Aufgabe 1: Schnelles Reversieren (8)

```
fun reverse xs = foldl op:: nil xs
```

Aufgabe 2: Count (7+4)

```
fun count y = foldl (fn (x,n) => if x=y then n+1 else n) 0
```

$\Theta(|xs|)$.

Aufgabe 3: Until (7+7)

```
fun until p =
    let
        fun until' n = if p n then n else until'(n+1)
    in
        until' 0
    end

fn x => until (fn n => n*n > x) - 1
```

Aufgabe 4: Dezimaldarstellung (7+7)

```
fun dec x  = if x<10 then [x] else dec(x div 10) @ [x mod 10]

fun int ds = foldl (fn (d,x) => 10*x+d) 0 ds
```

Aufgabe 5: Schnelles Potenzieren (8)

```
fun potenz (x,n) = if n<1 then 1
                   else let
                            val y = potenz(x, n div 2)
                        in
                            y * y * (if n mod 2 = 0 then 1 else x)
                        end
```

Aufgabe 6: Quicksort (7+8)

```
fun partition x = foldl (fn (y, (us,vs)) =>
                         if y<x then (y::us, vs)
                         else (us, y::vs))
                        (nil,nil)

fun qsort nil     = nil
  | qsort (x::xs) = let
                      val (us,vs) = partition x xs
                    in
                      qsort vs @ [x] @ qsort us
                    end
```

Aufgabe 7: Map für Terme (8)

```
fun mapterm f (T(x,ts)) = T(f x, map (mapterm f) ts)
```

Aufgabe 8: Spiegeln von geordneten Bäumen (6+6)

```
fun mirror (N(x,t,t')) = N(x, mirror t', mirror t)
  | mirror     t       = t


fun mirror (T(x,ts)) = T(x, rev(map mirror ts))
```

Aufgabe 9: Ebenen von geordneten Bäumen (8+8)

```
fun level (t,n) =
    if n<0 then []
    else (case t of
              L x       => if n=0 then [x]
                           else []
            | N(x,t,t') => if n=0 then [x]
                           else level(t,n-1) @ level(t',n-1))


fun level (T(x,ts), n) =
    if n=0 then [x]
    else foldr (fn (t,xs) => level(t,n-1) @ xs) nil ts
```

Aufgabe 10: Schneller Test auf Doppelauftreten (12)

```
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 11: Listeninduktion (12)

Beweis Durch Induktion über *xs*.

Sei $xs = nil$. Dann

$$
\begin{aligned}
|xs@ys| &= |ys| && \text{Definition von } @ \\
&= |xs| + |ys| && \text{Definition von } |\_|
\end{aligned}
$$

Sei $xs = x :: xr$. Dann

$$
\begin{aligned}
|xs@ys| &= |x :: (xr@ys)| && \text{Definition von } @ \\
&= 1 + |xr@ys| && \text{Definition von } |\_| \\
&= 1 + |xr| + |ys| && \text{Induktionsannahme} \\
&= |xs| + |ys| && \text{Definition von } |\_|
\end{aligned}
$$

□

Aufgabe 12: Laufzeitsynthese (5+2+8+5)

```
fun tlogn n  = if n<1 then ()
               else tlogn(n div 2)

fun tn n     = if n<1 then ()
               else tn(n-1)

fun tnlogn n = if n<1 then ()
               else (tn n ; tnlogn(n div 2) ; tnlogn(n div 2))

fun tn2 n    = if n<1 then ()
             else (tn n ; tn2(n-1))
```