universität innsbruck

Functional Programming	WS 2018/2019	LVA 703024

EXAM 1 — February 1, 2019

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

[25] **1** Consider the Haskell functions:

```
length :: [a] -> Int
length [] = 0
length (x:xs) = 1 + length xs

filter :: (a -> Bool) -> [a] -> [a]
filter p [] = []
filter p (x:xs) | p x       = x : filter p xs
                | otherwise = filter p xs

countLess :: Int -> [Int] -> Int
countLess y xs = length $ filter (< y) xs
```

[9] (a) Use equational reasoning to evaluate `countLess 3 [1,2,3,1]`.

[6] (b) Determine whether `length` and `filter` are tail recursive and guardedly recursive, respectively.

[10] (c) Give a tail recursive variant of `countLess` that considers each element of `xs` only once.

[25] **2** Consider the two Haskell functions:

```
[] ++ ys      =  ys
(x:xs) ++ ys  =  x : (xs ++ ys)

all p []      = True
all p (x:xs)  = p x && all p xs
```

Prove by induction that `all` p $(xs$ `++` $ys) =$ `all` p xs `&&` `all` p ys for all p, xs, and ys. Apart from the defining equations above, you may assume every property that holds for logical conjunction also for `&&`.

[6] (a) Prove the base case, applying a single equation at a time.

[9] (b) State the induction hypothesis and the statement you have to show in the step case.

[10] (c) Prove the step case, applying a single equation at a time.

[25] **3** Consider the λ-term $T = ((\lambda x.\,(x\ x))\ (\lambda y.\,(\lambda z.\,(y\ z))))$.

[4] (a) Write T as compactly as possible, using the notational conventions (for omitting parentheses etc.).

[10] (b) Stepwise β-reduce T to β-normal form, giving each β-reduction step, and indicate for each step whether α-renaming is needed to avoid variable capture.

[6] (c) Give one λ-term in *weak head normal form* and one that is not. In each case, justify your answer.

[5] (d) Give a λ-term on which *applicative order reduction* does not terminate, but *normal order reduction* does.

[25] **4** Consider the typing environment $E = P \cup \{\texttt{filter} :: (\alpha \to \mathsf{Bool}) \to \mathsf{List}(\alpha) \to \mathsf{List}(\alpha)\}$.

[8] (a) Use type checking to prove the typing judgment $E \vdash \textbf{if}\ (\lambda x.\,\mathsf{True})\ 1\ \textbf{then}\ 1\ \textbf{else}\ 0 :: \mathsf{Int}$.

[7] (b) Compute the mgu for the unification problem $\mathsf{Bool} \to \mathsf{Bool} \approx \mathsf{Bool} \to \alpha_2; \mathsf{Bool} \to \alpha_0 \approx \alpha_2 \to \alpha_1 \to \alpha_2$ if it exists.

[10] (c) Use the typing constraint rules to transform $E \rhd \texttt{filter}\ (\lambda y.\,1) :: \alpha_0$ into a unification problem.