Recursion and Induction
A recursive function

(* power : (int * int) -> int
   REQUIRES: k >= 0
   ENSURES: power(n,k) ==> n^k, with 0^0 = 1 *
)

fun power (_:int, 0:int) : int = 1
  | power (n:int, k:int) : int = n * power(n, k-1)
Today’s goals

• Write simple recursive functions

• Use induction to prove properties of functions
  • Simple (standard, mathematical)
  • Strong (complete)
**Theorem:** For all integer values $n$, for all integer values $k \geq 0$, $\text{power}(n, k)$ evaluates to $n^k$.

**Proof:** By mathematical (standard, simple) induction on $??$.
fun power (_:int, 0:int) : int = 1
| power (n:int, k:int) : int = n * power(n, k-1)

**Theorem:** For all integer values $n$, for all integer values $k \geq 0$, $\text{power}(n, k)$ evaluates to $n^k$.

**Proof:** By mathematical (standard, simple) induction on $k$. 
Simple induction

• To prove a property of the form
  \[ P(m), \text{ for all non-negative integers } m \]
• First, prove \( P(0) \).  \textit{base case}
• Then show that, for all \( k \geq 0 \),
  \[ P(k+1) \text{ follows logically from } P(k). \]  \textit{inductive step}
Why it works

- $P(0)$ gets a direct proof
- $P(1)$ follows from $P(0)$
- $P(2)$ follows from $P(1)$
- Similarly, for each $m \geq 0$ we can show $P(m)$
  - for $k > 0$, at the $k^{\text{th}}$ step we’ve already shown $P(k)$, so $P(k+1)$ follows logically
More efficient \texttt{power}

\begin{verbatim}
(* even : \texttt{int} -> \texttt{bool}  
  \texttt{REQUIRES: true }  
  \texttt{ENSURES: even(k) ==> true if k is even}  
  \texttt{==> false if k is odd.}  
*)

\textbf{fun} even \ (k:int) \ : \texttt{bool} = ((k mod 2) = 0)

(* square : \texttt{int} -> \texttt{int}  
  \texttt{REQUIRES: true}  
  \texttt{ENSURES: square(n) ==> n^2}  
*)

\textbf{fun} square \ (n:int) \ : \texttt{int} = n * n

(* powere : (\texttt{int} * \texttt{int}) -> \texttt{int}  
  \texttt{REQUIRES: k \geq 0}  
  \texttt{ENSURES: powere(n,k) ==> n^k, with 0^0 = 1.}  
  \texttt{powere computes} \ n^k \texttt{using O(log(k)) multiplies.}  
*)

\end{verbatim}
(* even : int -> bool  
  REQUIRES: true  
  ENSURES: even(k) returns true if k is even  
           returns false if k is odd.  
*)

fun even (k:int) : bool = ((k mod 2) = 0)

(* square : int -> int  
  REQUIRES: true  
  ENSURES: square(n) ==> n^2  
*)

fun square (n:int) : int = n * n

(* powere : (int * int) -> int  
  REQUIRES: k >= 0  
  ENSURES: powere(n, k) ==> n^k, with 0^0 = 1.  
  powere computes n^k using O(log(k)) multiplies.  
*)

fun powere (_:int, 0:int) : int = 1
| powere (n:int, k:int) : int =
  case even(k) of
    true  => square(powere(n, k div 2))
  | false => n * powere(n, k-1)
Theorem: For all integer values \( n \), for all integer values \( k \geq 0 \), \( \text{powere}(n, k) \) evaluates to \( n^k \).

Proof: By ???

```ocaml
fun powere ((_:int, 0:int) : int = 1
    | powere (n:int, k:int) : int =
      case even(k) of
        true  => square(powere(n, k div 2))
        false => n * powere(n, k-1)
```

**Theorem:** For all integer values $n$, for all integer values $k \geq 0$, $\text{powere}(n, k)$ evaluates to $n^k$.

**Proof:** By strong (complete) induction on $k$. 

```haskell
fun powere (_:int, 0:int) : int = 1
| powere (n:int, k:int) : int =
  case even(k) of
  true  => square(powere(n, k div 2))
| false => n * powere(n, k-1)
```
Strong (complete) induction

- To prove a property of the form

  \[ P(m), \text{ for all non-negative integers } m \]

- Show that, for all \( k \geq 0 \), \( P(k) \) follows logically from \( \{P(0), \ldots, P(k-1)\} \).
So far

• Simple and strong induction
• Examples of their use
• Just the beginning…

Next

• Examples with lists
• Structural induction
A list of integers is either

- 

- \[\text{nil}\]

- \(x :: xs\) where \(x: \text{int}\) and \(xs: \text{int list}\)

pronounced “cons"
(* length : int list -> int
   REQUIRES: true
   ENSURES: length(L) returns the number of elements in L
   *)

fun length([] : int list) : int = 0
  | length(x::xs) = 1 + length(xs)
**Theorem**: For all values $L : \text{int list}$, $\text{length}(L)$ evaluates to an integer value.
Structural induction for lists

To show that $P(L)$ for all values $L$ of type $t\ list$ one needs to show

- $P([],)$
- If the property holds for a value $L'$ of type $t\ list$, then it also holds for $\mathsf{v} :: L'$