ys
    else (hd xs)::append(tl xs,ys)

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: Why Functional Languages?

Functional Programming

Why spend 60-80% of course using *functional languages*:

- Mutation is discouraged
- Higher-order functions are very convenient
- One-of types via constructs like datatypes

Because:

1. These features are invaluable for correct, elegant, efficient software (great way to think about computation)
2. Functional languages have always been ahead of their time
3. Functional languages well-suited to where computing is going

Most of course is on (1), so a few minutes on (2) and (3) …

Ahead of their time

All these were dismissed as “beautiful, worthless, slow things PL professors make you learn”

- Garbage collection (Java didn’t exist in 1995, PL courses did)
- Generics (**`List<T>`** in Java, C#), much more like SML than C++
- XML for universal data representation (like Racket/Scheme/LISP/…)
- Higher-order functions (Ruby, Javascript, C#, …)
- Type inference (C#, Scala, …)
- Recursion (a big fight in 1960 about this – I’m told ☺)
- …

The future may resemble the past

Somehow nobody notices we are right… 20 years later

- “To conquer” versus “to assimilate”
- Societal progress takes time and muddles “taking credit”
- Maybe pattern-matching, currying, hygienic macros, etc. will be next

Recent-ish Surge, Part 1

Other popular functional PLs (alphabetized, pardon omissions)

- Clojure http://clojure.org
- Erlang http://www.erlang.org
- F# http://tryfsharp.org
- Haskell http://www.haskell.org
- OCaml http://ocaml.org
- Scala http://www.scala-lang.org

Some “industry users” lists (surely more exist):

- http://www.haskell.org/haskellwiki/Haskell_in_industry
- http://ocaml.org/companies.html
- In general, see http://cufp.org

Recent-ish Surge, Part 2

Popular adoption of concepts:

- C#, LINQ (closures, type inference, …)
- Java 8 (closures)
- MapReduce / Hadoop
  - Avoiding side-effects essential for fault-tolerance here
- …

Why a surge?

My best *guesses*:

- Concise, elegant, productive programming
- JavaScript, Python, Ruby helped break the Java/C/C++ hegemony
- Avoiding mutation is *the* easiest way to make concurrent and parallel programming easier
  - In general, to handle sharing in complex systems
- Sure, functional programming is still a small niche, but there is so much software in the world today even niches have room