Първо контролно по Функционално програмиране

спец. Информатика, 15.11.2016 г. *Вариант А*

Задача 1. (8 т.) Да се дефинира функция `find-root` от по-висок ред, която получава като аргумент тотална монотонна едноместна функция **f** с графика пресичаща абсцисата и намира корен на уравнението **f(x) = 0** с точност 10^{-6}, т.е. намира число **x**, така че $|x - x_0| < 10^{-6}$, където x_0 е единственият корен на горното уравнение.
Пример: `(find-root (lambda (x) (- (* 3 x) 6)) → 2`

Задача 2. (8 т.) Да се напише предикат `divsum`, който приема списък от списъци от цели числа. Предикатът да връща `#t` точно когато всеки от елементите на някой от списъците дели сумата на числата в един от другите списъци. Приемаме, че сумата на числата в празния списък е равна на **0** и съответно всяко число я дели.
Пример: `(divsum '((1 2) (1 3 6) (4 5))) → #t,`
Пример: `(divsum '((1 2) (5) (7 10))) → #f`

Задача 3. (7 т.) Да се напише предикат `all-increasing?`, който проверява дали всички колони на дадена матрица от числа образуват растяща редица.
Пример: `(all-increasing? '((1 2 3) (4 5 6) (7 8 9))) → #t`
Пример: `(all-increasing? '((1 9 3) (4 5 6) (7 8 9))) → #f`

Задача 4. (7 т.) Казваме, че списъкът **x** = (x_1 x_2 … x_{2n}) от цели числа се получава от прочитането (look-and-say) на списъка **y**, ако **y** се състои от последователно срещане на x_1 пъти x_2, последвано от x_3 пъти x_4, и така нататък до x_{2n-1} пъти x_{2n}. Да се дефинира функция `next-look-and-say`, която по даден списък **y** намира списъка **x**, получен от прочитането **y**.
Пример: `(next-look-and-say '(1 1 2 3 3)) → '(2 1 1 2 2 3)`

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

Първо контролно по Функционално програмиране

спец. Информатика, 15.11.2016 г. *Вариант Б*

Задача 1. (8 т.) Да се дефинира функция `solve` от по-висок ред, която получава като аргументи две едноместни тотални монотонни функции **f** и **g** с пресичащи се графики — едната от тях (неизвестно коя) растяща, а другата намаляваща и намира корен на уравнението **f(x) = g(x)** с точност 10^{-6}, т.е. намира число **x**, така че $|x - x_0| < 10^{-6}$, където x_0 е единственият корен на горното уравнение.
Пример: `(solve (lambda(x) (- (* x x x)) (lambda(x) (- x 2))) → 1`

Задача 2. (8 т.) Да се напише предикат `checksum`, който приема списък от списъци от цели числа. Предикатът да връща **#t** точно когато всеки от списъците съдържа елемент, който е равен на сумата на числата в някой от другите списъци. Приемаме, че сумата на числата в празния списък е равна на **0**.
Пример: `(checksum '((1 1) (2 3) (-1 2))) → #t`,
Пример: `(checksum '((1 2) (5) (7 10))) → #f`

Задача 3. (7 т.) Да се напише предикат `exists-increasing?`, който проверява дали съществува колона на дадена матрица от числа, която образува растяща редица.
Пример: `(exists-increasing? '((1 8 3) (10 9 6) (7 2 9))) → #t`
Пример: `(exists-increasing? '((7 8 9) (1 9 3) (4 5 6))) → #f`

Задача 4. (7 т.) Казваме, че списъкът **x** = (x_1 x_2 … x_{2n}) от цели числа се получава от прочитането (look-and-say) на списъка **y**, ако **y** се състои от последователно срещане на x_1 пъти x_2, последвано от x_3 пъти x_4, и така нататък до x_{2n-1} пъти x_{2n}. Да се дефинира функция `prev-look-and-say`, която по даден списък **x** намира от прочитането на кой списък **y** е получен той, или връща **#f**, ако **x** не може да е получен от прочитането на някой списък.
Пример: `(prev-look-and-say '(1 1 2 3 1 2)) → '(1 3 3 2)`
Пример: `(prev-look-and-say '(1 1 5 1 1 2)) → #f`
Забележка: използването на всички стандартни функции в R5RS, както и на функциите accumulate, filter, foldr и foldl е позволено, но не е задължително.