CSE 341 AA: Section 4

Porter Jones
pbjones@cs.washington.edu
Office Hours: Wednesdays 8:30 - 10:30am
Mutual Recursion

- What if we need function f to call g, and function g to call f?
- The attempt below doesn’t work.... :(

```haskell
fun earlier x =
  ...
  later x
  ...
fun later x =
  ...
  earlier x
```
One possible solution:

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

A more pleasant solution that uses special SML syntax:

```ml
fun earlier x =
  ...
  later x
  ...
and later x =
  ...
  earlier x
  ...
```
Module System

- Good for organizing code, and managing namespaces (useful, relevant)
- Good for maintaining invariants (interesting)
- Good for data hiding (useful)
sub not curried

fun sub (x, y) = x - y
sub curried

fun sub x y = x - y
SML List structure functions

- All of these take curried arguments!!!
  - List.map
  - List.filter
  - List.foldl
  - List.foldr
Tangent: foldl and foldr

List.foldl (fn (x, acc) => x+acc) 0 xs  List.foldr (fn (x, acc) => x+acc) 0 xs

0 [1, 2, 3, 4, 5]  [1, 2, 3, 4, 5] 0