University of Innsbruck | **Institute of Computer Science**
2nd Exam | **February 27, 2009**

Functional Programming | **WS 2008/2009** | **LVA 703017**

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

1. Consider the lambda-term $t = (\lambda p.p\ (\lambda xy.y))\ ((\lambda xyf.f\ x\ y)\ (\lambda x.x)\ (\lambda x.x))$.

[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.

[20] **2.** Consider the type

```
type 'a btree = Leaf of 'a | Node of ('a btree * 'a * 'a btree)
```

together with the functions

```
let hd(x::_) = x
let rec leftmost = function Leaf x      -> x
                          | Node(l,_,_) -> leftmost l
let rec flatten = function Leaf x      -> [x]
                         | Node(l,x,r) -> flatten l @ (x :: flatten r)
```

Prove by induction that `hd(flatten` t`) =` `leftmost` t for all trees t. You may use the fact

$$\texttt{hd(flatten}\ t\ \texttt{@}\ xs\texttt{)} = \texttt{hd(flatten}\ t\texttt{)} \qquad (\star)$$

for all trees t and lists xs.

3. Consider the OCaml function `replicate`, defined by:

```
let rec replicate m n = if n < 1 then [] else m :: replicate m (n-1)
```

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

[10] (b) Implement the function `split` that splits a list into two lists, where the first contains all elements satisfying the given predicate and the second all the others, e.g.,

```
split (fun x -> x <> 0) [1;2;0;3] = ([1;2;3],[0])
```

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

[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.

[10] **5.** (a) Transform the type inference problem $\varnothing \rhd \lambda x.x\ x : \alpha_0$ into a unification problem.

[10] (b) Solve the following unification problem (if possible).

$$\begin{aligned} \alpha_1 &\approx \alpha_3 \to \alpha_2 \\ \alpha_1 &\approx \alpha_3 \\ \alpha_0 &\approx \alpha_1 \to \alpha_2 \end{aligned}$$