



Programmierung: Musterlösung zum 6. Übungsblatt

Prof. Gert Smolka und Thorsten Brunklaus

Aufgabe 6.1: Binärbäume (1+2+2+2)

```
fun root (L x)      = x
| root (N(x, _, _)) = x

fun inner' (L _)      = nil
| inner' (N(x, t, t')) = [x] @ inner' t @ inner' t'

fun inner (L _)      = nil
| inner (N(x, t, t')) = inner' t @ inner' t'

fun mapTree f (L x)      = L(f x)
| mapTree f (N(x,t,t')) = N(f x, mapTree f t, mapTree f t')

val double = mapTree (fn x => 2*x)
```

Aufgabe 6.2: Symbolisches Differenzieren (2+4+4+2)

```
val u = Add(Pow(X,3),
            Add(Mul(Con 3, Pow(X,2)),
                Add(X, Con 2)))

fun derive (Con n)      = Con 0
| derive X              = Con 1
| derive (Add(u,v))     = Add(derive u, derive v)
| derive (Mul(u,v))     = Add(Mul(derive u, v), Mul(u, derive v))
| derive (Pow(u,n))     = Mul(Con n, Mul(Pow(u,n-1), derive u))

fun simplify1 (Add(Con 0, u)) = u
| simplify1 (Add(u, Con 0)) = u
| simplify1 (Mul(Con 0, u)) = Con 0
| simplify1 (Mul(u, Con 0)) = Con 0
| simplify1 (Mul(Con 1, u)) = u
| simplify1 (Mul(u, Con 1)) = u
| simplify1 (Pow(u, 0))    = Con 1
| simplify1 (Pow(u, 1))    = u
| simplify1 u              = u

fun simplify (Add(u,v)) = simplify1(Add(simplify u, simplify v))
| simplify (Mul(u,v))  = simplify1(Mul(simplify u, simplify v))
| simplify (Pow(u,n))  = simplify1(Pow(simplify u, n))
| simplify u           = u
```

Aufgabe 6.3: Dreifach-Partition (3+3)

```

fun partition f =
  let
    fun insert (x, (us, vs, ws)) =
      case f x of
        LESS      => (x::us, vs, ws)
      | EQUAL     => (us, x::vs, ws)
      | GREATER   => (us, vs, x::ws)
  in
    foldl insert (nil,nil,nil)
  end

fun myPartition (m,n) =
  partition (fn x => if x<m then LESS
                    else if x>n then GREATER
                    else EQUAL)

```

Aufgabe 6.4: Optionen (4)

```

fun find f (L x)      = if f x then SOME x else NONE
  | find f (N(x,t,t')) = if f x then SOME x
                        else case find f t of
                              NONE => find f t'
                              | x    => x

```

Aufgabe 6.5: Schneller Test auf Doppelaufreten (0+3+3+3)

```

fun test xs =
  let
    exception Double
    fun order (x,y) = if x< y then LESS
                      else if x>y then GREATER
                      else raise Double
    val dsort = Listsort.sort order
  in
    (dsort xs; false) handle Double => true
  end

```

Aufgabe 6.6: Terme(2+2+2+3+3)

[illegible]

```

fun subterm t          nil      = t
  | subterm (T(_, ts)) (n::ns) = subterm (List.nth(ts,n-1)) ns

fun balanced t =
  let
    exception Unbalanced
    fun forward (d,d') = if d=d' orelse d'<0 then d else raise Unbalanced
    fun depthb (T(_, ts)) = 1 + foldl forward ~1 (map depthb ts)
  in
    (depthb t ; true) handle Unbalanced => false
  end

```