PAUL G. ALLEN SCHOOL
OF COMPUTER SCIENCE & ENGINEERING

CSE 341
Section 4

Winter 2018

With thanks to Alexander Lent, Nick Mooney, Spencer Pearson

Today's Agenda

- Standard-Library Docs
- More Currying and Higher Order Functions
- Mutual Recursion

Standard Basis Documentation

Online Documentation

http://www.standardml.org/Basis/index.html

http://www.smlnj.org/doc/smlnj-lib/Manual/toc.html

Helpful Subset

Top-Level	http://www.standardml.org/Basis/top-level-chapter.html
List	http://www.standardml.org/Basis/list.html
ListPair	http://www.standardml.org/Basis/list-pair.html
Real	http://www.standardml.org/Basis/real.html
String	http://www.standardml.org/Basis/string.html

Higher-Order Functions Review

- A function that returns a function or takes a function as an argument.

Canonical Examples

- `map : ('a -> 'b) * 'a list -> 'b list`
  - Applies a function to every element of a list and return a list of the resulting values.
  - Example: `map (fn x => x*3, [1,2,3]) === [3,6,9]`
- `filter : ('a -> bool) * 'a list -> 'a list`
  - Returns the list of elements from the original list that, when a predicate function is applied, result in true.
  - Example: `filter (fn x => x>2, [~5,3,2,5]) === [3,5]`

Note: List.map and List.filter are similarly defined in SML but use currying.

Higher-Order Functions Review

- `foldl: (f: 'a*'b->'b) (acc: 'b) (l: 'a list) -> 'b`
  - f(l_n, f(…, (f(l_2, f(l_1, acc))))
  - Apply function to the current element and the accumulator as soon as possible
- `foldr: (f: 'a*'b->'b) (acc: 'b) (l: 'a list) -> 'b`
  - f(l_1, f(l_2, f(…, f(l_n, acc))))
  - Wait until the rest of the list has been evaluated and then apply function to the current element and result from rest of the list
- We've written `foldl` in lecture, write `foldr`

Broader Idea

Functions are Awesome!

- SML functions can be passed around like any other value.
- They can be passed as function arguments, returned, and even stored in data structures or variables.
- Functions like `map` are very pervasive in functional languages.
  - A function like `map` can even be written for other data structures such as trees.

Currying and High Order Functions

- Some functions from standard library:
  - List.map
  - List.filter
  - List.foldl
  - List.foldr

- Write our own higher order functions
  - Alternating 0 and 1

Mutual Recursion

- What if we need function f to call g, and function g to call f?
- This is a common idiom

```
fun earlier x =
      ...
      later x
      ...
fun later x =
      ...
      earlier x
      ...
```

Unfortunately this does not work ☹

Mutual Recursion Workaround

- We can use higher order functions to get this working
- It works, but there has got to be a better way!

```
fun earlier f x =
      ...
      f x
      ...
fun later x =
      ...
      earlier later x
      ...
```

Mutual Recursion with **and**

- SML has a keyword for that
- Works with mutually recursive **datatype** bindings too

```
fun earlier x =
      ...
      later x
      ...
and later x =
      ...
      earlier x
      ...
```