(* lec11 in class *)

(* f : int -> int *)
fun f x =
  if x > 3 (* x : int *)
  then 42 (* returns an int *)
  else x * 2 (* consistent! *)

(* g : ERROR *)
(* fun g x =    if x > 3    then true     else x * 2 *)

(* x : int *)
val x = 42

(* f : bool * int * 'a -> int *)
fun f (y, z, w) =
  if y
  then z + x
  else 0

(* z : int *)
fun f x =
  let
    val (y,z) = x
  in
    (abs y) + z
  end

(* sum : T1 -> T2 *)
fun sum xs =
  case xs
  of [] => 0
      | x::xs' => x + (sum xs')

fun broken_sum xs =
  case xs
  of [] => 0.0
      | x::xs' => x + (sum xs')

(* also borked
  fun broken_sum xs =    case xs of      [] => 0      | x::xs' => x + (broken_sum x) *)
fun length xs =  
  case xs of  
    [] ⇒ 0  
  | x::xs' ⇒ 1 + (length xs')

fun f (x,y,z) =  
  if true  
  then (x,y,z)  
  else (y,x,z)

fun compose (f,g) =  
  λ x ⇒ f (g x)

(*
val x = ref 42
val y = ref 42
val z = x
val _ = x := 43
val w = (!y) + (!z) (* 85 *)
Foo x = new Foo();
Foo y = new Foo();
x.myInt = 42;
y.myInt = 42;
Foo z = x;
x.myInt = 43;
print(y.myInt + z.myInt)
x*)