Functional Programming for BDA - List 2
Maps and folds

Marcin Michalski, DPM FPAM WUST 2021/2022

10.11.2021

Solutions should match the topic of the list, i.e. maps and folds, so use them plenty.

Exercise 1. Implement a function that for a list of integers calculates its sum of squares.

Exercise 2. Implement a function that for a list of integers calculates its sum of squares of prime members.

Exercise 3. Implement a function that for a list of integers returns how many even members it contains.

Exercise 4. Implement a function that for a list of integers calculates the mean of its members. Try not to use explicitly the length of the list.

Exercise 5. Implement `foldr` without checking it out in the documentation.

Exercise 6. Express `map` via `foldr` and `foldl`. *Hint: it may be a good idea to use* `z=[]`*.*

Exercise 7. Implement a function `rev_rev :: [[Char]] -> [[Char]]` that takes a list of strings and returns the list of reversed strings in reversed order, i.e.

```
rev_rev ["lorem", "ipsum"] == ["muspi", "merol"]
```

Exercise 8. Implement a function `my_filter :: a -> Bool -> [a] -> [a]` that takes a predicate `p :: a -> Bool`, list of elements, and returns a list of elements satisfying `p` in two ways:

(i) using recursion without maps or folds;

(ii) using maps or folds.

Exercise 9. Implement a function `approx_e :: Int -> Double` calculating for each natural $\sum_{k=0}^{n} \frac{1}{k!}$ for each natural n. It should work pretty fast, e.g. calculating $k!$ from the ground with each "iteration" is unacceptable. *Hint: use accumulator storing* $k!$

Exercise 10. Go back to the previous list and see which exercises could be done quicker with maps and folds. Implement new solutions.