Поправителен изпит по Функционално програмиране - 14.08.2023 г.
Специалност Информационни системи, първи курс

Задача 1

Даден е списък от цели числа (евентуално включващ дубликати) и положително цяло число `k`. Да се дефинира функция **`findMinMaxProduct :: [Integer] -> Int -> (Integer,Integer)`**, която намира минималното и максималното възможно произведение на `k` елемента, взети от списъка. Ако списъкът не съдържа достатъчно елементи, да се връща грешка с подходящ текст.

Примери:

```
findMinMaxProduct [1, -2, -3, 4, 6, 7] 1 → (-3, 7)
findMinMaxProduct [1, -2, -3, 4, 6, 7] 2 → (-21, 42)
findMinMaxProduct [1, -2, -3, 4, 6, 7] 3 → (-126, 168)
findMinMaxProduct [0, 6, 3, 5, 4]      4 → (0, 360)
findMinMaxProduct [5, 4, 3, 3, 6]      2 → (9, 30)
findMinMaxProduct [-4, -10, -1]        2 → (4, 40)
```

Задача 2

Да се дефинира функция **`intersectionPoints`**, която получава две едноаргументни целочислени функции и краищата на целочислен интервал. Функцията да връща списък от точките от дадения интервал, в които двете функции се пресичат (т.е имат една и съща стойност), или празния списък, ако функциите не се пресичат в този интервал.

Примери:

```
intersectionPoints (\x -> x) (\x -> x * x) (-5) 5 →  [0,1]
intersectionPoints (\x -> x) (\x -> x * x + 1) (-5) 5 →  []
```

Задача 3

Обществото на информатиците провежда избори за съвет на старейшините. „Кандидатура“ е наредена двойка от низове: името на кандидата и неговата специалност, а „кандидатската листа“ е списък от кандидатури. Може да се гласува за произволен брой кандидати. Попълнените „бюлетини“ представляват предикати, които приемат кандидатура и връщат истина или лъжа, в зависимост дали се гласува за съответната кандидатура или не.
Да се дефинира функция **`election :: [((String, String) -> Bool)] -> [(String, String)] -> [(String, Int)]`**, която връща сортиран в низходящ ред списък от наредени двойки от името на кандидата и броя от гласовете за него от бюлетините ballots в реда, в който кандидатите се срещат в листата cl.

Пример:

```
cl = [("Kernighan","C"), ("Ritchie","C"), ("Stroustrup","C++"),
("Steele","Scheme"),("Sussman","Scheme"),("Church","Lambda"),("Cur
ry","Lambda")]

b1 (name, specialty) = specialty == "Lambda" || last name == 'e'
b2 (name, specialty) = name == "Church" || head specialty == 'C'
b3 (name, specialty) = length name > 6 && specialty /= "C++"

election [b1, b2, b3] cl → [("Ritchie",3), ("Kernighan",2),
("Church",2), ("Stroustrup",1), ("Steele",1), ("Sussman",1),
("Curry",1)]
```

Задача 4

Нека е дефиниран алгебричен тип, представящ двоично дърво, както следва:

```
data Tree a = Empty | Node a (Tree a) (Tree a)
```

Да се дефинира функция **`(minCount bTree x)`** от тип **`minCount :: (Eq a) => Tree a -> a -> Int`**, която намира броя на върховете (включително корена) на най-ниското поддърво на дадено двоично дърво `bTree` със стойност в корена, равна на `x`.

Пример:

```
tr = (Node 2 (Node 4 (Node 4 Empty Empty) Empty) Empty)
minCount tr 4 → 1
minCount tr 2 → 3
```