

Prof. Dr. Jürgen Giesl  
Peter Schneider-Kamp  
Stephan Swiderski

## Exercises *Functional Programming* – Sheet 9

Solutions will be collected until Wednesday, June 20, 2007 in the exercise course.

Exercises can be solved both in English and in German.

### Exercise 1 ( 2 + 2 + 2 + 2 points)

We define the following algebraic data types and the following functions.

```
data Nats = Zero | Succ Nats
data List a = Nil | Cons a (List a)
```

```
pred :: Nats -> Nats
pred (Succ x) = x
```

```
isZero :: Nats -> Bool
isZero Zero = True
isZero _    = False
```

```
head :: List a -> a
head (Cons x _) = x
```

```
tail :: List a -> List a
tail (Cons _ xs) = xs
```

```
isNil :: List a -> Bool
isNil Nil = True
isNil _   = False
```

```
fst :: (a,b) -> a
fst (x,y) = x
```

```
snd :: (a,b) -> b
snd (x,y) = y
```

Give simple Haskell-programs that are equivalent to the following Haskell-programs. Your solutions may use some of the functions defined above. You do not have to use the transformation rules which will be presented later in the lecture.

- (a) `mult :: List Int -> Int`  
`mult = \ys -> case ys of`  
     `Nil                   -> 1`  
     `(Cons x xs) -> x * mult xs`
- (b) `take' :: Nats -> List a -> List a`  
`take' _           Nil           = Nil`  
`take' Zero       _           = Nil`  
`take' (Succ n) (Cons x xs) = Cons x (take' n xs)`
- (c) `take' :: Int -> List a -> List a`  
`take' _           Nil           = Nil`  
`take' 0           _           = Nil`  
`take' (n+1)      (Cons x xs) = Cons x (take' n xs)`
- (d) `flatList :: List (a,a) -> List a`  
`flatList (Cons (x,y) xs) = Cons x (Cons y (flatList xs))`  
`flatList Nil               = Nil`

## Exercise 2 ( 2 + 3 + 4 points)

For the following Haskell expressions give the value of  $\mathcal{V}al\llbracket\text{exp}_i\rrbracket\rho$  for  $i \in \{1 \dots 3\}$ ,  $\rho = \omega + \rho'$  and  $\rho'(y) = 3$ ,  $\rho'(x) = \text{False}$ .

Describe your computation in detail and for each higher-order function  $f : \text{Dom} \rightarrow \text{Dom}$  that occurs in the calculation, determine what the function  $f^i(\perp)$ ,  $i \in \mathbb{N}$ , computes.

```
exp1 := let greater5 = \x -> if x > 5 then True else False
        in greater5 y
```

```
exp2 := let even' = \x -> if x == 0 then True
                        else not (even' (x - 1))
        in even' 4
```

```
exp3 := let times = \x -> \y -> if y == 0 then 0
                        else y + times (x-1) y
        in times 2 4
```