CS_205 Declarative Programming 23 October 2020
Ulrich Berger, Monika Seisenberger

Lab Task 4: Higher Order Functions and Recursion

1. Define a function that takes a function of type `Int -> a` as input, and computes the result of the input function at value `7`.

Run this function at your Haskell prompt with two own functions (of different types).

2. Write a recursive function `extractDigits:: String -> String` that recursively goes through a given string and computes a new string containing all the digits in the given string.

Hint: you may use the function `isDigit :: Char -> Bool` from the library module Data.Char. To import that module start your file with `import Data.Char`.

3. For the next questions, define first a function that tests whether a string is a palindrome and a function that computes the square of an integer, as well as the list

```
testlist =["madam", "adam", "otto", "else", "kajak", "seas"]
```

a) Rewrite the following higher order functions using list comprehension expressions.

- `test0 = map (+3)  [2,3,4,5,6]`
- `test1 = map palindrome testlist`
- `test2 = filter palindrome testlist`

Solution for `test0`: `testlcompr0 = [x+3| x<- [2,3,4,5,6]]`. If you call `test0` and `testlcompr0` at the `ghci` prompt, they should produce the same lists.

b) Write down the following list comprehension expressions using higher order functions.

- `test3 = [(square x) + 3| x <-[1..500]]`

c) Challenge (optional)

- `challenge = [x|x <-testlist, length x == 4]`

4. Challenge (optional). Define recursive functions:

```
insertionSort :: Ord a => [a] -> [a]
insertElement :: Ord a => a -> [a] -> [a]
```

`insertionSort` is a function that sorts a given list and is defined recursively by pattern matching. It makes use of the function `insertElement` which inserts an element at the right place in a list (which you can assume to be already sorted).

Hint: `InsertElement` inserts an element in an already sorted list in such a way that the result list is still sorted.

Define these functions by completing the following templates:

```
insertElement x [] = ...
insertElement x (y:ys) = if (x < y) then ... else ...
insertionSort  [] =  ...
insertionSort (x:xs) = insertElement x (  ...   )
```