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 OCaml function `let s x = x * x`.

[10] (a) Evaluate the function call `s (s 10)` stepwise, using leftmost innermost reduction.

[10] (b) Evaluate the function call `s (s 10)` stepwise, using leftmost outermost reduction.

2 Consider the OCaml type `type tree = E | N of tree * tree` together with the function

```
let rec mirror = function
 | E        -> E
 | N (l, r) -> N (mirror r, mirror l)
;;
```

Prove by induction that `mirror` (`mirror` t) $= t$ for every value t of type `tree`.

[5] (a) Base case.

[15] (b) Step case.

3 Consider the OCaml functions `f` and `g`:

```
let rec f x = if x / 2 = 0 then 0 else 1 + f (x / 2);;
let rec g x = if x < 2 then 1 else g (x - 1) + 2 * g (x - 2);;
```

[10] (a) Give a tail recursive variant of `f`.

[10] (b) Use tupling to implement a more efficient variant of `g`.

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

[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

$$E = \{1 : \mathsf{int}, 2 : \mathsf{int}, \mathsf{cons} : \mathsf{int} \to \mathsf{list}(\mathsf{int}) \to \mathsf{list}(\mathsf{int}),\\ \mathsf{hd} : \mathsf{list}(\mathsf{int}) \to \mathsf{int}, \mathsf{nil} : \mathsf{list}(\mathsf{int}), \mathsf{tl} : \mathsf{list}(\mathsf{int}) \to \mathsf{list}(\mathsf{int})\}.$$

[10] (a) Prove the typing judgment $E \vdash \mathbf{let}\ x = \mathsf{tl}\ (\mathsf{cons}\ 1\ (\mathsf{cons}\ 2\ \mathsf{nil}))\ \mathbf{in}\ \mathsf{hd}\ x : \mathsf{int}$.

[10] (b) Solve the unification problem.

$$\begin{aligned} \alpha_3 \to \mathsf{list}(\alpha_3) \to \mathsf{list}(\alpha_3) &\approx \alpha_2 \to \alpha_1 \to \alpha_4;\\ \mathsf{bool} &\approx \alpha_2;\\ \mathsf{list}(\alpha_0) &\approx \alpha_1;\\ \mathsf{list}(\alpha_0) &\approx \alpha_4 \end{aligned}$$