

```
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

Exceptions

Exceptions

An exception binding introduces a new kind of exception

```
exception MyFirstException  
exception MySecondException of int * int
```

The `raise` primitive raises (a.k.a. throws) an exception

```
raise MyFirstException  
raise (MySecondException (7,9))
```

A handle expression can handle (a.k.a. catch) an exception

- If doesn't match, exception continues to propagate

```
e1 handle MyFirstException => e2  
e1 handle MySecondException(x,y) => e2
```

Actually...

Exceptions are a lot like datatype constructors...

- Declaring an exception adds a constructor for type **exn**
- Can pass values of **exn** anywhere (e.g., function arguments)
 - Not too common to do this but can be useful
- Handle can have multiple branches with patterns for type **exn**