CS52: Recursion Patterns

David Kauchak

Fall 2022

Numeric recursion

```
fun numrec n = if n < 0 then
                   negative_case – possibly an exception
               else if n = 0 then
                   base_case
               else
                   expression_involving_recursive_call;
```

Examples:
factorial
sumList (from assignment 0)

Numeric recursion + baggage

```
fun numrec m n = if n < 0 then
                    negative_case – possibly an exception
                 else if n = 0 then
                    base_case
                 else
                    expression_involving_m_and_recursive_call;
```

Some times we need additional information, *but the recursion is still just on one of the numbers.*

Examples:
- power
- interval
- interval2

Simple list recursion

```
fun listrec []        = base case
  | listrec (x::xs) =
      expression_involving_(listrec xs);
```

Examples:

appendAll	myLength
sumList (from lecture)	cubeAll
rev (version 1.0)	uniquify
	myAppend

Simple list recursion + baggage

```
fun listrec y []        = base case
  | listrec y (x::xs) =
        expression_involving_y_and_(listrec xs);
```

Some times we need additional information, *but the recursion is still just on the list.*

Examples:

lessThanList	member
myFilter	funPairs
map	consAll

Simultaneous list recursion

```
fun sumulrec []       _       = base case
  | sumulrec _        []      = base case2
  | sumulrec (x::xs) (y::ys) =
     expression_involving_(smulrec xs ys);
```

Some times we need additional information, *but the recursion is still just on the list.*

Examples:

myZip

Recursion with an accumulator

```
fun accumrec acc []       = expression_involving_acc
  | accumrec acc (x::xs) =
      expression_with_recursive_call_x_xs_and_acc
      (acc should be “added to” in the call)
```

Examples:
 revAux
 addAllAux (from Intro to SML)