

```
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

*Optional: Function Patterns*

# *Yet more pattern-matching*

[Your instructor has never preferred this style, but others like it and you are welcome to use it]

```
datatype exp = Constant of int
              | Negate    of exp
              | Add       of exp * exp
              | Multiply  of exp * exp

fun eval (Constant i) = i
  | eval (Add(e1,e2)) = (eval e1) + (eval e2)
  | eval (Negate e1)  = ~ (eval e1)
  | eval (Multiply(e1,e2)) = (eval e1) + (eval e2)
```

# *Nothing more powerful*

In general

```
fun f x =  
  case x of  
    p1 => e1  
  | p2 => e2  
  ...
```

Can be written as

```
fun f p1 = e1  
  | f p2 = e2  
  ...  
  | f pn = en
```

If you prefer (assuming **x** is not used in any branch)