Christian-Albrechts-Universität zu Kiel
Institut für Informatik
Lehrstuhl für Programmiersprachen und Übersetzerkonstruktion

Sandra Dylus, Priv.-Doz. Dr. Frank Huch, Niels Bunkenburg

Institut für Informatik

2. Klausur zur Vorlesung „Advanced Programming (pre-masters)“

WS 18/19

You can obtain 28 points within two assignments. To pass the test, you have to reach at least 14 points.

You have 90 minutes to complete the test. It is not allowed to use any material other than a pen. Electronic devices have to be turned off.

Hold your student card nearby, we will check it during the examination.

You will be informed about the results on Wednesday, December 19th (10:00 to 11:00 AM) in room number 703 in CAP 4.

Aufgabe 1 - Polymorphic Binary Leaf-Labeled Trees

16 Punkte

1. (4P) Define a polymorphic data type for node-labeled trees that allows nodes with **up to** two child nodes, that is, nodes with only one or no child node should also be possible.
2. (8P) We consider the following algebraic data type for leaf-labeled trees.

   ```
   data T a = L a | B (T a) (T a)
   ```

   Implement the following functions.

   ```
   -- counts the number of leaves
   count :: T a -> Int

   -- similar to map on lists
   mapTree :: (a -> b) -> T a -> T b

   -- converts a tree into a list containing all values of the tree
   treeToList :: T a -> [a]

   -- converts a list into a tree containing all values of the list;
   --  the tree can be completely unbalanced
   listToTree :: [a] -> T a
   ```

3. (4P) We consider the following function.

   ```
   foldTree :: (a -> b) -> (b -> b -> b) -> T a -> b
   foldTree leaf branch (L x)     = leaf x
   foldTree leaf branch (B tl tr) =
     branch (foldTree leaf branch tl) (foldTree leaf branch tr)
   ```

   a) Define a function `sumTree` that sums up the values of all leaves of an tree of type `T Int` using `foldTree`, e.g., your definition should look like `sumTree t = foldTree ...`.
   b) Reimplement the function `mapTree` using `foldTree`, e.g., your definition should look like `mapTree f t = foldTree ...`.

Aufgabe 2 - Multiple Choice and Questions 12 Punkte

Answer the following questions directly on this sheet of paper.

For the first five questions, simply mark all correct answers by checking the circle in front of the statement. Note that any number of answers can be correct. Each question is worth 1.5 points. For each incorrect answer 0.5 points are deducted; negative scores are not possible.

1. Which of the following expressions will yield a type error?

   a) O `[(1),(2)] ++ [()]`
   b) O `(1 : 2 : []) : [[3]]`
   c) O `Just Nothing`
   d) O `[(1,2),(1,2,3)]`
   e) O `(False, [True])`

2. The function `twice f x = f (f x)` has the following type.

   a) O `a -> a -> a`
   b) O `(a -> a) -> a -> a`
   c) O `a -> (a -> a) -> a`
   d) O `a -> a -> (a -> a)`
   e) O `(a -> a) -> (a -> a)`

3. The following expressions are valid with respect to the data type
   `data Tree a b = Tip a | Branch (Tree a b) b (Tree a b)`.

   a) O `Branch (Tip 2) (Tip True) (Tip 4)`
   b) O `Branch (Tip 1) (Tip 2) (Tip 3) (Tip 4)`
   c) O `Branch (Tip 1) "Hallo" (Tip 2)`
   d) O `Branch (Tip "Hallo") () (Tip "Welt")`
   e) O `Tip (Tip 4)`

4. Which of the following expressions evaluate to `36`?

   a) O `foldr (+) 5 [5,11,20]`
   b) O `uncurry (+) (20,16)`
   c) O `foldl (*) 3 [1,3,4]`
   d) O `[1..100] !! 35`
   e) O `fst (6 + 30, 42)`

5. Which of the following types are correct?

   a) O `map :: (a -> b) -> [a] -> [b]`
   b) O `map :: a -> b -> [a] -> [b]`
   c) O `map (+) :: [Int] -> [Int -> Int]`
   d) O `map (\ s -> '!') :: String -> String`
   e) O `flip map [1..10] :: (Int -> a) -> [a]`

1. (1P) We define the following algebraic data type for Boolean expressions.

```
data BExp = Var String
          | Or BExp BExp
          | And BExp BExp
          | Neg BExp
```

Define a smart constructor `impl :: BExp -> BExp -> BExp` which constructs a boolean formula for implication.

Note that implication is defined as $A \longrightarrow B := \neg A \vee B$.

2. (1.5P) We define the following algebraic data type.

```
data A a = A a
         | B (A a)
```

Give three different values of type `A Int` which may only contain `73` as value of type `Int`.

3. (2P) We define the following function.

```
fun p [] = []
fun p (x:xs) | p x       = x : fun p xs
             | otherwise = []
```

Give the type of `fun`.

Explain the behaviour of `fun` (i.e. describe its semantics) with at most two sentences.