Tutorial 8A: Folds

In this tutorial we will look at **folds** over lists, in combination with `map` and `filter`, and practice some more recursion.

Exercise 1: The structure of this exercise is as follows: you are given a description of a function, and you are asked to implement it in three ways:

1. using **recursion**,
2. using a **list comprehension** and **library functions**, and
3. using `map`, `filter`, `foldr` and `foldl`.

For each implementation, you may not need to use each part: for instance, you won't always need both a `filter` and a `map`. But you shouldn't use the key functionality of the **other** implementations: for example, for implementation 2 you shouldn't use recursion or a filter, and for implementation 3 you shouldn't use a list comprehension.

Library functions that you might want to use are: `and`, `or`, `any`, `all`, `maximum`, `minimum`, `product`, `sum`, `concat`. If you don't know some of them, look up the type and try them out.

In any of the assignments you may use basic functions such as `(==)`, `(<=)`, `(&&)`, `(||)`, `(++)`, `even`, `odd`, `max`, `min`, `head`, `tail`, and `length`.

a) The function `allTrue` that, given a list of booleans, returns whether all of them are `True`.

b) The function `longestLength` that, given a non-empty list of lists, returns the length of the longest list. An empty list may return an error. (This would be a good case to use `foldl1` or `foldr1`.)

c) The function `sumOddSquares` that, given a list of integers, selects the odd ones, squares them, and sums the result.

d) The function `shortFWords` that, given a list of non-empty strings, returns whether any of them is a four-letter word starting with capital letter `'F'`.

e) The function `wordScore` that, given a string, returns the sum "score" of its letters, where the letter `'a'` or `'A'` scores one, `'b'` or `'B'` scores two, etc. until `'z'` or `'Z'` scores 26. Any other symbol gets score zero. You can use the following helpful functions (partly from `Data.Char`):

   - `toUpper` and `toLower` to change the case of a letter,
   - `ord` to get the ASCII integer index of a letter,

- `subtract` to get from `ord 'A'` to 1; the minus symbol `(-)` doesn't make sections easily, since e.g. `(-3)` is interpreted as the integer "negative three".

Use `where`-clauses where needed.

f) The function `concatCheapWords` which, given a list of words, selects the ones with a `wordScore` of 42 or less, adds a space at the front of each, and then concatenates them.

```
*Main> ws =
["Smoke","me","a","kipper","I'll","be","back","for","breakfast"]

*Main> allTrue [False,True,False]
False

*Main> allTrue []
True

*Main> longestLength ws
9

*Main> sumOddSquares [1..100]
166650

*Main> shortFWords ["Fish","for","breakfast","??"]
True

*Main> map wordScore ws
[63,18,1,75,33,7,17,39,83]

*Main> concatCheapWords ws
" me a I'll be back for"
```