University of Innsbruck | **Institute of Computer Science**
3rd Exam | **November 5 2010**

Functional Programming | **WS 2009/2010** | **LVA 703017**

Name: | **Matr.Nr.:**

1. Consider the λ-term $S = \lambda x\, y\, z.x\ z\ (y\ z)$

[12] (a) Reduce the term $S\ S\ S$ to normal form, using the leftmost innermost reduction strategy.

[13] (b) Reduce the term $S\ S\ S$ to normal form, using the leftmost outermost reduction strategy.

2. Consider the three OCaml functions

```
let rec take n xs = if n < 1 then [] else match xs with
  | [] -> []
  | x::xs -> x :: take (n-1) xs

let rec length = function [] -> 0
                        | _::xs -> 1 + length xs

let rec init = function x::y::xs -> x :: init (y::xs)
                      | _ -> []
```

Prove by induction that `init` xs = `take` (`length` $xs - 1$) xs for every list xs. (*Hint:* In the step case, you will need a further case distinction on the tail of the list.)

[5] (a) Base case.

[20] (b) Step case.

3. Consider the OCaml function:

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

[12] (a) Implement a tail-recursive variant of `sum`.

[13] (b) Use tupling to implement a function `average : int list -> int`, producing the same results as if defined via `average` xs = `sum` xs/`length` xs.

4. Consider the typing environment $E = \{\mathsf{true} : \mathsf{bool}\}$.

[12] (a) Use type checking to decide whether the expression **let** $x = \mathsf{true}$ **in** $x\ x$ is of type bool with respect to the environment E. Justify your answer.

[13] (b) Solve (if possible) the unification problem:

$$\alpha_1 \to \alpha_2 \to \alpha_3 \approx \alpha_4 \to (\alpha_2 \to \alpha_2) \to \alpha_5$$