Tugas 4

Tags

Identitas

- Nama: Fahdii Ajmalal Fikrie
- NPM: 1906398370

Soal 1

Berikut adalah definisi fungsi length

```
length :: [a] -> Int
length [] = 0
length (x:xs) = 1 + length xs
```

Buatlah definisi fungsi length baru menggunakan map dan fold!

```
length' :: [a] -> Int
length' [] = 0
length' (x:xs) = 1 + length' xs

newLength :: [a] -> Int
newLength xs = foldr f 0 xs
    where f x y = y + 1
```

Soal 2

Diberikan fungsi

```
addUp ns = filter greaterOne (map addOne ns)
```

dimana

```
greaterOne n = n > 1
```

dan

```
addOne n = n + 1
```

definisikan ulang fungsi tersebut `(fun1, fun2)` dengan `filter` sebelum `map`, misalnya `addUp ns = map fun1 (filter fun2 ns)`

```
greaterOne :: Int -> Bool
greaterOne n = n > 1

addOne :: Int -> Int
addOne n = n + 1

addUp :: [Int] -> [Int]
addUp ns = filter greaterOne (map addOne ns)

fun1 :: Int -> Int
fun1 n = n + 1

fun2 :: Int -> Bool
fun2 n = n >= 1

newAddUp :: [Int] -> [Int]
newAddUp ns = map fun1 (filter fun2 ns)
```

Soal 3

Definisikan fungsi *sum of the squares* dari 1 sampai n dengan cara berikut!

a. map dan fold

```
sumOfSquares1 :: Int -> Int
sumOfSquares1 n = foldr (\x y -> x*x + y) 0 [1..n]
```

b. fold dan list comprehension

```
sumOfSquares2 :: Int -> Int
sumOfSquares2 n = sum [x * x | x <- [1..n]]
```

c. Jelaskan perbedaan dua pendekatan tersebut!

> 💡 Perbedaan terletak pada apabila menggunakan foldr, pendekatannya menggunakan rekursi, sedangkan dengan list comprehension, yakni menggunakan list comprehension
>
> Saya masih belum paham cara integrasiin map dengan foldr, saya juga bingung menjelaskan perbedaannya bagaimana :(

Soal 4

Definisikan fungsi yang mengembalikan jumlah bilangan kelipatan 5 dalam sebuah list!

```
isMultipleOfFive :: Int -> Bool
isMultipleOfFive n = (n `mod` 5) == 0

multipleOfFive :: [Int] -> Int
multipleOfFive xs = length (filter isMultipleOfFive xs)
```

Soal 5

Definisikan fungsi `total` dimana `total :: (Int -> Int) -> (Int -> Int)` sehingga `total f` adalah fungsi ketika mendapat nilai `n` memberikan total dari f 0 + f 1 + ... + f n!

```
(Pass)
```

Soal 6

Buatlah fungsi reverse dengan menggunakan foldr!

```
reverse' :: [Int] -> [Int]
reverse' xs = foldr (\f g x -> g (f : x)) id xs []
```

Soal 7

Uraikan langkah evaluasi dari ekspresi berikut!
`[x+y|x<-[1..4], y<-[2..4], x > y]`

💡 Ini adalah bentuk list comprehension di mana dilakukan merging pada list x & y. Merging dilakukan dengan melakukan filtering (x > y) lalu menambahkan kedua index masing-masing dari x dan y

Soal 8

Buatlah fungsi noUpperAndIdent yang menghapus seluruh karakter kapital dan karakter non-alfabet dari argumen String yang diberikan! (Hint: Gunakan library function elem dan isUpper)

```
noUpperIndent :: String -> String
noUpperIndent = filter (\x -> (x `elem` ['a'..'z']))
```

Soal 9

Buatlah struktur data yang menyatakan representasi data dalam bentuk tree dengan elemen data berada hanya pada leaves saja.

```
data Tree a
    = Nil
    | Node a (Tree a) (Tree a) -- value, left child, right sibling
```

Soal 10

Buatlah fungsi foldTree, sehingga dengan menggunakan fungsi foldTree (+) , bila diberikan parameter sebuah tree (mengikuti nomor 9), maka bisa menjumlahkan seluruh
elemen leaves pada tree tersebut .

```
(Pass)
```