Haskell 1st tasks

Владимирова Элина

Декабрь 2022

```
import Data.Char
```

> *Задание 1.* Написать программу для нахождения n-го члена последовательности, заданной следующей рекуррентной формулой.
> $a_0 = 1;\ a_1 = 2;$
> $a_n = 3 * a_{n-1} - 2 * a_{n-2} + 1$ при $n = 2, 3, \ldots$
> Имейте в виду, что прямое программирование данной формулы «как есть» приводит к крайне неэффективной программе!

```
seq1 :: Integer -> Integer
seq1 0 = 1
seq1 1 = 2
seq1 n = 3 * seq1(n-1) - 2 * seq1(n-2) + 1

seq2 :: Integer -> Integer
seq2' :: Integer -> Integer -> Integer -> Integer -> Integer
seq2' n k sk1 sk | k == n  = sk
                 | k < n   = seq2' n (k+1) sk (3*sk - 2*sk1 + 1)
seq2 0 = 1
seq2 1 = 2
seq2 n = seq2' n 1 1 2
```

> *Задание 2.* Совершенным числом называется натуральное число, равное сумме всех своих делителей, включая единицу, но исключая само это число. Так, например, число 28 – совершенное, поскольку $28 = 1 + 2 + 4 + 7 + 14$. Написать программу для нахождения первых n совершенных чисел.

```
isprime :: Integer -> Integer      --check if n is prime
isprime' :: Integer -> Integer -> Integer
isprime n | n == 1     = 0
          | otherwise  = isprime' n 2
isprime' n m | m*m > n       = 1
             | mod n m == 0  = 0
             | otherwise     = isprime' n (m+1)

degOf2 :: Integer -> Integer        --count nth deg of 2
degOf2 0 = 1
degOf2 n = 2 * degOf2(n-1)
```

```haskell
sumDegsOf2 :: Integer -> Integer    --count sum 2^0 + 2^1 + ... + 2^n
sumDegsOf2 n | n < 0      = 0
             | n == 0     = 1
             | otherwise  = sumDegsOf2(n-1) + degOf2(n)

nPerfect :: Integer -> [Integer]    --return array of first n perfect nums
nPerfect' :: Integer -> [Integer] -> Integer -> [Integer]
nPerfect' n perfs t | toInteger(length perfs) == n  = perfs
                    | toInteger(length perfs) < n   = nPerfect' n
                                                      (perfs ++ (if isprime(sumDegsOf2(t-1)) == 0
                                                         then []
                                                         else [(sumDegsOf2(t-1)) * (degOf2(t-1))]))
                                                      (t+1)
nPerfect 0 = []
nPerfect n = nPerfect' n [] 1
```

Задание 3. Близнецами называется пара натуральных чисел, каждое из которых равно сумме делителей другого числа. Так, например, числа 220 и 284 – близнецы (проверьте!). Написать программу для нахождения первых n пар близнецов.

```haskell
sumDivrs :: Integer -> Integer
sumDivrs' :: Integer -> Integer -> Integer -> Integer
sumDivrs' n i sum | i*i > n       = sum
                  | mod n i == 0  = sumDivrs' n (i+1) (sum + i + (div n i))
                  | otherwise     = sumDivrs' n (i+1) sum
sumDivrs n = sumDivrs' n 2 1

findTwin :: Integer -> Integer
findTwin n | sumDivrs(sumDivrs n) == n && sumDivrs n /= n  = sumDivrs n
           | otherwise                                     = 0

nTwins :: Integer -> [[Integer]]
nTwins' :: Integer -> [[Integer]] -> [Integer] -> Integer -> [[Integer]]
nTwins' n twins found t | toInteger(length twins) == n  = twins
                        | findTwin t /= 0 && notElem t found && notElem (findTwin t) found
                                                        = nTwins' n (twins ++ [[t, findTwin t]])
                                                          (found ++ [t] ++ [findTwin t]) (t+1)
                        | otherwise                     = nTwins' n twins found (t+1)
nTwins 0 = []
nTwins n = nTwins' n [] [] 2
```

Задание 4. В заданном списке строк найти самую длинную строку.

```haskell
longestStr :: [String] -> String
longestStr' :: [String] -> Int -> String -> String
longestStr' [] maxlen maxstr = maxstr
longestStr' (str:ls) maxlen maxstr | maxlen > length str   = longestStr' ls maxlen maxstr
                                   | otherwise             = longestStr' ls (length str) str
longestStr [] = ""
longestStr ls = longestStr' ls 0 ""
```

Задание 5. По заданному списку строк построить список строк, в котором содержатся те же строки, что и в исходном списке, но каждая вторая строка выброшена из списка (то есть в списке останутся только строки с нечетными номерами), а в каждой из оставшихся строк каждый второй символ также выброшен. Так, например, если аргументом программы является список строк

`["Близнецами", "называется", "пара", "натуральных", "чисел"]`

то результатом работы должен быть список строк

`["Бинцм", "пр", "чсл"]`

```
noEvenElts :: [a] -> [a]
noEvenElts' :: [a] -> Integer -> [a]
noEvenElts' [] i = []
noEvenElts' (elt:ls) i | odd i      = elt : noEvenElts' ls (i+1)
                       | otherwise  = noEvenElts' ls (i+1)
noEvenElts ls = noEvenElts' ls 1

noEvenEltsRec :: [[a]] -> [[a]]
noEvenEltsRec ls = noEvenElts(map noEvenElts ls)
```

Задание 6. По заданному списку строк построить список строк, в котором содержатся те же строки, что и в исходном списке, но выброшены строки, содержащие хотя бы одну цифру. Проверить, является ли некоторый символ `c` цифрой, можно с помощью вызова стандартной функции *Haskell* `Char.isDigit`.

```
hasNum :: String -> Bool
hasNum [] = False
hasNum (letr:str) | isDigit letr  = True
                  | otherwise  = hasNum str

noNumsStrs :: [String] -> [String]
noNumsStrs [] = []
noNumsStrs (str:ls) = (if (hasNum str) then [] else [str]) ++ noNumsStrs(ls)
```

Задание 7. Заданный числовой список разбить на подсписки из возрастающих подпоследовательностей максимальной длины рядом стоящих чисел. Так, например, если исходный список состоял из чисел [2, 7, 10, 8, 3, 4, 9, 1, 2, 0, 8, 3, 2, 5], то результатом работы программы должен быть следующий список списков: [[2, 7, 10], [8], [3, 4, 9], [1, 2], [0, 8], [3], [2, 5]].

```
growSeqs :: [Integer] -> [[Integer]]
growSeqs' :: [Integer] -> [Integer] -> [[Integer]]
growSeqs' [] seq = [seq]
growSeqs' (x:ls) seq | null seq || last seq < x  = growSeqs' ls (seq ++ [x])
                     | otherwise                 = seq : growSeqs' ls [x]
growSeqs ls = growSeqs' ls []
```

Задание 8. Для заданного вещественного числа x найти сумму числового ряда с общим членом $u_n = x^n / (n!+1)$. Суммирование производить, пока очередной член ряда не окажется по абсолютной величине меньше заданного числа.

```haskell
factl :: Integer -> Integer
factl 0 = 1
factl n = n * factl(n-1)

countUn :: Double -> Integer -> Double
countUn x n = x^n / (fromIntegral(factl n) + 1)

scarySum :: Double -> [Double]
scarySum' :: Double -> Integer -> [Double] -> [Double]
scarySum' x n [sum, step]
                | abs(countUn x n) < abs(x)  = [sum, fromIntegral n]
                | otherwise                  = scarySum' x (n+1) [sum + countUn x n, fromIntegral(n+1)]
scarySum x = scarySum' x 2 [x,2]
```

Тестирование:

```haskell
main = do
    --1
    print(seq2 1, seq2 2, seq2 3, seq2 4, seq2 5, seq2 10, seq2 50)
    print(seq1 1, seq1 2, seq1 3, seq1 4, seq1 5, seq1 10)
    --2
    print(nPerfect 0, nPerfect 1, nPerfect 3, nPerfect 4, nPerfect 8)
    --3
    print(nTwins 6)
    --4
    print(longestStr ["62", "Ali4", "Abracada~bra14", "Yeee5", "CadabraPokemon16"],
          longestStr [""], longestStr ["42"])
    --5
    print(noEvenEltsRec ["Twins", "are", "pair", "of", "natural", "numbers"])
    --6
    print(noNumsStrs ["62", "Ali4", " Abracada~bra ", "Yeee5", "CadabraPokemon"])
    --7
    print(growSeqs [1,2,3,4,7,5,4,3,6,8,5])
    --8
    print(scarySum 2, scarySum 4, scarySum 10)
```

```
PS C:\Users\admin\OneDrive\Рабочий стол\Haskell_V_semester> runhaskell 2_exercises.hs
(2,5,12,27,58,2037)
([],[6],[6,28,496],[6,28,496,8128],[6,28,496,8128,33550336,8589869056,137438691328,2305843008139952128])
[[220,284],[1184,1210],[2620,2924],[5020,5564],[6232,6368],[10744,10856]]
("CadabraPokemon16","","42")
["Tis","pi","ntrl"]
[" Abracada~bra ","CadabraPokemon"]
[[1,2,3,4,7],[5],[4],[3,6,8],[5]]
([2.0,2.0],[42.859999006583294,7.0],[21943.635191290403,22.0])
```