universität innsbruck

Functional Programming | SS 2020 | LVA 703025

Exercise Sheet 11, 10 points | Deadline: **Tuesday, June 2, 2020, 10am**

- Watch the lecture of week 11 and study part 8 of the slides[1].
- Please write all the Haskell code into a single .hs-file and upload it in OLAT.
- You can use the template .hs-files that are provided on the proseminar page[2].
- Your .hs-files should be compilable with ghci.
- Don't forget to mark your completed exercises in OLAT.

Exercise 11.1 *Brackets* **3 p.**

Implement a function `validBrackets` that checks if brackets in a string are correctly set. Bracket characters are `()`, `[]` and `{}`. A string contains correctly set parentheses if:

- Open brackets are closed by the same type of brackets.
- Open brackets are closed in the correct order.

Examples:

```
validBrackets "(a[bc])" = True
validBrackets "(){ab}c" = True
validBrackets "({ab)}c" = False
```

Exercise 11.2 *Longest Palindrom* **3 p.**

Define a function `longestPalindrom` that finds the longest palindrom in a list of elements of type `Eq a => a`. Examples:

```
longestPalindrom "xabai" = "aba"
longestPalindrom "hannah steht neben dem regallager." = "regallager"
longestPalindrom [1,2,1,3,4,3] = [1,2,1] -- [3,4,3] is also a valid answer
```

Exercise 11.3 *Valid Binary Search Trees* **2 p.**

Consider the following Haskell datatype representing a binary tree:

```
data Tree a = Empty | Node (Tree a) a (Tree a)
```

Write a function `validBST :: Ord a => Tree a -> Bool` which checks if a binary tree is a valid binary search tree (BST). A binary tree is a BST if:

- The left subtree of a node contains only nodes with keys less than the node's key.
- The right subtree of a node contains only nodes with keys greater than the node's key.
- Both the left and right subtrees must also be BSTs.

[1] http://cl-informatik.uibk.ac.at/teaching/ws19/fp/slides/08x1.pdf
[2] http://cl-informatik.uibk.ac.at/teaching/ss20/fp/index.php#exercises

Examples:

```
validBST Empty = True
validBST (Node (Empty) 5 (Node (Node Empty 4 Empty) 9 Empty)) = False
validBST (Node (Node Empty 4 Empty) 5 (Node Empty 8 Empty)) = True
```

Note: The lecture notes contain function definitions which convert binary trees to lists consisting of all elements of the tree.

Exercise 11.4 *IO* **2 p.**

Write a function `addNumbers :: IO ()` which ask for two integers as input, adds them up and shows the output. After running the function your terminal should look like this:

```
Number 1:
43
Number 2:
5
Sum: 48
```

`43` and `5` were entered by the user. The rest is output of the programm. On non-valid inputs (i.e. entering something else then an integer) your programm can simply fail.