IF1210 Dasar Pemrograman

Operasi-Operasi pada List dengan Elemen Sederhana

Tim Pengajar IF1210

Sekolah Teknik Elektro dan Informatika

Review – List dengan elemen sederhana – Notasi Haskell

- Definisi type list

```
-- DEFINISI DAN SPESIFIKASI LIST

-- type List of <type_elemen>: [ ] atau [e o List]
-- Definisi type List of <type_elemen>
-- Basis: List of <type_elemen> kosong adalah list of <type_elemen>
-- Rekurens: List tidak kosong dibuat dengan menambahkan sebuah
-- elemen bertype <type_elemen> di awal sebuah list

-- type List of <type_elemen>: [ ] atau [List • e]
-- Definisi type List of <type_elemen>:
-- Basis: List of <type_elemen> kosong adalah list of <type_elemen>
-- Rekurens: List tidak kosong dibuat dengan menambahkan sebuah
-- elemen bertype <type_elemen> di akhir sebuah list
```

Review – List dengan elemen sederhana – Notasi Haskell

- Konstruktor

```
-- DEFINISI DAN SPESIFIKASI KONSTRUKTOR

konso :: <type_elemen> -> [<type_elemen>] -> [<type_elemen>]
-- konso(e,l) menghasilkan sebuah list dari e (sebuah
-- elemen) dan l (list of elemen),
-- dengan e sebagai elemen pertama: e o l -> l'
-- REALISASI
konso e l = [e] ++ l

konsDot :: [<type_elemen>] -> <type_elemen> -> [<type_elemen>]
-- konsDot(l,e) menghasilkan sebuah list dari l (list of
-- elemen) dan e (sebuah elemen),
-- dengan e sebagai elemen terakhir: l • e -> l'
-- REALISASI
konsDot l e = l ++ [e]
```

Review – List dengan elemen sederhana – Notasi Haskell

- Selektor

```
-- DEFINISI DAN SPESIFIKASI SELEKTOR

-- head : [<type_elemen>] -> <type_elemen>
-- head(l) menghasilkan elemen pertama list l, l tidak kosong

-- tail : [<type_elemen>] -> [<type_elemen>]
-- tail(l) menghasilkan list tanpa elemen pertama list l,
-- l tidak kosong

-- last : [<type_elemen>] -> <type_elemen>
-- last(l) menghasilkan elemen terakhir list l, l tidak kosong

-- init : [<type_elemen>] -> [<type_elemen>]
-- init(l) menghasilkan list tanpa elemen terakhir list l,
-- l tidak kosong
```

Review – List dengan elemen sederhana – Notasi Haskell

- Predikat dasar

```
-- DEFINISI DAN SPESIFIKASI PREDIKAT

isEmpty :: [<type_elemen>] -> Bool
-- isEmpty(l) true jika list of elemen l kosong
-- REALISASI
isEmpty l = null l

isOneElmt :: [<type_elemen>] -> Bool
-- isOneElmt(l) true jika list of integer l hanya
-- mempunyai satu elemen
-- REALISASI
isOneElmt l = (length l) == 1
```

Menghitung banyaknya elemen (nbElmt)

- **nbElmt**:: [<type_elemen>] -> Int
- Memanfaatkan definisi list terdiri atas FirstElmt + Tail
- Rekursif
  - Basis 0: list kosong, nbElmt = 0
  - Rekurens: nbElmt(l) = 1 + nbElmt(tail(l))
- Realisasi

Contoh:

```
=> nbElmt []
0
=> nbElmt [1,2,3]
3
```

```
-- DEFINISI DAN SPESIFIKASI
nbElmt :: [<type_elemen>] -> Int
-- nbElmt l menghasilkan banyaknya elemen list, nol
-- jika list kosong

-- REALISASI
nbElmt l = if (isEmpty l) then 0       -- Basis
           else 1 + (nbElmt (tail l)) -- Rekurens
```

Menghitung banyaknya elemen (nbElmt)

```
-- DEFINISI DAN SPESIFIKASI
nbElmt :: [<type_elemen>] -> Int
-- nbElmt l menghasilkan banyaknya elemen list, nol
-- jika list kosong

-- REALISASI
nbElmt l = if (isEmpty l) then 0       -- Basis
           else 1 + (nbElmt (tail l)) -- Rekurens
```

```
-- DEFINISI DAN SPESIFIKASI, LIST OF INTEGER
nbElmt :: [Int] -> Int
-- nbElmt l menghasilkan banyaknya elemen list, nol
-- jika list kosong

-- REALISASI
nbElmt l = if (isEmpty l) then 0       -- Basis
           else 1 + (nbElmt (tail l)) -- Rekurens
```

```
-- DEFINISI DAN SPESIFIKASI, LIST OF CHARACTER
nbElmt :: [Char] -> Int
-- nbElmt l menghasilkan banyaknya elemen list, nol
-- jika list kosong

-- REALISASI
nbElmt l = if (isEmpty l) then 0       -- Basis
           else 1 + (nbElmt (tail l)) -- Rekurens
```

Mengecek keanggotaan sebuah elemen (isMember)

- Fungsi **isMember** mengecek apakah suatu elemen adalah member dari suatu list

```
isMember::<type_elemen> -> [<type_elemen>] -> Bool
{- isMember x l menghasilkan true jika x adalah elemen
   dari l -}
```

- Contoh:

```
=> isMember 5 []
False
=> isMember 5 [1,2,3]
False
=> isMember 5 [4,5,6]
True
```

Rekursif

- **Basis**: jika list kosong maka nilai keluaran (output) adalah false (basis 0)
- **Rekurens:**
  Jika nilai elemen pertama (atau terakhir) dari list adalah x, maka output adalah true. Tapi jika bukan x, maka tail (atau head) harus dicek.

Fungsi isMember

```
-- Definisi FirstElmt+Tail
isMember1 :: <type_elemen> -> [<type_elemen>] -> Bool
-- isMember1 x l true jika x adalah elemen list l
-- REALISASI isMember1 menggunakan head dan tail
isMember1 x l = if (isEmpty l) then False  -- Basis
                else if (head l) == x then True
                     else (isMember1 x (tail l))  -- Rekurens
```

```
-- Definisi Head+LastElmt
isMember2 :: <type_elemen> -> [<type_elemen>] -> Bool
-- isMember2 x l true jika x adalah elemen list l
-- REALISASI isMember2 menggunakan init dan last
isMember2 x l  = if (isEmpty l) then False  -- Basis
                 else if (last l) == x then True
                      else (isMember2 x (init l))  -- Rekurens
```

Menyalin (copy) list

- Proses mengambil satu persatu elemen dari List sumber dan membuat elemen baru untuk List target hasil copy

```
copy:: [<type_elemen>] -> [<type_elemen>]
{- copy l menyalin satu per satu elemen l untuk membentuk
   list baru hasil copy -}
```

- Contoh:

```
=> copy []
[]
=> copy [2,3,4]
[2,3,4]
```

Rekursif

- **Basis**: isEmpty → memberi list kosong [ ]
- **Rekurens**: mengambil elemen pertama list sumber kemudian memasukkannya sebagai elemen pertama list target ATAU mengambil elemen terakhir dari list sumber kemudian memasukkannya sebagai elemen terakhir list target

Fungsi copy

```
-- DEFINISI DAN SPESIFIKASI
copy :: [<type_elemen>] -> [<type_elemen>]
-- copy(l) menghasilkan list yang identik dengan list asal

-- REALISASI (versi 1: menggunakan konso)
copy l = if (isEmpty l) then []                    -- Basis
         else (konso (head l) (copy (tail l)))  -- Rekurens

-- REALISASI (versi 2: menggunakan konsDot)
copy l = if (isEmpty l) then []                     -- Basis
         else (konsDot (copy (init l)) (last l)) -- Rekurens
```

Mengecek apakah 2 list sama atau tidak (isEqual)

- **isEqual::** [<type_elemen>] -> [<type_elemen>] -> Bool

- Contoh:

```
=> isEqual [] []
True
=> isEqual [] [a]
false
=> isEqual [a] []
False
=> isEqual [a,b,c] [a,b,c]
True
```

Rekursif

- **Basis**: Jika keduanya kosong maka true, jika hanya salah satu kosong maka false
- **Rekurens**: Cek elemen pertama dari kedua list dan kemudian cek tail kedua list tersebut ATAU cek elemen terakhir dari kedua list dan kemudian cek head kedua list

Fungsi isEqual

```
-- DEFINISI DAN SPESIFIKASI
isEqual :: [<type_elemen>] -> [<type_elemen>] -> Bool
-- isEqual l1 l2 true jika semua elemen list l1 sama dengan l2:
-- sama urutan dan sama nilai per elemen pada posisi yang sama

-- REALISASI
isEqual l1 l2
   | (isEmpty l1) && (isEmpty l2) = True      -- Basis
   | (isEmpty l1) && not (isEmpty l2) = False -- Basis
   | not (isEmpty l1) && (isEmpty l2) = False -- Basis
   | not (isEmpty l1) && not (isEmpty l2) = -- Recc
       (
          (head l1)==(head l2) && (isEqual (tail l1) (tail l2))

       )
```

Menggabung (konkatenasi) 2 List

```
konkat:: [<type_elmt>] -> [<type_elmt>] -> [<type_elmt>]
-- konkat l1 l2 menghasilkan konkatenasi l1 dan l2
-- dengan list l2 "sesudah" list l1
```

- Contoh:

```
=> konkat [] []
[]
=> konkat [a] [b,c]
[a,b,c]
```

- Rekursif terhadap l1
  - Basis: l1 adl. list kosong, maka output adl. l2
  - Rekurens: mengambil elemen pertama dari l1 dan menggabungkannya dengan konkatenasi antara tail(l1) dan l2.

Fungsi konkat

```
-- DEFINISI DAN SPESIFIKASI
konkat:: [Int] -> [Int] -> [Int]
-- konkat(L1,L2) menghasilkan konkatenasi l1 dan l2,
-- dengan list l2 "sesudah" list l1

-- REALISASI
konkat l1 l2 =
     if (isEmpty l1) then l2  -- Basis
     else -- Rekurens
         (konso (head l1) (konkat (tail l1) l2))
```

Ambil elemen ke-N

- **`elmtKeN :: Int -> [<type_elemen>] -> <type_elemen>`**
- Prekondisi: Parameter integer bernilai besar dari 0, list tidak kosong, dan parameter integer <= jumlah elemen list
- Contoh:

  `=> elmtKeN 0 []` → tidak terdefinisi, karena dalam spek, list input harus tidak kosong dan N ≥ 1, N ≤ jumlah elemen List

  ```
  => elmtKeN(1,[a,b,c])
  a
  ```

- Rekurens dilakukan terhadap N (dikurangi 1) dan List (tail-nya)

elmtKeN

```
-- DEFINISI DAN SPESIFIKASI
elmtKeN:: Int -> [<type_elemen>] -> <type_elemen>
-- elmtKeN n l mengembalikan elemen ke-n dari list l
-- Prekondisi: n>0 dan n<=jumlah elemen l,
-- list l tdk kosong

-- REALISASI
elmtKeN  n l = if (n == 1) then (head l)  -- Basis
               else -- Rekurens
                  (elmtKeN (n-1) (tail l))
```

Apakah X adalah elemen ke N?

```
isXElmtkeN ::
      <type_elemen> -> Int -> [<type_elemen>] -> Bool
```

- Cara 1 (Rekursif):
  - Basis 1: N=1, dan kemudian mengecek apakah FirstElmt(L)=X
  - Rekurens: N dikurangi 1, X tetap dan L diambil Tail-nya
- Cara 2:
  - Menggunakan fungsi antara elmtKeN(N,L), mengecek apakah elmtKeN(N,L) = X ?
- Buatlah sebagai latihan fungsi **isXElmtkeN**.

Latihan 1

a. Tuliskan realisasi dari fungsi **countFactorOfX** berikut ini:

```
countFactorOfX :: Int -> [Int] -> Int
{- countFactorOfX n l mengembalikan banyaknya kemunculan bilangan yang
   merupakan faktor dari n pada l -}
```

b. Tuliskan realisasi dari fungsi **delNthElmt** berikut ini:

```
delNthElmt :: Int -> [Char] -> [Char]
{- delNthElmt n l menghilangkan elemen ke-n dari l.
   Asumsi: n lebih kecil atau sama dengan jumlah elemen l;
   l tidak kosong -}
```

List of Integer

- **List of integer** adalah list yang elemennya berupa integer

Element	Integer
List of elemen	List of integer

- Contoh
  - Konso: elemen, List → List *{untuk list of elemen}*
  - Konso: integer, List of integer → List of integer *{untuk list of integer}*

Nilai Maksimum

- Fungsi menghasilkan elemen bernilai maksimum dari list bilangan integer yang tidak kosong
- **`maxlist :: [Int] -> Int`**
- Rekursif
  - Basis list 1 elemen: jika elemen list berjumlah satu maka ambil nilai terakhir dari list. Basis 1 digunakan karena jika list kosong maka nilai maksimum tidak terdefinisi
  - Rekurens: membandingkan nilai elemen terakhir list dengan nilai maksimum dari head list

```
-- DEFINISI DAN SPESIFIKASI
maxList:: [Int] -> Int
-- maxList l mengembalikan elemen terbesar pada l
-- prekondisi: list tidak kosong
-- REALISASI
maxList l = if isOneElmt l then last l -- Basis 1
            else -- rekurens
                 (max2 (last l) (maxList (init l)))
```

Penjumlahan Dua List Integer

- Fungsi untuk menjumlahkan dua list integer yang hasilnya disimpan dalam satu list integer. Asumsi: kedua list input memiliki dimensi (jumlah elemen) yang sama.
- **`listPlus`**`:: [Int] -> [Int] -> [Int]`
- Rekursif
  - Basis list kosong: Jika list kosong maka list output adalah [ ]
  - Rekurens: mengambil elemen pertama dari kedua list, menjumlahkan kedua elemen tadi kemudian memasukkannya sebagai elemen pertama dari list output

```
-- DEFINISI DAN SPESIFIKASI
listPlus:: [Int] -> [Int] -> [Int]
-- listPlus(l1,l2) mengembalikan list dengan elemen
-- hasil penjumlahan elemen di l1 dan l2.
-- prekondisi: l1 dan l2 memiliki jumlah elemen sama
-- REALISASI
listPlus l1 l2 = if isEmpty l1 then [] -- Basis 0
                 else -- rekurens
                    konso ((head l1) + (head l2))
                          (listPlus (tail l1) (tail l2))
```

Kemunculan Nilai Maks

- Fungsi menghasilkan nilai maksimum dan jumlah kemunculan nilai maksimum tsb pd list bilangan integer

```
maxNb: [Int] → (Int,Int)
```

- Contoh: `=> maxNb [11,3,4,5,11,6,11]`

  `(11,3)`

- Rekursif
  - Basis (Basis 1): List dgn satu elemen e menghasilkan <e,1>
  - Rekurens:

Jika m adalah nilai maksimum dari Tail(Li) Nilai maks: m; Jumlah kemunculan: n
dan n adalah jumlah kemunculan m pada Tail(Li)
maka ketika memeriksa e, ada 3 kemungkinan:
m < e: nilai maksimum diganti yang baru (m ← e), n=1
m = e: nilai maksimum tetap m, nilai kemunculan n ditambah 1
m > e: nilai maksimum tetap m, nilai kemunculan tetap n

Fungsi maxNb

```
-- DEFINISI DAN SPESIFIKASI
maxNb :: [Int] -> (Int,Int)
-- maxNb(li) menghasilkan <nilai max, kemunculan nilai max>
-- dari suatu list of integer li: <m,n> dengan m adalah nilai
-- maksimum di li dan n adalah jumlah kemunculan m dalam li
-- REALISASI
maxNb  li = if (isOneElmt li) then (head li,1) -- Basis
            else -- Rekurens
                let (m,n) = maxNb (tail li) in
                   if (m < head li) then (head li,1)
                   else if (m > (head li)) then (m,n)
                        else (m,n+1)
```

Latihan 2

Tuliskan definisi, spesifikasi, dan realisasi dari:

a. Fungsi **sumIsiList** menghitung hasil penjumlahan dari seluruh elemen sebuah list of integer l yang tidak kosong.

b. Fungsi **filterGanjil** melakukan filtering terhadap sebuah list of integer li sehingga menghasilkan list dengan elemen yang hanya terdiri atas bilangan ganjil yang muncul di li. Diasumsikan semua elemen li adalah bilangan integer positif atau 0. li mungkin kosong.

List of Character

- **List of Character** adalah list yang elemennya berupa character

Element	Character
List of elemen	Teks { List of character }

- Contoh
  - Konso: elemen, List → List *{ untuk list of elemen }*
  - Konso: character, Teks →Teks *{ untuk list of character }*

Hitung A

- **`nbA :: [Char] -> Int`**
- Contoh:
```
=> nbA ['b','c','a','d','a','n','a']
3
```
- Rekursif
  - Basis {basis 0}: teks kosong, output: 0
  - Rekurens: Periksa huruf pertama dari teks, jika ‘a’ maka 1 ditambahkan ke nilai kemunculan ‘a’ pada tail, jika bukan ‘a’ maka 0 ditambahkan ke nilai kemunculan ‘a’ pada tail

```
-- DEFINISI DAN SPESIFIKASI
nbA :: [Char] -> Int
-- nbA(lc) menghasilkan banyaknya kemunculan huruf 'A'
-- di teks lc
-- REALISASI
nbA lc = if isEmpty lc then 0  -- Basis
         else -- Rekurens
              nbA (tail lc) + (if (head lc)=='A' || (head lc)=='a'
                               then 1 else 0)
```

Latihan 3

Tuliskan definisi, spesifikasi, dan realisasi dari fungsi **isEqFront** menerima masukan 2 buah list of character yang tidak kosong, misalnya T1 dan T2 dan menghasilkan true jika potongan awal list T2 mengandung T1 (dengan panjang dan urutan karakter yang sama). Banyaknya elemen T2 selalu lebih dari atau sama dengan T1.
Contoh:

T1: [’a’, ’b’, ’c’]	T2: [’a’, ’b’, ’c’, ’d’, ’e’]	Hasil: true
T1: [’a’, ’b’, ’c’]	T2: [’a’, ’b’, ’c’]	Hasil: true
T1: [’a’, ’b’, ’c’]	T2: [’a’, ’b’, ’a’, ’b’, ’c’, ’d’]	Hasil: false
T1: [’a’, ’b’, ’c’]	T2: [’a’, ’b’, ’d’, ’a’, ’b’, ’c’]	Hasil: false

Pada dasarnya semua jenis list dikelola dengan cara yang sama

- Menghitung kemunculan sebuah elemen pada list
  - List of integer
    - `nbXInt ::  Int -> [Int] -> Int`
  - List of character
    - `nbC ::  Char -> [Char] -> Int`

Menghitung Kemunculan X

- Rekurens
  - Basis: untuk list kosong, kemunculan adalah 0
  - Rekurens: dicek apakah elemen pertama adalah X, jika iya maka nilai 1 ditambahkan pada hasil penghitungan kemunculan X pada tail dari list, jika tidak maka nilai 0 yang ditambahkan

```
nbX x l = if isEmpty l then 0  -- Basis
          else  -- rekurens
            (if ((head l) == x) then 1
                                else 0)
             + nbX x (tail l)
```

List of Elemen

```
nbX x l = if isEmpty l then 0  -- Basis
          else  -- rekurens
            (if ((head l) == x) then 1
                                else 0)
              + nbX x (tail l)
```

List of Integer

```
nbXInt x li = if isEmpty li then 0  -- Basis
              else  -- rekurens
                (if ((head li) == x) then 1
                                     else 0)
                  + nbXInt x (tail li)
```

List of Character

```
nbC x lc = if isEmpty lc then 0  -- Basis
           else  -- rekurens
             (if ((head lc) == x) then 1
                                  else 0)
               + nbC x (tail lc)
```

Latihan 4

Tuliskan realisasi dari fungsi **isOrdered** berikut ini:

```
isOrdered :: [Int] -> Bool
{- isOrdered l menghasilkan true jika elemen-elemen pada l
   terurut membesar, false jika tidak.
   Prekondisi: l tidak kosong -}
```

Latihan 5

Tuliskan realisasi dari fungsi **mergeList** berikut ini:

```
mergeList:: [Int] -> [Int] -> [Int]
{- mergeList li1 li2 menghasilkan list of integer yang
   merupakan hasil penggabungan li1 dan li2, dan tetap
   terurut membesar.
   Prekondisi: li1 dan li2 adalah list terurut membesar
   dan mungkin kosong -}
```

Latihan 6

Tuliskan realisasi dari fungsi **splitList** berikut ini:

```
splitList:: [Int] -> ([Int],[Int])
{- splitList li menghasilkan 2 list of integer, list
   pertama memuat bilangan positif dan 0 yang merupakan
   elemen dari li (dengan urutan kemunculan yang tidak
   berubah), sedangkan list kedua memuat bilangan negatif
   elemen li.
   Prekondisi: li mungkin kosong -}
```

Bahan

- Diktat “Dasar Pemrograman, Bag. Pemrograman Fungsional” oleh Inggriani Liem, revisi Februari 2014

Latihan Tambahan

Soal 1

Tuliskan realisasi dari fungsi **isUnique** berikut ini:

```
isUnique :: [Char] -> Bool
{- isUnique(lc) menghasilkan true jika lc
   adalah list dengan elemen unik, yaitu tidak
   ada elemen pada lc yang muncul lebih
   dari 1 kali -}
```

Soal 2

Tuliskan realisasi dari fungsi **posOfX** berikut ini:

```
posOfX :: Char -> [Char] -> int
{- posOfX(e,lc) menghasilkan sebuah bilangan integer yang
   menyatakan posisi e pada list of character lc.
   Jika e bukan elemen dari lc, fungsi akan menghasilkan 0.
   Prekondisi: lc memiliki elemen unik -}
```

Soal 3

Tuliskan realisasi dari fungsi **splitAlternate** berikut ini:

```
splitAlternate :: [Char] -> ([Char],[Char])
{- splitAlternate(l) menghasilkan dua buah list, misalnya l1
   dan l2.
   l1 berisi semua elemen l pada posisi ganjil,
   l2 berisi semua elemen l pada posisi genap -}
```