FACULTY OF COMPUTER SCIENCE

Functional Programming Semester Gasal 2019 Term 1

Tambahan Ide Latihan Persiapan Ujian Tengah Semester

Lakukan latihan paling tidak sehari sebelum ujian (istirahat yang cukup semalam sebelumnya) Lakukan commit dan push untuk setiap upaya belajar atau latihan yang dilakukan walaupun masih salah atau belum berhasil.

Beberapa soal ujian akan menuntut latihan terkait sebelumnya. Penilaian jawaban soal pada ujian akan dipengaruhi apakah ada latihan terkait soal tersebut atau tidak dan apakah peserta melakukan variasi latihan atau hanya sekedar me-run definisi. Berlatihlah dengan melakukan variasi-variasi dan mengobservasi hasilnya. Akan lebih baik lagi bila peserta menuliskan komentar pemahaman atau hasil observasinya ketika berlatih. Walaupun masih kurang atau masih salah tidak apa-apa karena dalam apresiasi belajar, proses belajar adalah bagian yang paling penting. Akan lebih diapresiasi bila melakukan banyak kesalahan dan terus berlatih serta mengobservasi hasil latihan nya dan mengupayakan perbaikannya (walau masih belum berhasil). Hal ini menunjukkan kemampuan *meta cognitive* yang akan terlihat memberikan manfaat di kemudian hari. Hasil observasi yang dituliskan tersebut dapat membuat peserta mendapat nilai maksimal pada soal terkait. Tanpa menunjukkan latihan terkait (commit pada repository) nilai pada soal tersebut bisa berkurang antara 40%-20%.

Berikut beberapa tambahan latihan terutama bagi rekan2 yang sebelumnya masih bingung perlu latihan apa.

Selamat berlatih!

Contoh Kasus: Bahasa Aritmatika Sederhana

Pada Buku acuan, Haskell School of Expression Bab 7, dibahas tentang struktur data tree dan menyatakan bagaimana struktur data tersebut juga digunakan untuk menyusun bahasa sederhana untuk operasi aritmatika. Dipelajari juga bagaimana *Higher Order Function* tidak hanya dapat didefinisikan untuk struktur data list, namun juga untuk struktur data lain seperti tree ataupun bahkan struktur data Ekspresi.

Latihan Persiapan (lakukan commit push untuk setiap upaya kerja):

1. Rancanglah, *higher order function* yang bekerja pada struktur data ekspresi dan memiliki semantik seperti: Fungsi map pada list, Fungsi fold pada tree
2. Pada buku dan contoh sebelumnya, fungsi evaluasi didefinisikan secara rekursif. Gunakan *higher order function* yang baru dibuat untuk mendefinisikan fungsi evaluasi tersebut. Pastikan memiliki makna semantik yang sama dengan definisi sebelumnya.
3. Tambahkan konstruksi Let pada struktur data Ekspresi menjadi:

```
> data Expr = C Float | Expr :+ Expr | Expr :- Expr
>           | Expr :* Expr | Expr :/ Expr
>           | V String
>           | Let String Expr Expr
>      deriving Show
```

4. Sesuaikan fungsi evaluasi versi sebelumnya (versi definisi rekursif).
5. Sesuaikan fungsi evaluasi versi menggunakan fold.

6. Bandingkan proses modifikasi yang perlu dilakukan.
7. Gunakan fold untuk mendefinisikan fungsi terhadap struktur data tersebut seperti misalnya menghitung jumlah konstanta yang digunakan, jumlah operator, jumlah variable, melihat apakah ada pembagian by zero atau tidak. Dll
8. Saat ini deklarasi Let hanya bisa menyatakan satu buah variable setiap deklarasi. Bila ada beberapa variable harus dinyatakan secara nested. Mungkinkan kita membuat deklarasi Let menyimpan list of variable? Bagaimana pengaruhnya terhadap implementasi sebelumnya baik versi rekursif maupun versi higher order function?
9. Misalkan anda ingin tambahkan kemampuan pemanggilan fungsi yang menerima list of ekspresi dan menghasilkan sebuah ekspresi. Bagaimana mendeklarasikan tipe nya dan evaluasinya?
10. Apakah anda sempat mempelajari tentang teknik *De Bruijn Index* yang efisien untuk variable binding (penggunaan Let untuk variable dan parameter binding pada pemanggilan fungsi)? Bagaimana menerapkan nya?

Contoh Kasus: *Imperative Robot Language*

Pada buku acuan terdapat studi kasus Domain Specific Language: Imperatif Robot Language. Sebuah bahasa untuk memerintahkan pergerakan robot. Pahami implementasi penyusunan bahasa tersebut dalam haskell dan bagaimana Monad digunakan untuk menyederhanakan bagi pengguna bahasa robot tersebut.

Latihan Persiapan (lakukan commit push untuk setiap upaya kerja):

1. Apakah kita bisa menambahkan pada RobotState informasi tentang *Energy*? Misalnya robot hanya bisa bergerak bisa masih ada energi-nya dan energinya akan terpakai. Versi awal setiap gerakan mengkonsumsi energi yang sama, versi selanjutnya tiap gerakan memiliki kebutuhan energi nya masing-masing termasuk penDown misalnya. Fungsi tambahan apa lagi yang dibutuhkan?
2. Pelajari implementasi dari `moven`. Apakah tipe dari fungsi `mapM_` ? apa makna dari fungsi tersebut?
3. Perhatikan implementasi fungsi spiral yang dicontohkan. Apakah bisa dibuatkan perintah baru **`forloop`** misalnya sehingga programmer robot bisa mengimplementasikan fungsi spiral yang sama dengan cara yang lebih mudah misalnya seperti berikut ini saja:

```
-- spiral movement using for forloop

> spiral = forLoop [1..20] action
>  where action = \i -> (turnRight >> moven i >> turnRight >> moven i)

-- zigzag with for loop

> zigzag = forLoop [1..10] action
>    where action  = \i -> (moven 5 >> turnRight >> move >> turnRight
>                           >> moven 5 >> turnLeft  >> move >> turnLeft
```