Второ контролно по Функционално програмиране

спец. Информатика, 13.01.2017 г. *Вариант А*

Задача 1 (8 т.) [Scheme/Haskell] Нека е даден списък от наредени двойки от числа, представляващи точки в равнината. Да се напише функция `findMedoid`, която намира такава точка от списъка, за която сумата от квадратите на разстоянията до всички останали точки в списъка е минимална.
Пример: `findMedoid [(2,8),(-2,4),(1,2),(-4,-1),(5,0)] → (1,2)`

Задача 2 (8 т.) [Scheme/Haskell] Да се напише функция `sumLast`, която приема две положителни естествени числа **k** и **n** и генерира безкрайния поток, в който първото число е **k**, а всяко следващо число е равно на сумата от предходните **n** числа в потока.
Пример: `sumLast 3 5 → [3, 3, 6, 12, 24, 48, 93, 183, … ]`

Задача 3 (8 т.) [Scheme/Haskell] Нека е даден ориентиран граф със символи по върховете и целочислени тегла по върховете, който е представен чрез списъци от наследници по следния начин:

```
Scheme:
(define G '((a 2 b c)))
           (b 4 a c)))
           (c 1 a b))))
```

```
Haskell:
g :: [(Char,Int,[Char])]
g = [('a', 2, "bc"]),
     ('b', 4, "ac"),
     ('c', 1, "ab")]
```

Да се напише функция `eulerCycleCost`, която проверява дали графът съдържа Ойлеров цикъл и ако да, връща цената му (сборът от теглата на върховете, през които минава). Ако Ойлеров цикъл няма, да се върне 0.

Задача 4 (8 т.) [Scheme] Да се напише функция `transformSum`, която преобразува дърво с елементи цели числа в ново дърво със същата структура, в което всеки елемент е заменен със сумата на елементите в поддървото с този корен в началното дърво.
Бонус (4 т.): `transformSum` да работи за време O(n) в най-лошия случай.

Забележка: използването на всички стандартни функции в R^5RS, както и на функциите accumulate, filter, foldr и foldl е позволено, но не е задължително.

Второ контролно по Функционално програмиране

спец. Информатика, 13.01.2017 г. *Вариант Б*

Задача 1 (8 т.) [Scheme/Haskell] Нека е даден списък от наредени двойки от числа, представляващи точки в равнината. Да се напише функция `findPoint`, която намира такава точка от списъка, за която сумата от квадратите на разстоянията до всички останали точки в списъка е максимална.

Пример: `findPoint [(2,8),(-2,4),(1,2),(-4,-1),(5,0)] → (-4,-1)`

Задача 2 (8 т.) [Scheme/Haskell] Да се напише функция `multLast`, която приема две положителни естествени числа **k** и **n** и генерира безкрайния поток, в който първото число е **k**, а всяко следващо число е произведението от предходните **n** числа в потока.

Пример: `multLast 2 3 → [2, 4, 8, 64, 2048, … ]`

Задача 3 (8 т.) [Scheme/Haskell] Нека е даден ориентиран граф със символи по върховете и целочислени тегла по върховете, който е представен чрез списъци от наследници по следния начин:

```
Scheme:
(define G '((a 2 b c)))
           (b 4 a c)))
           (c 1 a b))))
```

```
Haskell:
g :: [(Char, Int, [Char])]
g = [('a', 2, "bc"]),
     ('b', 4, "ac"),
     ('c', 1, "ab")]
```

Да се напише функция `eulerPathCost`, която проверява дали графът съдържа Ойлеров път, който не е цикъл, и ако да, връща цената му (сборът от теглата на върховете, през които минава), а ако не, връща 0.

Задача 4 (8 т.) [Scheme] Да се напише функция `transformCount`, която преобразува дърво с елементи цели числа в ново дърво със същата структура, в което всеки елемент е заменен с броя на елементите в поддървото с този корен в началното дърво.

Бонус (4 т.): `transformCount` да работи в O(n) време в най-лошия случай.

Забележка: използването на всички стандартни функции в R^5RS, както и на функциите accumulate, filter, foldr и foldl е позволено, но не е задължително.