Изпит по Функционално програмиране

спец. Информатика, Компютърни науки, 1 поток и избираема

8.02.2023 г. Вариант А

Задача 1. (12 т.) Нека е дадена редица от числа x_0, x_1, x_2, … За двойка елементи (x_m, x_n) числото n – m наричаме **обхват** на двойката, а всяка друга двойка от вида (x_{m+i}, x_{n-i}) за i > 1 и m + i ≤ n – i наричаме **симетрично вложена** в първата двойка. Да се реализира функция **somos**, която по дадено положително цяло число **k** генерира безкрайна редица от числа, в която първите k члена са равни на 1 и произведението на всяка двойка елементи с обхват **k** е равно на сумата от произведенията на всички симетрично вложени в нея двойки елементи.

Пример: `somos 4` → `1 1 1 1 2 3 7 23 59 314 1529 …` (напр. 1.23 = 2.7 + 3.3)

Бонус: (5 т.) Да се намерят **k** и **n**, такива че n-тият елемент на редицата **somos k** НЕ Е цяло число.

Задача 2. (20 т.) Недетерминиран краен автомат N наричаме наредена четворка (Q, q_0, T, F), където Q е крайно множество от състояния, $q_0 \in Q$ е начално състояние, $T \subseteq Q \times \{a,b\} \times Q$ е релация на преходите, а $F \subseteq Q$ е множество на крайни състояния. Казваме, че думата $c_1c_2...c_n$ се разпознава от автомата N ако съществува редица от преходи (q_0, c_1, q_1), (q_1, c_2, q_2), …, (q_{n-1}, c_n, q_n), такава че $q_n \in F$. Да се реализира функция **rejectedWord**, която връща някоя дума, която **НЕ СЕ** разпознава от даден автомат, а ако автоматът разпознава всички възможни думи, да се върне подходящ резултат. Представянето на думата в Scheme е като списък от еднобуквени символи, а в Haskell е като низ. Представянето на автомата е по ваш избор.

Упътване: автомат с **n** състояния разпознава всички възможни думи точно тогава, когато разпознава всички възможни думи с дължина до **n**.

Пример: Автоматът ({0, 1, 2}, 0, {(0,a,1),(1,b,1),(1,a,2),(2,a,2),(2,b,1)}, {0, 1, 2}) не разпознава думата "b", но ако добавим прехода (0,b,1) вече разпознава всички думи.

Задача 3. **(12 т.)** Нека е даден списък **f** от двуместни числови функции f_0, f_1, …, f_{m-1} и списък **l** от числа x_0, x_1, …, x_{n-1}. **Циклично прилагане** на **f** над **l** на позиция **i** наричаме числото

$f_i(x_0, f_{i+1}(x_1, \ldots, f_{m-1}(x_{m-i-1}, f_0(x_{m-i}, f_1(x_{m-i+1}, \ldots, f_j(x_{n-2}, x_{n-1})\ldots)))\ldots))$, където **j** е съответният последен индекс.

Да се реализира функция **findMin**, която по даден списък от функции **f** и списък от числа **l** намира минималното от **m**-те възможни циклични прилагания на **f** над **l**.

Пример: `findMin [(+),(-),(*)] [1,2,0,3,-1]` → `-7 (= 1 – (2 * (0 + (3 - (-1)))))`

Задача 4. (12 т.) Група приятели играят на бордова игра, в която получават цяло число точки. Преди да започнат играта, всеки от приятелите се опитва да предскаже колко точки ще получи накрая. След приключване на играта за всеки играч се изчислява абсолютната стойност на разликата между предсказания и реално получения брой точки, която наричаме **грешка.** Играчите се нареждат по точност на предсказанията (т.е. по нарастващ ред на грешката). Финалният резултат се определя като играчът с k-тото най-добро предсказание получава бонус точки в размер на грешката на играча с k-тото най-лошо предсказание. Да се реализира функция **finalScores**, която по списък от наредени тройки от име на играч (низ), предсказание (цяло число точки) и реален резултат (цяло число точки) връща списък от двойки от име на играч и негов финален резултат.

Пример: `finalScores [("Ангел",14,15),("Андрей",8,10),("Атанас",10,3),("Георги",6,4)]` → `[("Андрей",22),("Ангел",12),("Георги",6),("Атанас",3)]`

Изпит по Функционално програмиране

спец. Информатика, Компютърни науки, 1 поток и избираема

8.02.2023 г. Вариант Б

Задача 1. (12 т.) Да се реализира функция **`primitive`**, която по дадено положително цяло число **k**, генерира безкраен поток от всички числа, които могат да се представят като произведение на точно **k** прости числа, в нарастващ ред и без повторения.

Пример: `primitive 3 → [8, 12, 18, 20, 27, 28, ...]`

Бонус: (5 т.) Да се изрази `primitive k` рекурсивно чрез `primitive (k-1)` и потока на всички прости числа.

Задача 2. (20 т.) Недетерминиран краен автомат N наричаме наредена четворка (Q, q_0, T, F), където Q е крайно множество от състояния, $q_0 \in Q$ е начално състояние, $T \subseteq Q \times \{a,b\} \times Q$ е релация на преходите, $F \subseteq Q$ е множество на крайни състояния. Казваме, че думата $c_1c_2...c_n$ се разпознава от автомата N ако съществува редица от преходи $(q_0, c_1, q_1), (q_1, c_2, q_2), ...,(q_{n-1}, c_n, q_n)$, такава че $q_n \in F$. Да се реализира функция **`detectedWords`**, която връща краен списък с всички думи, които се разпознават от даден автомат, а ако автоматът разпознава безкрайно множество от думи, да се върне подходящ резултат. Представянето на думата в Scheme е като списък от еднобуквени символи, а в Haskell е като низ. Представянето на автомата е по ваш избор.

Упътване: езикът на автомат е безкраен, ако има път от началното му състояние до финално състояние, който съдържа цикъл.

Пример: Автоматът ({0, 1, 2, 3}, 0, {(0,a,1),(1,b,1),(0,a,2),(0,b,2),(2,a,3)}, {0, 2, 3}) разпознава думите {"", "aa", "ab"}, но ако добавим прехода (1,a,2), езикът на автомата става безкраен.

Задача 3. (12 т.) Нека е даден списък **f** от двуместни числови функции $f_0, f_1, ..., f_{m-1}$ и списък **l** от числа $x_0, x_1, ..., x_{n-1}$. **Циклично прилагане** на **f** над **l** на позиция **i** наричаме числото

$f_0(x_i, f_1(x_{i+1},..., f_{n-i-1}(x_{n-1}, f_{n-i}(x_0, f_{n-i+1}(x_1, ... ,f_{m-1}(x_{j-1}, x_j) ...))...))$, където **j** е съответният последен индекс.

Да се реализира функция **`findMax`**, която по даден списък от функции **f** и списък от числа **l** намира максималното от **n**-те възможни циклични прилагания на **f** над **l**.

Пример: `findMax [(+),(*),(-)] [2,5] → 11 (= 5 + (2 * (5 – 2)))`

Задача 4. (12 т.) Група приятели играят на бордова игра, в която получават цяло число точки. Преди да започнат играта, всеки от приятелите се опитва да предскаже колко точки ще получи накрая. След приключване на играта за всеки играч се изчислява абсолютната стойност на разликата между предсказания и реално получения брой точки, която наричаме **грешка.** Играчите се нареждат по точност на предсказанията (т.е. по нарастващ ред на грешката). Финалният резултат се определя като играчът с k-тото най-лошо предсказание получава бонус точки в размер на грешката на играча с k-тото най-добро предсказание. Да се реализира функция **`finalScores`**, която по списък от наредени тройки от име на играч (низ), предсказание (цяло число точки) и реален резултат (цяло число точки) връща списък от двойки от име на играч и негов финален резултат.

Пример: `finalScores [("Деян",10,14),("Еслин",5,8),("Мария",11,10),("Мартин",1,6)]`
`→ [("Мария",15),("Еслин",12),("Деян",17),("Мартин",7)]`