ИЗПИТ ПО ФУНКЦИОНАЛНО ПРОГРАМИРАНЕ
Специалност „Информационни системи“, 28.08.2020 г.

Задача 1. Да се дефинира функция `listOfIndexes :: Int -> [Int] -> [Int]`, която получава естествено число `n` и списък `xs` и връща списък от индексите (поредните номера) на елементите на `xs`, които са равни на `n`. Предполага се, че индексирането на елементите на списъка започва от 0.
Примери:
`listOfIndexes 3 [1,2,3,4,3,5,3,2,1]` ⟶ `[2,4,6]`
`listOfIndexes 4 [1,2,3,2,1,2,3,2,1]` ⟶ `[]`

Задача 2. Нека `n` е положително цяло число, а `xs` е списък от числа.
(а) Да се дефинира функция `digits n`, която връща списък от цифрите на `n` в подредба от най-старшата към най-младшата.
Пример:
`digits 4321` ⟶ `[4,3,2,1]`

(б) Да се дефинира функция `decreasing xs`, която проверява дали елементите на списъка `xs` са подредени в строго намаляващ ред.
Примери:
`decreasing [4,3,2,1]` ⟶ `True`
`decreasing [4,3,5,1]` ⟶ `False`
`decreasing [4,3,3,1]` ⟶ `False`

(в) Да се дефинира функция `decDigits n`, която проверява дали цифрите на `n` са подредени в строго намаляващ ред от първата към последната. За първа се приема най-старшата цифра на `n`.
Примери:
`decDigits 4321` ⟶ `True`
`decDigits 4322` ⟶ `False`
`decDigits 7635` ⟶ `False`

Задача 3. Да се дефинира функция `averageFunction fs`, която за даден непразен списък от едноаргументни аритметични функции `fs = [`f_1`,`f_2`, … ,`f_n`]` връща като резултат функция, чиято стойност в дадена точка `x` се получава като средното аритметично на стойностите на функциите от `fs` в тази точка.
Пример:
`(averageFunction [(+1),(**0.5),(2**)]) 2` ⟶ `2.804738`

Задача 4. Нека представим двоично дърво от цели числа чрез конструкцията `data BTree = Empty | Node Int BTree Btree .`
Да се дефинира предикат `isBalanced :: BTree -> Bool`, който проверява дали дадено двоично дърво от цели числа е балансирано.
Казваме, че едно двоично дърво е балансирано, ако за всеки негов връх височините на лявото и дясното му поддърво се различават най-много с 1.
Примери:

```
t1 :: BTree                                    --    5
t1 = Node 5 Empty                              --     \
            (Node 4 (Node 5 Empty Empty)       --      4
                    (Node 7 Empty Empty))      --     / \
                                               --    5   7

t2 :: BTree                                    --    5
t2 = Node 5 (Node 3 Empty Empty)               --   / \
            (Node 4 (Node 5 Empty Empty)       --  3   4
                    (Node 7 Empty Empty))      --     / \
                                               --    5   7
```

```
isBalanced t1 ⟶ False
isBalanced t2 ⟶ True
```