Converting Anagrams

Oliver Krischer

September 2020

1 Problem Description

Write a function that creates a printout of anagrams of all the words with a given length in a text with the following specification:

```
anagrams :: Int → Text → Dictionary
```

2 solution

```
module Anagrams where
import Data.Char (toLower)
import Data.List (sort)
import qualified Data.Set as Set
import Flow


type Dictionary = String
type Wort = String
type Text = String
type Label = String


anagrams :: Int → Text → Dictionary
anagrams n text = getWords n text
  |> map addLabel
  |> sortLabels
  |> groupByLabel
  |> showEntry
```

extract the words of length n

```
getWords :: Int → Text → [Wort]
getWords n text = [word | word ← words (map toLower text), length word ≡ n]
```

take each word and add a label to it

```
addLabel :: Wort → (Label, Wort)
addLabel word = (sort word, word)
```

sort list of label-word-tuples in aplphabetical order

```
sortLabels :: [(Label, Wort)] → [(Label, Wort)]
sortLabels xs = sort xs
```

replace each group of labelled words with the same label with a single entry using an accumulator (Set.empty) and a helper function

```
groupByLabel :: [(Label, Wort)] → [(Label, [Wort])]
groupByLabel xs = groupByHelp Set.empty xs
groupByHelp :: Set.Set Label → [(Label, Wort)] → [(Label, [Wort])]
groupByHelp _ [] = []
groupByHelp set xs
  | ¬ (Set.member label set) =
    (label, [word | (label', word) ← xs, label ≡ label'])
    : groupByHelp (Set.insert label set) (tail xs)
  | otherwise = groupByHelp set (tail xs)
  where label = fst (head xs)
```

replace each enty by a string and concatenate the results

```
showEntry :: [(Label, [Wort])] → Dictionary
showEntry [] = []
showEntry (x : xs) = fst x ++ ": " ++ unwords (snd x) ++ "\n" ++ showEntry xs
```