Tugas Pemrograman 4

Higher Order Function

Waktu Pengerjaan	: Satu pekan
Tipe Tugas	: Programming Individu,
Tipe Submission	: Submission berkas, Tugas4-NPM-NamaMahasiswa.zip
Due Date	: Lihat SCeLe Kuliah

Penjelasan Umum

- Baca perintah soal dengan teliti, perhatikan contoh input dan output pada soal.
- Untuk soal-soal pemrograman berikut, masukkan implementasi tersebut kedalam satu file .hs dengan format penamaan Tugas4-NPM-NamaMahasiswa.hs
- Untuk soal-soal evaluasi dan penjelasan program yang dibuat, jelaskan dalam bentuk .pdf dengan format penamaan Tugas4-NPM-NamaMahasiswa.pdf
- Kumpulkan kedua file tersebut dengan melakukan zip terlebih dahulu, untuk penamaan .zip adalah Tugas4-NPM-NamaMahasiswa.zip
- Apabila terdapat hal yang kurang jelas, silahkan tuliskan tambahan keterangan atau bertanya kepada asisten.

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`!

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)
   ```
   Contoh eksekusi:
   ```
   Prelude> addUp [0,1,2,3]
   [2,3,4]
   ```

3. Definisikan fungsi *sum of the squares* dari *1* sampai *n* dengan cara berikut!
   ```
   Prelude> sumOfSquares 3
   14
   ```
   a. map dan fold
   b. fold dan *list comprehension*
   c. Jelaskan perbedaan dua pendekatan tersebut!

4. Definisikan fungsi yang mengembalikan jumlah bilangan kelipatan 5 dalam sebuah list!
   ```
   Prelude> multipleOf5 [1,2,3,4,5,6,7,8,9,10]
   2
   ```

5. Definisikan fungsi total dimana `total :: (Int -> Int) -> (Int -> Int)` sehingga `total f` adalah fungsi ketika mendapat nilai *n* memberikan *total* dari $f0 + f1 + ... + fn$!
   ```
   Prelude> total (+1) 3
   9
   ```

6. Buatlah fungsi reverse dengan menggunakan foldr!
   ```
   Prelude> reverse [1,2,3,4,5]
   [5,4,3,2,1]
   ```

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

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`)
   ```
   Prelude> noUpperAndIdent "FunPro MOOC"
   "unro"
   ```

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

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 .

Selamat Mengerjakan!

Terakhir diedit 20 Sept 11:45 oleh Ade Azurat