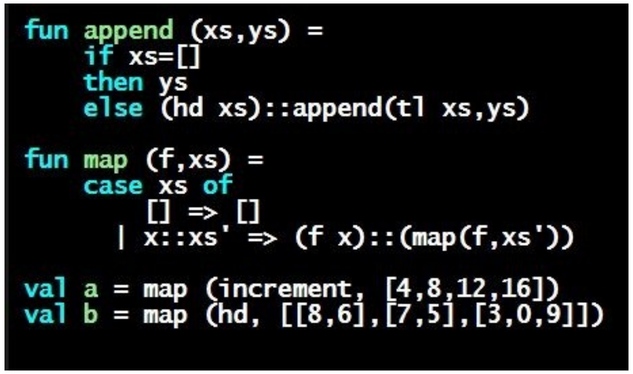

```
fun append (xs,ys) =
    if xs=[]
    then ys
    else (hd xs)::append(tl xs,ys)

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

ML Rules for Expressions (Seen So Far)

A very simple ML program

This program has integers, variables, addition, if-expressions, less-than, subtraction, and calling a pre-defined function

```
(* My first ML program *)

val x = 34;

val y = 17;

val z = (x + y) + (y + 2);

val q = z + 1;

val abs_of_z = if z < 0 then 0 - z else z;

val abs_of_z_simpler = abs z
```

Expressions

- We have seen many kinds of expressions:

  `34   true    false    x` ***`e1+e2   e1<e2`***

  `if` ***`e1`*** **`then`** ***`e2`*** **`else`** ***`e3`***
- Can get arbitrarily large since any subexpression can contain subsubexpressions, etc.

- Every kind of expression has
  1. Syntax
  2. Type-checking rules
     - Produces a type or fails (with a bad error message ☹)
     - Types so far: **`int   bool   unit`**
  3. Evaluation rules (used only on things that type-check)
     - Produces a value (or exception or infinite-loop)

Variables

- Syntax:
  sequence of letters, digits, _, not starting with digit
- Type-checking:
  Look up type in current static environment
  - If not there, fail
- Evaluation:
  Look up value in current dynamic environment

Addition

- Syntax:

  `e1 + e2` where ***`e1`*** and ***`e2`*** are expressions

- Type-checking:

  If ***`e1`*** and ***`e2`*** have type **`int`**,

  then ***`e1 + e2`*** has type **`int`**

- Evaluation:

  If ***`e1`*** evaluates to **`v1`** and ***`e2`*** evaluates to **`v2`**,

  then ***`e1 + e2`*** evaluates to sum of **`v1`** and **`v2`**

Values

- All values are expressions
- Not all expressions are values
- Every value "evaluates to itself" in "zero steps"
- Examples:
  - **`34`**, **`17`**, **`42`** have type **`int`**
  - **`true`**, **`false`** have type **`bool`**
  - **`()`** has type **`unit`**

A slightly tougher one

What are the syntax, typing rules, and evaluation rules for conditional expressions?

Let’s write it out…

Now you try one

Syntax, type-checking rules, and evaluation rules for less-than comparisons?