# Functional Programming

This exam consists of five exercises. ***Explain your answers.*** The available points for each item are written in the margin. You need at least 50 points to pass.

**1** Consider the lambda-term $t = (\lambda xyz.x\ z\ (y\ z))\ (\lambda xy.x)\ (\lambda x.x)\ (\lambda x.x)$.

[10] (a) Reduce $t$ stepwise to normal form, using the leftmost innermost strategy.

[10] (b) Reduce $t$ stepwise to normal form, using the leftmost outermost strategy.

**2** Consider the OCaml type `type 'a tree = E | N of 'a tree * 'a * 'a tree` together with the functions

```
let rec preorder = function
 | E           -> []
 | N (l, a, r) -> a :: (preorder l @ preorder r)
;;

let rec sum_tree = function
 | E           -> 0
 | N (l, a, r) -> a + (sum_tree l + sum_tree r)
;;

let rec sum = function
 | []       -> 0
 | x :: xs  -> x + (sum xs)
;;
```

Prove by induction that `sum` (`preorder` $t$) = `sum_tree` $t$ for every value $t$ of type `int tree`. You may use the equality

$$\texttt{sum}\ (xs\ \texttt{@}\ ys) = (\texttt{sum}\ xs) + (\texttt{sum}\ ys) \qquad (\star)$$

for all integer lists $xs$ and $ys$.

[ 5] (a) Base case.

[15] (b) Step case.

**3** Consider the OCaml functions `mem` and `unique`

```
let rec mem y = function
 | []      -> false
 | x :: xs -> x = y || mem y xs
;;

let rec unique = function
 | []      -> []
 | x :: xs ->
  if mem x xs then unique xs else x :: unique xs
;;
```

[10] (a) Implement a tail-recursive variant of `unique`.

[10] (b) Use tupling to implement a function `percentage: 'a -> 'a list -> float` that determines for a given element $x$ in a list $xs$ the percentage it constitutes to the full list, e.g.,

`percentage 'a' ['a';'b';'c';'a'] = 0.5`.

**4** Consider the $\lambda$-term $t = (\lambda x.y\ x)\ (\lambda yz.z\ y)\ w$.

[ 5] (a) Reduce $t$ to normal form.

[ 5] (b) Give the set $\mathcal{FV}\mathsf{ar}(t)$ of free variables of $t$.

[ 5] (c) Give the set $\mathcal{BV}\mathsf{ar}(t)$ of bound variables of $t$.

[ 5] (d) Give the set $\mathcal{S}\mathsf{ub}(t)$ of all subterms of $t$.

**5** Consider the typing environment

$$E = \{1 : \mathsf{int},\ + : \mathsf{int} \to \mathsf{int} \to \mathsf{int},\ \mathsf{p} : \mathsf{int} \to \mathsf{int} \to \mathsf{pair}(\mathsf{int}, \mathsf{int})\}.$$

[10] (a) Prove the typing judgment $E \vdash \mathbf{let}\ x = 1\ \mathbf{in}\ \mathsf{p}\ x\ (x + x) : \mathsf{pair}(\mathsf{int}, \mathsf{int})$.

[10] (b) Transform the type inference problem $E \triangleright \mathbf{let}\ x = 1\ \mathbf{in}\ \mathsf{p}\ x\ (x + x) : \alpha_0$ into a unification problem.