exception ListLengthMismatch

fun old_zip3 (l1,l2,l3) =
  case (sign x1,sign x2)
  of (Z,_) ⇒
    Z
  | (_,Z) ⇒
    Z
  | (P,P) ⇒
    P
  | (N,N) ⇒
    P
  else
    raise ListLengthMismatch

fun shallow_zip3 (l1,l2,l3) =
  case
    xs
  of
case
    l1
  of
    [] ⇒
      []
  | _ :: xs' ⇒
    1 + len xs'
  else
    raise ListLengthMismatch

fun len xs =
  case
    xs
  of
    [] ⇒
      0
  | _ :: xs' ⇒
    1 + len xs'

fun zip3 list_triple =
  case
    list_triple
  of
    ([],[],[]) ⇒
      1
  | _ ⇒
    raise ListLengthMismatch

fun unzip3 lst =
  case
    lst
  of
    [] ⇒
      ([],[],[])
  | (a,b,c) :: t1 ⇒
    let
      val (head,neck,rest) = unzip3 t1
    in
      (a :: head, b :: neck, c :: rest)
    end

fun cumulative_sum xs =
  case
    xs
  of
    [] ⇒
      xs
  | x :: xs' ⇒
    let
      datatype sgn = P | N | Z
      fun multisign (x1,x2) =
        let
          fun sign x =
            if x=0 then Z else
              if x>0 then P else N
        in
          case
            (sign x1,sign x2)
          of
            (Z,_) ⇒
              Z
          | (_ Z) ⇒
              Z
          | (P,P) ⇒
              P
          | (N,N) ⇒
              P
          else
            raise ListLengthMismatch
        in
          case
            (sign x,sign x')
          of
            (Z,_) ⇒
              Z
          | (_ Z) ⇒
              Z
          | (P,P) ⇒
              P
          | (N,N) ⇒
              P
          else
            raise ListLengthMismatch
          end
        end

val x = maxlist ([3,4,5],MyUndesirableCondition)
val y = maxlist ([3,4,5],MyUndesirableCondition)
val w = maxlist ([3,4,5],MyUndesirableCondition)
fun sum1 xs = 
  case xs of 
    [] ⇒ 0 
    | i::xs' ⇒ i + sum1 xs'

fun sum2 xs = 
  let fun f (xs, acc) = 
    case xs of 
      [] ⇒ acc 
      | i::xs' ⇒ f(xs', i+acc)
    in 
      f(xs, 0) 
  end

fun rev1 xs = 
  case xs of 
    [] ⇒ [] 
    | x::xs' ⇒ (rev1 xs') @ [x]

fun rev2 xs = 
  let fun aux(xs, acc) = 
    case xs of 
      [] ⇒ acc 
      | x::xs' ⇒ aux(xs', x::acc)
    in 
      aux(xs, [])