Контролна работа 2

Изисквания към задачите

Решете задачите на език racket/base. За целта в DrScheme изберете “Determine language from source” и на първия ред на програмата напишете:
`#lang racket/base.`
В решенията не може да се използват деструктивни операции
За всяка от функциите трябва да напишете и подходящи Unit test-ове.

Изисквания за задача 1

Както и в заданията за домашните, в програма на Scheme ще представяме дърво по следния начин:

Празното дърво се представя като празния списък ().
Дърво с корен N и наследници L и R представяме като списъка (N L R).
В задачата ще работим с дървета, които съдържат в корените си произволни цели числа.
По-долу е даден пример за дърво и неговото представяне:

```
   5
  / \
 22  1
/ \   \
2  6   3
      /
    111
```

Възможни представяния в Scheme:

```
(quote
  (5 (22 (2 () ()) (6 () ())) (1 () (3 (111 () ()) ()))))
)

(list 5
      (list 22
            (list 2 '() '())
            (list 6 '() '()))
      (list 1
            '()
            (list 3
                  (list 111 '() '())
                  '())))
```

Изисквания за задача 2

Важно: В решението на задача 2 не може да се използва рекурсия. Вместо това тя трябва да се реши с подходящи обръщения към една или повече от

функциите map, filter, foldl, foldr. Напомняме, че редът, в който те получават аргументите си е следният:

- `(map proc lst ...)`
- `(filter pred? lst)`
- `(foldl proc init lst)`
- `(foldr proc init lst)`
- За foldl и foldr, процедурата има вид (proc item result), където item е поредният елемент, а result -- натрупаният до момента резултат.

Вариант за групата от 09:45

Задача 1

Напишете функция (weight-balanced? tree), която проверява дали дървото tree е балансирано по тегло. Напомняме, че това означава, че за всеки възел в дървото, броят на елементите в лявото и дясното му поддървета, могат да се различават най-много с 1.

Ако tree е празното дърво, функцията да връща #t.

Задача 2

Важно: В решението на задача 2 не може да се използва рекурсия. Вместо това тя трябва да се реши с подходящи обръщения към една или повече от функциите map, filter, foldl, foldr. Напомняме, че редът, в който те получават аргументите си е следният:

- `(map proc lst ...)`
- `(filter pred? lst)`
- `(foldl proc init lst)`
- `(foldr proc init lst)`
- За foldl и foldr, процедурата има вид (proc item result), където item е поредният елемент, а result -- натрупаният до момента резултат.

Резултат от явяване на изпит по даден предмет ще представяме като списък от следния вид:

`(list <фн> <предмет> <оценка>)`

където:

- <фн> е естествено число;
- <предмет> е символен низ;
- <оценка> е реално число.

Възможно е един студент да се яви на даден изпит няколко пъти, например за поправка или за повишаване на оценката.

Напишете функция `(attempts subj res)`, която получава списък от резултати от изпити `res` и име на предмет `subj`. Функцията да връща нов списък от двойки от следния вид:

```
(list (cons fn1 c1) (cons fn2 c2) ... )
```

- fn1, fn2, ... са факултетните номера на студентите, които са се явили на изпит по subj. Те трябва да се подредени точно в реда, в който се срещат в res.
- c1 е броят пъти, които fn1 се е явил на изпит по subj;
- c2, е броят за fn2;
- и т.н.

В резултата от функцията не трябва да има повторения. Тоест, за всяко i, j, трябва да е изпълнено fi ≠ fj.

Ако в res няма информация за студенти, които да са се явили по subj, функцията да връща празния списък ().

Упътване: Филтрирайте входа и оставете само данните за subj. Напишете функция, която връща списък от факултетните номера на всички студенти, които са се явили по subj. Напишете и функция, която премахва всички повторения от даден списък. Използвайте ги, за да решите задачата.

Примери:

```
(attempts "a" (list (list 12345 "a" 5)
                    (list 12345 "b" 6)
                    (list 12345 "b" 6)
                    (list 54321 "a" 5)
                    (list 54321 "b" 6)
                    (list 54321 "a" 5)
                    (list 54321 "b" 6)
                    (list 54321 "c" 6))) ; връща (list (cons 12345 1)
                                         ;             (cons 54321 2)))

(attempts "d" (list (list 12345 "a" 5)
                    (list 12345 "b" 6)
                    (list 54321 "b" 6)) ; връща '()
```