Изпит по Функционално програмиране

спец. Информатика и Софтуерно инженерство, 31.01.2018 г. *Вариант А*

Задача 1. (10 т.) Да се напише функция, `generateExponents`, която по дадени различни естествени числа **k** и **l**, генерира безкрайния поток от всички числа от вида $\mathbf{x}^k\mathbf{y}^l$, без повторения и подредени в нарастващ ред.
Пример: `generateExponents 2 3 → [1, 4, 8, 9, 16, 25, 27, ... ]`

Задача 2. (10 т.) Път от корен до възел в двоично дърво кодираме с поредица от цифри 0 и 1, която започва с цифрата 1, а за всяка следваща цифра 0 означава завиване по левия клон, а 1 — по десния. Да се реализира функция `sameAsCode`, която в двоично дърво от числа връща такова число **x**, което съвпада по стойност с двоичното число, кодиращо пътя от корена до **x**, или 0, ако такова число няма. Представянето на дървото е по ваш избор.
Пример: `(sameAsCode '(5 (3 () (2 () ())) (4 (6 () ()) ())))  → 6`

Задача 3. (12 т.) Дадени са списък от **n** списъци от числа l_1, …, l_n и списък от **n** функции f_1, …, f_n. Да се реализира функция `allEqual`, която връща списък x_1, …, x_n, където $x_i \in l_i$ и $f_i(x_i)$ са еднакви по стойност. В случай, че такива x_i не съществуват, функцията да връща празния списък.
Пример: `allEqual [[1,2], [3,4], [5,6]] [(+1), id, (8-)] → [2,3,5]`

Задача 4. Лекарство се задава със наредена двойка от име (низ) и списък от активни съставки, зададени като наредени двойки от име (низ) и количество в мг (цяло число). Казваме, че лекарството **A** е **заместител** на лекарството **B**, ако **A** има точно същите активни съставки като **B** в същата пропорция.

a. (4 т.) Да се реализира функция `isSubstitute`, която по две дадени лекарства проверява дали едното е заместител на другото.
b. (6 т.) Да се реализира функция `bestSubstitutes`, която по лекарство **A** и списък от лекарства **L** намира името на "най-добрия" заместител на **A** в **L**, чиито активни съставки са най-близки по количество до тези на **A**, без да ги надхвърлят, или празният низ, ако такъв няма.
c. (8 т.) Да се реализира функция `groupSubstitutes`, по даден списък от лекарства ги групира по “заместителство”, т.е. връща списък от списъци от лекарства, където всички лекарства в даден списък са заместители един на друг.

Пример:
```
l = [("A",[("p",6),("q",9)]),("B",[("p",2),("q",3)]),("C",[("p",3)])]
isSubstitute (l!!0) (l!!1) → True       bestSubstitute (l!!0) (tail l) → "B"
groupSubstitutes l → [[("A",...),("B",...)],[("C",...)]]
```

Изпит по Функционално програмиране

спец. Информатика и Софтуерно инженерство, 31.01.2018 г. *Вариант Б*

Задача 1. (10 т.) Да се напише функция `generatePowers`, която по дадени различни естествени числа **k**, **l** ≥ 2, генерира безкрайния поток от всички числа от вида $\mathbf{k}^x\mathbf{l}^y$, без повторения и подредени в нарастващ ред.
Пример: `generatePowers 2 3 → [1, 2, 3, 4, 6, 8, 9, ... ]`

Задача 2. (10 т.) Път от корен до възел в двоично дърво кодираме с поредица от цифри 0 и 1, която започва с цифрата 1, а за всяка следваща цифра 0 означава завиване по левия клон, а 1 — по десния. Да се реализира функция `countCodes`, която в двоично дърво от числа намира броя на числата **x**, които съвпадат по стойност с двоичното число, кодиращо пътя от корена до **x**. Представянето на дървото е по ваш избор.
Пример: `(countCodes '(1 (2 () (3 () ()) (4 (6 () ()) ()))) → 3`

Задача 3. (12 т.) Даден е списък от **n** списъци от числа l_1, …, l_n и списък от **n** функции f_1, …, f_n. Да се реализира функция `aProg`, която връща списък x_1, …, x_n, където $x_i \in l_i$ и $f_1(x_1)$, $f_2(x_2)$, ... $f_n(x_n)$ образуват аритметична прогресия. В случай, че такива x_i не съществуват, функцията да връща празния списък.
Пример: `aProg [[1,2], [3,4], [5,7]] [(+3), id, (7-)] → [1, 3, 5]`

Задача 4. Лекарство се задава със наредена двойка от име (низ) и списък от активни съставки, зададени като наредени двойки от име (низ) и количество в мг (цяло число). Казваме, че лекарството **A** е **по-силно** от лекарството **B**, ако **A** има всички активни съставки на **B** (евентуално и още други) в поне същите количества, като поне една от съставките е в по-голямо количество.

a. (4 т.) Да се реализира функция `isStronger`, която по две дадени лекарства проверява дали първото е по-силно от второто.
b. (6 т.) Да се реализира функция `leastStronger`, която по лекарство **A** и списък от лекарства **L** намира името на лекарство **B**, което е по-силно от **A** и сумата от разликите в количествата на активните съставки на **B** и **A** е възможно най-малка, или празният низ, ако такова няма.
c. (8 т.) Да се реализира функция `strongRelation`, която по списък от лекарства **L** връща списък от наредени двойки от лекарство и списък от имена на лекарства в **L**, които са по-силни от него.

Пример:
```
l = [("A",[("p",5),("q",3)]),("B",[("p",4),("q",3)]),("C",[("p",3)])]
isStronger (l!!0) (l!!1) → True          leastStronger (l!!2) l → "B"
strongRelation l → [(("A",...),[]),(("B",...),["A"]),(("C",...),["A","B"])]
```