Изпит по Функционално програмиране
Специалност „Информационни системи“, I курс, 30.06.2023 г.

Задача 1.

Нека дума е максимален подниз, състоящ се само от символи, различни от интервали (думата се състои само от малки и големи букви и цифри).

Даден е низ str, състоящ се от поне две думи и интервали. Дефинирайте функция `countLetters :: String -> Int`, която получава низа `str` и връща дължината на предпоследната дума в `str`.

Примери:

```
countLetters "Hello World"               → 5
countLetters "   haskell is    great "  → 2
countLetters "Information Systems 2023" → 7
```

Задача 2.

Анаграма е дума или фраза, образувана чрез пренареждане на буквите на друга дума или фраза, като обикновено всички оригинални букви се използват точно веднъж.

Дефинирайте функция `groupEquals :: [String] -> [[String]]`, която получава списък от низове `strs` и групира анаграмите от `strs` заедно в подсписъци. Можете да върнете отговора в произволен ред.

Примери:

```
groupEquals ["eat","tea","tan","ate","nat","bat"] →
                        [["bat"],["nat","tan"],["ate","eat","tea"]]
groupEquals [""]  → [[""]]
groupEquals ["a"] → [["a"]]
```

Задача 3.

Нека е дадена правоъгълна целочислена матрица `mat`, представена като списък от списъци. Напишете едноместна функция от по-висок ред `resetMatrix :: [[Int]] -> ((Int -> Bool) -> [[Int]])`, която получава като аргумент матрицата `mat`.

Функцията `resetMatrix` трябва да върне нова едноместна функция, която получава като единствен аргумент предикат `p`.

Резултатът от изпълнението на върнатата функция трябва да е нова матрица, получена от `mat` чрез зануляване на всички колони, в които е имало стойност, която удовлетворява предиката `p`.

Примери:

```
fn = resetMatrix [[1,2,3,4],[5,6,7,8],[9,10,11,12]]
fn (== 5)  → [[0,2,3,4],[0,6,7,8],[0,10,11,12]]
fn (>= 11) → [[1,2,0,0],[5,6,0,0],[9,10,0,0]]
fn (> 20)  → [[1,2,3,4],[5,6,7,8],[9,10,11,12]]
```

Задача 4.

Нека е дадена следната дефиниция за двоично дърво:
```
data BTree = Empty | Node Int BTree BTree deriving (Eq, Show)
```

Дефинирайте функция `pruneTree :: BTree -> Int -> BTree`, която получава двоично дърво `bt` и стойност `n`. Функцията трябва да върне като резултат ново двоично дърво със същата структура като `bt`, но в което са премахнати всички поддървета, които не съдържат възел със стойност `n`. Всяко дърво е поддърво на себе си.

Примери:
```
bt0 = Node 6 Empty Empty
bt1 = Node 1 Empty
             (Node 0 (Node 0 Empty Empty)
                     (Node 1 Empty Empty))
bt2 = Node 1 (Node 0 (Node 0 Empty Empty)
                     (Node 0 Empty Empty))
             (Node 1 (Node 0 Empty Empty)
                     (Node 1 Empty Empty))
bt3 = Node 1 (Node 1 (Node 1 (Node 0 Empty Empty)
                             Empty)
                     (Node 1 Empty Empty))
             (Node 0 (Node 0 Empty Empty)
                     (Node 1 Empty Empty))


pruneTree bt0 1 → Empty
pruneTree bt1 1 → Node 1 Empty
                         (Node 0 Empty
                                 (Node 1 Empty Empty))
pruneTree bt2 1 → Node 1 Empty
                         (Node 1 Empty
                                 (Node 1 Empty Empty))
pruneTree bt3 1 → Node 1 (Node 1 (Node 1 Empty Empty)
                                  (Node 1 Empty Empty))
                          (Node 0 Empty
                                  (Node 1 Empty Empty))
```