Functional Programming for BDA - List 1

Marcin Michalski, DFCS FFPT WUST 2020/2021

Deadline: 04.11.2020 $(0:00-\varepsilon)$

Treat exercises as a warm-up, **submit tasks only**! Use ePortal for submission, there is the assignment named List 1 attached to Lecture 2.

Exercise 1. Upgrade the following implementation of the factorial so that it is tail recursive and *fast*

```
factorial 0 = 1
factorial n = n * (factorial (n-1))
```

Exercise 2. Upgrade the following implementation of list reversing so that it is tail recursive and *fast*

```
rev [] = []
rev (x:xs) = (rev xs) ++ [x]
```

Exercise 3. Implement a function that for a given natural n quickly counts amount of zeros at the end of $n!$

Exercise 4. Implement your own functions that curry and decurry functions, i.e. for $f \in C^{(A\times B)}$ and $g \in (C^B)^A$

```
(my_curry f) a b = f (a,b),
(my_decurry g) (a,b) = g a b.
```

Exercise 5. Implement a function that computes $\binom{n}{k}$. Don't use $\frac{n!}{k!(n-k)!}$, it is too expensive - use recursion instead.

Task 1. Implement the sieve Eratosthenes. Your solution should be fast.

Task 2. The Euler's totient function $\varphi : \mathbb{N}_+ \to \mathbb{N}$ is defined as follows

$$\varphi(n) = |\{k \in \mathbb{N}_+ : \gcd(k, n) = 1\}|$$

Implement

(a) the Euler's totient function.

(b) a function $f(n) = \sum_{d\in\{k\in\mathbb{N}_+ : k|n\}} \varphi(d)$. Put forward a hypothesis and try to prove it.

Task 3. (a) Implement a function that calculates the n-th member of Fibonacci sequence in a linear time.

(b) The same for the sequence

$$a_0 = 1,$$
$$a_1 = 1,$$
$$a_n = n + a_{n-1} + a_{n-2}.$$

Task 4. Implement a function that for a given string (which is a list of characters)

(a) `ecd` that eliminates consecutive duplicates, i.e.

```
ecd [1,1,2,3,3] == [1,2,3].
```

(b) `encode` that encodes consecutive duplicates with integer, i.e.

```
encode [a,a,a,b,b,c] == [(a,3), (b,2), (c,1)]
```

(c) `decode` that decodes the previous one, i.e.

```
decode (encode xs) == xs
```

Task 5. Implement a function

(a) `power_list` that for a given list returns the lists of all its sublists, i.e.

```
power_list [1,2] == [[],[1], [2], [1,2]]
```

(b) `perm` that for a given list returns a list of all its permutations.