Responsi

Praktikum 2 - Shift 2

made with love by labpro

Soal 1 (isPalindrom)

Time limit	1 s
Memory limit	64 MB

Nama File: ListOfCharacter.hs

Header: module ListOfCharacter where

Salinlah definisi list of character dalam file ListOfCharacter.hs.

Buatlah fungsi **isPalindrom** yang menerima masukan sebuah list of character, misalnya lc, dan menghasilkan true jika lc adalah palindrom, yaitu jika dibaca dari awal maupun dari akhir sama. List kosong adalah palindrom. List 1 elemen adalah list palindrom.

Contoh aplikasi dan hasil:

Aplikasi	Hasil
isPalindrom ['s','a','y']	False
isPalindrom ['k','o','k']	True
isPalindrom ['k']	True
isPalindrom []	True

Soal 1 (isPalindrom)

```
module ListOfCharacter where

-- Inverse
-- DEFINISI DAN SPESIFIKASI
-- fungsi yang menerima masukan sebuah list of character, misalnya lc, dan menghasilkan list,
misalnya lc', yang berisi elemen-elemen lc dengan urutan yang dibalik

inverse :: [Char] -> [Char]

-- REALISASI
inverse lc =
      | isEmpty lc = []
      | konso (last lc) (inverse (init lc))
```

Soal 1 (isPalindrom)

```
-- isPalindrom
-- DEFINISI DAN SPESIFIKASI
-- fungsi yang menerima masukan sebuah list of character, misalnya lc, dan menghasilkan true jika
lc adalah palindrom. List kosong adalah palindrom. List 1 elemen adalah list palindrom.
isPalindrom :: [Char] -> Bool

-- Realisasi
isPalindrom lc
      | inverse lc == lc = True
      | otherwise = False

-- Aplikasi
-- isPalindrom ['s','a','y'] => False
-- isPalindrom ['k'] => True
-- isPalindrom [] => True
```

Soal 2 (Is Equal)

Nama File: ListOfInteger.hs

Header: module ListOfInteger where

Salinlah definisi list of integer dalam file ListOfInteger.hs.

Buatlah fungsi **isEqual** yang menerima masukan 2 buah list of integer, misalnya l1 dan l2, dan menghasilkan true jika l1 sama dengan l2, yaitu jika banyaknya elemen l1 = banyaknya elemen l2 dan jika tiap elemen pada urutan yang sama adalah sama. l1 dan l2 mungkin kosong.

Contoh aplikasi dan hasil:

Aplikasi	Hasil
isEqual [] []	True
isEqual [] [1]	False
isEqual [1] []	False
isEqual [1,2,3] [1,2,3]	True

Soal 2 (Is Equal)

```
module ListOfInteger where

-- DEFINISI DAN SPESIFIKASI
isEqual :: [Int] -> [Int] -> Bool
{-
   Mengecek apakah dua buah list sama persis atau tidak

   Basis:
   - [] [] = True
   - [x] [] = False
   - [] [y] = False

   Rekurens:
   Cek head l1 dan l2 lalu rekursif dengan fungsi yang sama pada tail(l1) dan tail(l2)
   - [a|p] [a|q] (head l1 == head l2) dan isEqual p q (tail(l1) == tail(l2))
-}
```

Soal 2 (Is Equal)

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

-- APLIKASI
-- isEqual [] []
-- isEqual [1,2,3,4] [1,2,3]
```

Soal 3 (ListOfInteger)

Time limit	1 s
Memory limit	64 MB

Nama File: ListOfInteger.hs

Header: module ListOfInteger where

Salinlah definisi list of integer dalam file ListOfInteger.hs.

Buatlah fungsi **maxNb** yang menerima masukan sebuah list of integer, misalnya l, yang tidak kosong dan menghasilkan pasangan nilai/tuple (max,nbmax) dengan max berisi nilai maksimum elemen l dan nbmax adalah banyaknya kemunculan max dalam l.

Contoh aplikasi dan hasil:

Aplikasi	Hasil
maxNb [1,2,2]	(2,2)
maxNb [3,3,-4,-5,3,3,0]	(3,4)

HINTS

Untuk membantu, dapat membuat fungsi berikut:

-- Definisi dan Spesifikasi

minList :: [Int] -> Int

{- **minList l** mengembalikan nilai minimum dari seluruh elemen list -}

nbX :: Int -> [Int] -> Int

{- **nbX x l** menghasilkan banyaknya kemunculan x pada l -}

Soal 3 (ListOfInteger)

```
module ListOfInteger where

-- DEFINISI DAN SPESIFIKASI LIST
{- type List of Int: [ ] atau [e o List] atau [List o e]
   Definisi type List of Int
   Basis: List of Int kosong adalah list of Int
   Rekurens:
   List tidak kosong dibuat dengan menambahkan sebuah elemen bertype Int di awal
   sebuah list atau
   dibuat dengan menambahkan sebuah elemen bertype Int di akhir sebuah list -}

-- DEFINISI DAN SPESIFIKASI KONSTRUKTOR
konso :: Int -> [Int] -> [Int]
{- konso e li menghasilkan sebuah list of integer dari e (sebuah integer) dan li
   (list of integer), dengan e sebagai elemen pertama: e o li -> li' -}
-- REALISASI
konso e li = [e] ++ li
```

Soal 3 (ListOfInteger)

```
konsDot :: [Int] -> Int -> [Int]
{- konsDot li e menghasilkan sebuah list of integer dari li (list of integer) dan
   e (sebuah integer), dengan e sebagai elemen terakhir: li o e -> li' -}
-- REALISASI
konsDot li e = li ++ [e]

-- DEFINISI DAN SPESIFIKASI SELEKTOR
-- head :: [Int] -> Int
-- head l menghasilkan elemen pertama list l, l tidak kosong

-- tail :: [Int] -> [Int]
-- tail l menghasilkan list tanpa elemen pertama list l, l tidak kosong

-- last :: [Int] -> Int
-- last l menghasilkan elemen terakhir list l, l tidak kosong
```

Soal 3 (ListOfInteger)

```
-- init :: [Int] -> [Int]
-- init l menghasilkan list tanpa elemen terakhir list l, l tidak kosong

-- DEFINISI DAN SPESIFIKASI PREDIKAT
isEmpty :: [Int] -> Bool
-- isEmpty l  true jika list of integer l kosong
-- REALISASI
isEmpty l = null l

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

Soal 3 (ListOfInteger)

```
-- Solution
-- DEFINISI DAN SPESIFIKASI PREDIKAT
maxList :: [Int] -> Int
-- maxList mengembalikan nilai maksimum dari list of integer
-- REALISASI
maxList [x] = x
maxList (x:xs) = max x (maxList xs)

-- DEFINISI DAN SPESIFIKASI PREDIKAT
nbX :: Int -> [Int] -> Int
-- nbX mengembalikan jumlah kemunculan suatu integer pada list of integer
nbX _ [] = 0
nbX x (y:ys)
      | x == y = 1 + nbX + ys
      | otherwise = nbX x ys
```

Soal 3 (ListOfInteger)

```
-- DEFINISI DAN SPESIFIKASI PREDIKAT
maxNb :: [Int] -> (Int, Int)
-- maxNb mengembalikan tuple nilai maksimum dan jumlah kemunculannya pada list of integer
maxNb l = (maxVal, countMaxVal)
      where
            maxVal = maxList l
            countMaxVal = nbX maxValue l
```

Soal 4 (Duel)

Nama header : module Duel where
Nama file : Duel.hs

Tuan Vin adalah seorang koboi. Saat berduel dengan lawannya, ia harus bersiap-siap mendengar kata "desperado". Apabila ia mendengar kata "desperado", maka ia harus mengatakan "BANG".
Bantulah Tuan Vin untuk mengubah kata "desperado" menjadi "BANG" dari sebuah list agar Tuan Vin menang dengan nama fungsi **duel** yang menerima list bertipe String dan mengeluarkan list bertipe String

Contoh :
li = ["one", "two", "desperado", "cowboy", "guns", "horse", "desperado", "desperado", "desperado"]
Hasil Keluaran = ["one", "two", "BANG", "cowboy", "guns", "horse", "BANG", "BANG", "BANG"

Hint: Gunakan operator : untuk melakukan konkatenasi String dengan List of String

Soal 4 (Duel)

```haskell
module Duel where

-- DUEL
-- DEFINISI DAN REALISASI
duel :: [String] -> [String]
{- fungsi yang mengubah kaya "desperado" menjadi "BANG" dari sebuah list
fungsi duel menerima list bertipe string dan mengeluarkan list bertipe list -}
duel li
    | null li = []
    -- apabila list null, mengembalikan list kosong
    | head li == "desperado" = ["BANG"] ++ duel(tail li)
    {- apabila elemen pertama list adalah "desperado" maka akan diganti menjadi "BANG"
    dan dikontatenasi dengan hasil duel sisa elemen yang lain -}
    | otherwise = [head li] ++ (duel(tail li))
    {- apabila elemen pertama list bukan merupakan "desperado" makan akan mengembalikan
    elemen head yang dikontatenasi dengan hasil duel sisa elemen yang lain -}
-- soal di atas menggunakan konkatenasi string dengan list of string serta rekursi
```

Soal 5 (Pecah Berdasarkan Posisi)

Nama File: ListOfCharacter.hs

Header: module ListOfCharacter where

Salinlah definisi list of character dalam file ListOfCharacter.hs.

Buatlah fungsi **splitAlternate** berikut ini yang menerima masukan sebuah list of character, misalnya l dan menghasilkan dua buah list of character, misalnya l1 dan l2. l1 berisi semua elemen l pada posisi ganjil, l2 berisi semua elemen l pada posisi genap.

Definisinya adalah sebagai berikut:

```
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.

  l mungkin kosong.

-}
```

Contoh aplikasi dan hasil:

Aplikasi	Hasil
splitAlternate ['x','y','z']	("xz","y")
splitAlternate ['k','l','x','z','y']	("kxy","lz")
splitAlternate ['n']	("n","")
splitAlternate ['p','q','r','s']	("pr","qs")
splitAlternate []	("","")

Tips: Perhatikan perbedaan list dengan panjang genap dan ganjil!

Soal 5 (Pecah Berdasarkan Posisi)

```
module ListOfCharacter where

-- DEFINISI DAN SPESIFIKASI
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.
  l mungkin kosong. -}

-- REALISASI
Akan dijelaskan di slide selanjutnya

-- APLIKASI
-- splitAlternate ['p','q','r','s']
```

Soal 5 (Pecah Berdasarkan Posisi) Alternatif 1 - Dari Belakang

```
-- REALISASI
splitAlternate a =
  let
    oddArr t
      | isEmpty t = t -- Mengembalikan list kosong
      | mod (length t) 2 == 0 = oddArr(init t) -- Tidak mengambil elemen apabila panjang genap
      | otherwise = konsDot (oddArr(init t)) (last t) -- Mengambil elemen apabila panjang ganjil

    evenArr t
      | isEmpty t = t -- Mengembalikan list kosong
      | mod (length t) 2 == 0 = konsDot (evenArr(init t)) (last t) -- Mengambil elemen apabila
panjang genap
      | otherwise = evenArr(init t) -- Tidak mengambil elemen apabila panjang ganjil
  in
    (oddArr a, evenArr a)
```

Soal 5 (Pecah Berdasarkan Posisi) Alternatif 2 - Dari Depan

```
-- REALISASI
splitAlternate a =
  let
    lengthParity = mod (length a) 2

    oddArr t
      | isEmpty t = t
      | mod (length t) 2 == lengthParity = konso (head t) (oddArr(tail t)) -- Mengambil elemen apabila
paritas sama
      | otherwise = oddArr(tail t) -- Tidak mengambil elemen apabila paritas beda

    evenArr t
      | isEmpty t = t
      | mod (length t) 2 == lengthParity = evenArr(tail t) -- Tidak mengambil elemen apabila paritas
sama
      | otherwise = konso (head t) (evenArr(tail t)) -- Mengambil elemen apabila paritas beda
  in
    (oddArr a, evenArr a)
```

Soal 6 (pecahListPosNeg)

Nama File: ListOfInteger.hs

Header: *module ListOfInteger where*

Salinlah definisi list of integer dalam file ListOfInteger.hs.

Buatlah sebuah fungsi **pecahListPosNeg** (definisi, spesifikasi, dan realisasi) yang menerima masukan sebuah list of integer (l) dan mengembalikan dua buah list of integer (l1, l2). l1 memuat semua elemen l yang bernilai positif atau 0 sedangkan l2 memuat semua elemen l yang bernilai negatif. Urutan kemunculan elemen pada l1 dan l2 tetap sama dengan urutan elemen pada l.

No	Input	Output
1.	pecahListPosNeg [1,3,0,4,-1,4,-9]	([1,3,0,4,4],[-1,-9])
2.	pecahListPosNeg [2,3,4,5]	([2,3,4,5],[])
3.	pecahListPosNeg []	([],[])

Soal 6 (pecahListPosNeg)

```
module ListOfInteger where

-- DEFINISI DAN SPESIFIKASI
pecahListPosNeg :: [Int] -> ([Int],[Int])
{- pecahListPosNeg(l) menerima masukan sebuah list of integer (l) dan mengembalikan dua
buah list of integer (l1,l2). l1 memuat semua elemen l yang bernilai positif atau 0
sedangkan l2 memuat semua elemen l yang bernilai negatif. Urutan kemunculan elemen
pada l1 dan l2 tetap sama dengan urutan elemen pada l. -}

-- REALISASI
Akan dijelaskan di slide selanjutnya

-- APLIKASI
-- pecahListPosNeg [1,3,0,4,-1,4,-9]
```

Soal 6 (pecahListPosNeg)

```
-- REALISASI
pecahListPosNeg l =
  let
      listPosZero l1 -- Rekursi untuk mencari semua elemen l yang bernilai positif atau 0
            | isEmpty l1 = [] -- Jika kosong, kembalikan list kosong (basis)
            | head l1 >= 0 = konso (head l1) (listPosZero (tail l1))
                -- Jika elemen pertama positif atau 0, tambahkan ke
                -- list l1 dengan elemen tsb menjadi elemen pertama agar urutan tetap sama
                -- lakukan rekursi untuk elemen sisanya
            | otherwise = listPosZero (tail l1)
                -- Jika elemen pertama negatif, maka skip elemen tersebut
                -- dan lakukan rekursi untuk elemen sisanya
```

Soal 6 (pecahListPosNeg)

```
  listNeg l2 -- Rekursi untuk mencari semua elemen l yang bernilai negatif
        | isEmpty l2 = [] -- Jika kosong, kembalikan list kosong (basis)
        | head l2 < 0 = konso (head l2) (listNeg (tail l2))
            -- Jika elemen pertama negatif, tambahkan ke
            -- list l2 dengan elemen tsb menjadi elemen pertama agar urutan tetap sama
            -- lakukan rekursi untuk elemen sisanya
        | otherwise = listNeg (tail l2)
            -- Jika elemen pertama positif atau 0, maka skip elemen tsb
            -- dan lakukan rekursi untuk elemen sisanya
in
    (listPosZero l, listNeg l)
```

Soal 7 (Alternate Sort)

Soal ini soal bonus. Kerjakan hanya bila soal-soal sebelumnya sudah selesai dikerjakan.

Nama File: AlternateSort.hs

Header: module AlternateSort where

Diberikan sebuah list, Pak Engi memiliki sebuah algoritma prosedural sebagai berikut.

1. Urutkan list tersebut
2. Bagi list menjadi 2 sama besar, misal l1 dan l2. Jika panjang list ganjil, maka l1 akan memiliki 1 elemen lebih banyak dibanding l2
3. Ambil elemen terkecil dari l1, masukkan ke akhir l3.
4. Ambil elemen terbesar dari l2, masukkan ke akhir l3.
5. Ulangi langkah 3 dan 4 sampai kedua list kosong.

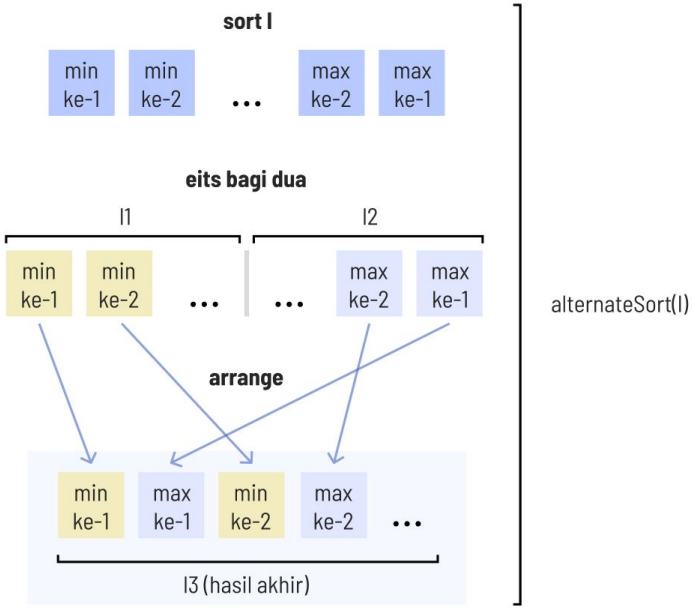
Contohnya, jika list awal adalah [9,10,11,12], maka l3 akan menjadi [9,12,10,11]

Pak Engi telah selesai membuat algoritma prosedural tersebut. Anda, sebagai pemrogram handal, ingin membuat versi fungsional dari kode tersebut. Namun, anda menyadari bahwa langkah prosedural tersebut terlalu kompleks untuk diimplementasikan dalam waktu 2 jam, sehingga anda ingin mencari cara lain untuk mengimplementasikan algoritma tersebut. Buatlah program yang dapat melakukan algoritma tersebut!

Contoh aplikasi fungsi dan hasilnya:

```
> alternateSort[9,10,11,12]
[9,12,10,11]
> alternateSort[5,2,5,2,1]
[1,5,2,5,2]
```

Apa sih yang sebenarnya dilakukan?



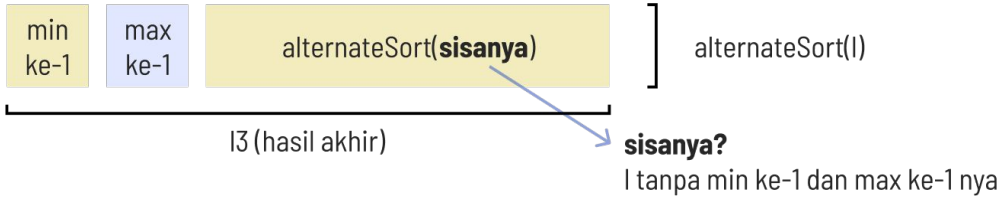
alternateSort(l)

inget hasil akhir, **jangan terpaku sama prosedur!**

Perhatikan bahwa hasil akhir yang diinginkan adalah list yang selang-seling antara nilai maksimum dan minimumnya.

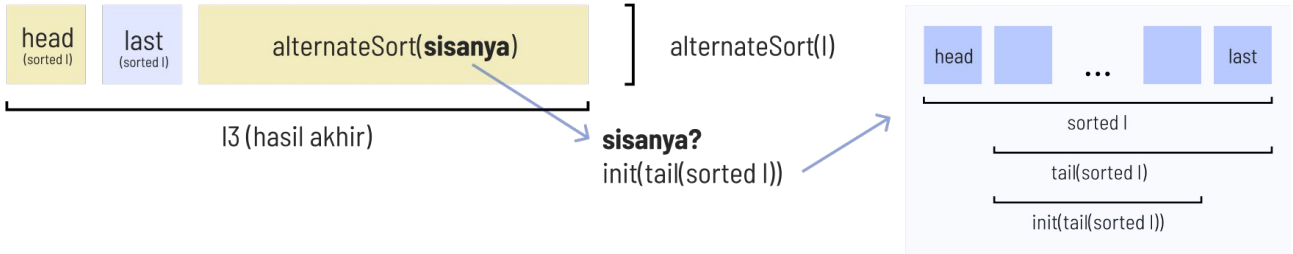
Bagaimana ***cara meng-achieve hasil yang sama*** dengan pendekatan fungsional?

Pendekatan Fungsional (Rekursif)



Gimana mengimplementasikan ini di list rekursif yang cuman ada primitif: head(l), tail(l), init(l), last(l), dll?

Jika kita dapat men-sort l secara ascending, kita bisa dapat:



Kode

```
module AlternateSort where
-- definisikan isEmpty, isOneElmt, dan konso

-- DEFINISI DAN SPESIFIKASI FUNGSI ANTARA
minEl :: [Int] -> Int
{- minEl l menghasilkan nilai terkecil dari l
-}
-- REALISASI
minEl l
  | isOneElmt l = head l
  | head l < minEl (tail l) = head l
  | otherwise = minEl (tail l)

del :: Int -> [Int] -> [Int]
{- del x l menghasilkan elemen l yang bernilai
x yang muncul pertama kali -}
-- REALISASI
del x l
  | isEmpty l = l
  | head l == x = tail l
  | otherwise = konso (head l) (del x (tail l))
```

```
sort :: [Int] -> [Int]
{- sort l menghasilkan list l yang terurut membesar -}
-- REALISASI
sort l
  | isEmpty l = l
  | otherwise = konso (minEl l) (sort (del (minEl l) l))

-- DEFINISI DAN SPESIFIKASI FUNGSI UTAMA
alternateSort :: [Int] -> [Int]
{- alternateSort l menghasilkan list yang sama dengan list
l yang sudah diurutkan, lalu dibagi dua menjadi l1 dan l2,
lalu diambil elemen terkecil dari l1 lalu elemen terbesar
dari l2 hingga kedua list kosong -}
-- REALISASI
alternateSort l
  | isEmpty l || isOneElmt l = l
  | otherwise = konso (head (sort l)) (konso (last (sort
l)) (alternateSort (init(tail(sort l)))))

-- APLIKASI
-- alternateSort [9,10,11,12]
```

Mudah kan? :D
Ada pertanyaan?