ІП-11 Панченко Сергій Віталійович

Варіант 1

Завдання

Визначте функції вищого порядку для виконання вказаних завдань. За необхідності завдання можна розширити. Наприклад у завданні 1.1 результат повертати у формі кортежів ('a',3) або (3,'a'), або ('a',3,"a3") тощо.
Визначте вказані функції в кожному з завдань: а) без застосування, б) із застосуванням вбудованих функцій (визначених у модулі Prelude).

1) Визначити довжину кожної неперервної послідовності тотожних елементів списку, напр.: "aaabbcaadddd"⇒ [(’a’,3), (’b’,2), (’c’,1), (’a’,2), (’d’,4)].
2) Циклічний лівий зсув списку на n позицій.

Код

```
module Lib
    ( oneOneA,
      oneOneB,
      twoOneA,
      twoOneB,
      someFunc
    ) where

someFunc :: IO ()
someFunc = putStrLn "someFunc"

myTakeWhile :: (a -> Bool) -> [a] -> [a]
myTakeWhile _ [] = []
myTakeWhile p (x:xs)
  | p x = x : myTakeWhile p xs
  | otherwise = []

myDropWhile :: (a -> Bool) -> [a] -> [a]
myDropWhile _ [] = []
myDropWhile p xs@(x:xs')
    | p x = myDropWhile p xs'
    | otherwise = xs

myLength :: [a] -> Int
myLength [] = 0
myLength (_:xs) = 1 + myLength xs

oneOneA :: Eq a => [a] -> [(a, Int)]
oneOneA [] = []
oneOneA (x:xs) = (x, myLength (myTakeWhile (== x) (x:xs))) :
oneOneA (myDropWhile (== x) xs)
```

```
oneOneB :: Eq a => [a] -> [(a, Int)]
oneOneB [] = []
oneOneB (x:xs) = (x, length (takeWhile (== x) (x:xs))) : oneOneB
(dropWhile (== x) xs)

myTake :: Int -> [a] -> [a]
myTake 0 _ = []
myTake n (x:xs) = x : myTake (n - 1) xs

myDrop :: Int -> [a] -> [a]
myDrop 0 l = l
myDrop n (_:xs) = myDrop (n - 1) xs

myMod :: Int -> Int -> Int
myMod _ 0 = error "Division by zero is undefined"
myMod x y
    | x < 0 = myMod (x + y) y
    | x < y = x
    | otherwise = myMod (x - y) y

twoOneA :: [a] -> Int -> [a]
twoOneA l n = myDropK l ++ myTakeK l
  where
    k = myMod n (myLength l)
    myDropK = myDrop k
    myTakeK = myTake k

twoOneB :: [a] -> Int -> [a]
twoOneB l n = dropK l ++ takeK l
  where
    k = mod n (myLength l)
    dropK = drop k
    takeK = take k
```

Код тестів

```
import Lib


import Test.HUnit


oneOneCases :: [(String, [(Char, Int)])]

oneOneCases = [

    ("aaaabbbcccbb", [('a', 4), ('b', 3), ('c', 3), ('b', 2)]),

    ("a aa aaa aaaa", [('a', 1), (' ', 1), ('a', 2), (' ', 1),
('a', 3), (' ', 1), ('a', 4)]),

    ("abcdefgh", [('a', 1), ('b', 1), ('c', 1), ('d', 1), ('e',
1), ('f', 1), ('g', 1), ('h', 1)])
```

```
  ]


-- Функція тестування для першого завдання.
-- Для зручності зробив вивід номера тестового варіанта, назву
createOneTestCases :: (Eq a, Show a) => [Char] -> (t -> a) -> [(t, a)] -> [Test]
createOneTestCases baseName func testCases = [TestCase (assertEqual (baseName ++ " " ++ show i) y (func x)) | (i, (x, y)) <- zip [0..] testCases]


oneOneATests :: [Test]
oneOneATests = createOneTestCases "OneOneA" oneOneA oneOneCases


oneOneBTests :: [Test]
oneOneBTests = createOneTestCases "OneOneB" oneOneB oneOneCases


twoOneCases :: [(([Int], Int), [Int])]
twoOneCases = [
    (([1, 2, 3, 4, 5], 0), [1, 2, 3, 4, 5]),
    (([1, 2, 3, 4, 5], 1), [2, 3, 4, 5, 1]),
    (([1, 2, 3, 4, 5], 2), [3, 4, 5, 1, 2]),
    (([1, 2, 3, 4, 5], 3), [4, 5, 1, 2, 3]),
    (([1, 2, 3, 4, 5], 4), [5, 1, 2, 3, 4]),
    (([1, 2, 3, 4, 5], 5), [1, 2, 3, 4, 5]),
    (([1, 2, 3, 4, 5], 6), [2, 3, 4, 5, 1]),
    (([1, 2, 3, 4, 5], 7), [3, 4, 5, 1, 2]),
    (([1, 2, 3, 4, 5], 8), [4, 5, 1, 2, 3]),
    (([1, 2, 3, 4, 5], 9), [5, 1, 2, 3, 4]),
    (([1, 2, 3, 4, 5], 10), [1, 2, 3, 4, 5])
  ]


-- Функція тестування для другого завдання.
```

```
createTwoTestCases :: (Eq a, Show a) => String -> (c -> b -> a) ->
[((c, b), a)] -> [Test]

createTwoTestCases baseName func testCases = [TestCase
(assertEqual (baseName ++ " " ++ show i) (func x y) z) | (i, ((x,
y), z)) <- zip [0..] testCases]


-- Створюємо тести для twoTenA

twoOneATest :: [Test]

twoOneATest = createTwoTestCases "twoTenA" twoOneA twoOneCases


-- Створюємо тести для twoTenB

twoOneBTest :: [Test]

twoOneBTest = createTwoTestCases "twoTenB" twoOneB twoOneCases


-- Створюємо тестовий список

tests :: Test

tests = TestList (oneOneATests ++ oneOneBTests ++ twoOneATest ++
twoOneBTest)


-- Головна функція, виводить результати тестування.

main :: IO ()

main  = do

    result <- runTestTT tests

    print (if failures result > 0 then "Failure" else "Success")
```

Результати тестування

```
/usr/local/bin/stack test Lab2:test:Lab2-test --test-arguments --
color

Lab2-0.1.0.0: unregistering (local file changes:

.stack-work/dist/x86_64-linux-tinfo6/ghc-9.6.3/build/autogen/
Paths_Lab2.hs)

Lab2> build (lib + test)
```

```
Preprocessing library for Lab2-0.1.0.0..
Building library for Lab2-0.1.0.0..
Preprocessing test suite 'Lab2-test' for Lab2-0.1.0.0..
Building test suite 'Lab2-test' for Lab2-0.1.0.0..
Lab2> copy/register
Installing library in
/home/sideshowbobgot/university/FunctionalProgrammingHaskell/Labs/
Lab2/.stack-work/install/x86_64-linux-tinfo6/
dfa84d1f673cb7aef8dd197ae01305e2729e853b43dcaec5ae5682fbb979a8e6/9
.6.3/lib/x86_64-linux-ghc-9.6.3/Lab2-0.1.0.0-
HhyekjqdsrU9YQkdSkpBX8
Installing executable Lab2-exe in
/home/sideshowbobgot/university/FunctionalProgrammingHaskell/Labs/
Lab2/.stack-work/install/x86_64-linux-tinfo6/
dfa84d1f673cb7aef8dd197ae01305e2729e853b43dcaec5ae5682fbb979a8e6/9
.6.3/bin
Registering library for Lab2-0.1.0.0..
Lab2> test (suite: Lab2-test, args: --color)


Cases: 28  Tried: 28  Errors: 0  Failures: 0
"Success"




Lab2> Test suite Lab2-test passed
Completed 2 action(s).


Process finished with exit code 0
```