Responsi - 2

Oleh: Tim Asisten IF1210 Dasar Pemrograman 2021-2

Agenda Responsi

1. Penjelasan dari Asisten
2. Mengerjakan soal praktikum yang sama pada olympia yang berdurasi 2 jam.
3. **Kerjakan saja soal yang kalian dapat minggu lalu atau bebas pilih 1 soal 1 opsi, yang lainnya kosongkan/menjadi opsional untuk dikerjakan.**

Disclaimer

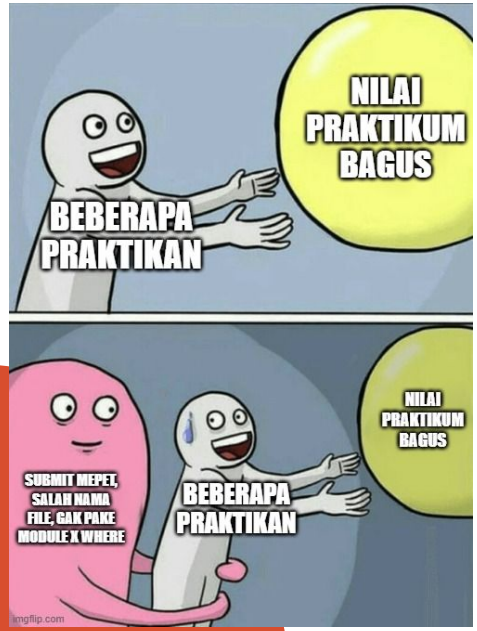
Wahai anak-anak tukang *copy-paste* 😑 (*lewati slide ini jika anda tidak merasa*), **Gunakan SLIDE ini buat referensi ide untuk bahan belajar dan menjawab soal.** Penyalahgunaan **SLIDE Responsi** merupakan tanggung jawab masing-masing. Setidaknya, mengetik ulang **dan mencoba untuk memahami.**

~~Ketahuan *copas* lagi slide gakan dikasih akses slide responsi~~ 😠 #jk

Kesalahan **FATAL** Paling Umum

1. Wahai praktikan - praktikan yang masih salah **penamaan nama file** bahkan masih tidak tahu cara pake **module X where**, bertobatlah kalian 😠!!! *Asisten greget liat kalian gak dapet nilai gara-gara salah upload nama file bahkan jawaban bener tapi masih gak ngikutin standar cara jawab di olympia.* ***YANG NILAI PRAKTIKUM KALIAN ITU GRADER = KOMPUTER***, Walaupun benar akan tetap 0 apabila tidak ternilai olympia.
2. Tidak menekan submit lalu finish attempt meskipun sudah selesai lebih awal.

Kesalahan **FATAL** Paling Umum



3. Attempt tapi gak dijawab, dicoba ngoding aja engga. Sebelum menyerah, coba dulu, yuk bisa yuk 😄.
4. Gak ikut praktikum, bahkan gak buka olympia sama sekali dari praktikum 0. Tolong bantu ingetin temennya yak 🙂.
5. Submit sangat mepet. Diingatkan lagi, **olympia kalau rame-rame submit bakal lag/lemot dan kalian bakal dihitung telat, untuk menghindari hal tersebut submit 10 menit sebelum deadline (lebih baik).** *Intinya cari aman ya…*

~~Saran-Saran~~ Paksaan Untuk Menghindari Kesalahan

1. ~~Usahakan~~ Paksakan kurang 5 menit sudah mulai check dan submit jawaban kalian.
2. Kalau misalnya mau nyoba soal tapi waktu mepet, finish attempt pertama kalian dan reattempt praktikum. Nilai akan diambil tertinggi kok, gunakan jumlah attempt praktikum kalian dengan baik 😄.
3. Kalau kalian mau mengadu tentang masalah pengumpulan, kasih *screenshot* (**no ss = hoax**) bukti submit kalian dengan waktu submit yang terlihat pada satu *screenshot* yang sama, kalau kalian terbukti submitnya < 1 menit dari deadline ga akan dibantu ya. *Kalau bisa sih <3 menit supaya ga cuman doi yang dikasih* 💖

~~Saran-Saran~~ Paksaan Untuk Menghindari Kesalahan

3. Jangan ngerjain bonus kalau waktu mepet. Mending bonus ga selesai daripada seluruh praktikum gak dapet nilai.
4. Baca dan/atau ketahui instruksi dengan benar, pada *responsi lalu* ada yang mengerjakan ke 21 soal dan tidak berhasil mengumpulkan karena waktu tidak cukup.

Perubahan pada Responsi Ini

1. Akan dikasih grace period 10 menit setelah waktu praktikum selesai, di waktu ini kalian tidak bisa **mengerjakan** soal lagi dan hanya untuk **submit**.
2. Fitur ini masih eksperimental karena gak pernah dicoba, jadi usahakan kumpulkan responsi sebelum waktu selesai.

Soal 1

Banyak Elemen List (nbElmt)

Nama File: ListOfInteger.hs

Module: ListOfInteger

Salinlah definisi list of integer dalam file ListOfInteger.hs.

Buatlah fungsi **nbElmt** yang menerima masukan sebuah list of integer (mungkin kosong) dan menghasilkan banyaknya elemen dalam list (0 jika list kosong) **secara rekursif**. ***Dilarang menggunakan fungsi standar di Haskell (kecuali untuk selektor).***

Berikut adalah definisi dan spesifikasinya:

Contoh aplikasi dan hasil:

Aplikasi	Hasil
nbElmt []	0
nbElmt [1]	1
nbElmt [1,2,3]	3

Banyak Elemen List (nbElmt)

```haskell
module ListOfInteger where

-- DEFINISI DAN SPESIFIKASI
nbElmt :: [Int] -> Int

-- REALISASI
nbElmt l
      | (null l) = 0    -- basis
      | otherwise = nbElmt (tail l) + 1    -- rekurens
```

Atau

```haskell
module ListOfInteger where

-- DEFINISI DAN SPESIFIKASI
nbElmt :: [Int] -> Int
isEmpty :: [Int] -> Bool

-- REALISASI
isEmpty l = null l
nbElmt l =    if (isEmpty l) then 0  -- basis
                    else 1 + (nbElmt (tail l)) -- rekurens
```

Banyak Kemunculan x pada List (nbOcc)

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 **nbOcc** yang menerima masukan sebuah list of integer, misalnya l, dan sebuah integer, misalnya x, dan menghasilkan berapa banyak kemunculan x pada list of integer l. l mungkin kosong.

Contoh aplikasi dan hasil:

Aplikasi	Hasil
nbOcc [] 10	0
nbOcc [10] 10	1
nbOcc [10,20,20] 20	2
nbOcc [10,20,20] 3	0

Banyak Kemunculan x pada List (nbOcc)

```
module ListOfInteger where

-- DEFINISI DAN SPESIFIKASI

nbOcc :: [Int] -> Int -> Int

--REALISASI

nbOcc l x

 | null l = 0

 | otherwise =

    if head l == x then 1 + nbOcc (tail l) x

    else nbOcc (tail l) x
```

Elemen Ke-N List

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 **elmtKeN** yang menerima masukan sebuah list of integer, misalnya l, dan sebuah integer, misalnya n, dan menghasilkan elemen ke-n dari list of integer l. Diasumsikan 0 < n <= banyaknya elemen l dan list l tidak kosong (minimum terdiri atas 1 elemen).

Contoh aplikasi dan hasil:

Aplikasi	Hasil
elmtKeN [10] 1	10
elmtKeN [10,20,30] 3	30

Elemen Ke-N List

```haskell
-- DEFINISI DAN SPESIFIKASI
module ListOfInteger where


-- REALISASI
elmtKeN :: [Int]->Int->Int

elmtKeN l n = if n == 1 then head l          --basis
              else elmtKeN (tail l) (n-1)    --rekurens
```

Soal 2

Buang Tujuh

Nama module : BuangTujuh
Nama file : BuangTujuh.hs

Salinlah definisi list of integer dalam file ListOfInteger.hs.

Tuan Vin membenci angka 7. Setiap kali dia melihat angka 7 atau kelipatan dari 7, ia selalu ingin membuangnya. Bantulah Tuan Vin untuk membuang setiap angka yang berhubungan dengan 7 dari sebuah list of integer dengan membuat fungsi **buangTujuh**. Angka di dalam list hanya bernilai satuan atau puluhan.

Contoh aplikasi:

```
> buangTujuh [1, 2, 7, 17, 13, 14, 21, 20, 71, 73]
[1, 2, 13, 20]
> buangTujuh []
[]
```

Catatan: Angka yang digunakan hanya satuan dan puluhan.

Buang Tujuh

Jawaban

```
module BuangTujuh where

-- DEFINISI DAN SPESIFIKASI
buangTujuh :: [Int] -> [Int]
{- buangTujuh(l) menghasilkan sebuah list baru yang merupakan list li
tetapi tanpa elemen yang merupakan kelipatan 7 atau ada 7 di elemennya.
-}

-- REALISASI
buangTujuh l =
     if isEmpty l then l -- BASIS
       else -- REKURENS
           if mod (head l) 7 == 0 || mod (head l) 10 == 7 || div (head
l)   10 == 7 then
               buangTujuh (tail l)
           else konso (head l) (buangTujuh (tail l))

-- APLIKASI
-- buangTujuh [1, 2, 7, 17, 13, 14, 21, 20, 71, 73]
-- [1, 2, 13, 20]
```

Happy Five

Nama module : HappyFive
Nama file : HappyFive.hs

Tuan Vin menyukai angka 5. Hanya angka-angka yang berhubungan dengan 5 yang dapat membuat Tuan Vin senang.

Bantulah Tuan Vin untuk membuang setiap angka yang tidak berhubungan dengan 5 atau kelipatan dari 5 dari sebuah list agar Tuan Vin senang dengan nama fungsi **happyFive**. Angka didalam list hanya bernilai satuan atau puluhan.

Contoh :
li = [1, 2, 7, 5, 15, 17, 13, 14, 21, 20, 51, 52]
Hasil Keluaran = [5, 15, 20, 51, 52]

Catatan :
Angka yang digunakan hanya satuan dan puluhan.

Happy Five

```
module HappyFive where

-- DEFINISI DAN SPESIFIKASI
happyFive :: [Int] -> [Int]
  {- happyFive l akan menerima masukkan sebuah list dan membuang elemen integer di dalamnya
yang tidak berhubungan dengan
  angka 5 atau kelipatan 5 -}
  -- Angka di dalam list 0 <= el <= 100

konso :: Int -> [Int] -> [Int]
isEmpty :: [Int] -> Bool

-- REALISASI
konso e l = [e] ++ l
isEmpty l = null l
happyFive l =
   if (isEmpty l) then []
   else
      if ((head l) >= 50 && (head l) < 60) || ((head l) `mod` 5 == 0) then (konso (head l) (happyFive (tail l)))
      else
         happyFive (tail l)
```

Soal 3

Inverse List

Nama File: ListOfCharacter.hs

Header: module ListOfCharacter where

Salinlah definisi list of character dalam file ListOfCharacter.hs.

Buatlah fungsi **inverse** yang menerima masukan sebuah list of character, misalnya lc, dan menghasilkan list, misalnya lc', yang berisi elemen-elemen lc dengan urutan yang dibalik. Jika lc = [e1,e2,...en-1,en] maka lc' = [en,en-1,...,e2,e1].

Contoh aplikasi dan hasil:

Aplikasi	**Hasil**
inverse ['s','a','y']	"yas"
inverse []	""
inverse ['k']	"k"
inverse ['s','a','y','n','i','c','e']	"ecinyas"

Inverse List

Opsi Jawaban

```haskell
module ListOfCharacter where

    -- DEFINISI DAN SPESIFIKASI
    inverse :: [Char] -> [Char]
    konso :: Char -> [Char] -> [Char]

    -- REALISASI
    konso e l = [e] ++ l
    inverse lc
        | (null lc) = []    -- basis
        | (length lc == 1) = lc
        | otherwise = konso (last lc) (inverse (init lc))    -- rekurens
```

Make Unique

Nama File: ListOfCharacter.hs

Header: module ListOfCharacter where

Salinlah definisi list of character dalam file ListOfCharacter.hs.

Buatlah fungsi **makeUnique** yang menerima masukan sebuah list of character, misalnya lc, dan menghasilkan list dengan elemen-elemen unik, yaitu, kemunculan setiap elemen (jika tidak kosong), hanya 1.

Contoh aplikasi dan hasil:

Aplikasi	Hasil
makeUnique ['s','a','y']	"say"
makeUnique []	""
makeUnique ['k','k']	"k"
makeUnique ['s','a','y','y','a','s']	"say"

Make Unique

```
-- DEFINISI DAN SPESIFIKASI PREDIKAT

isEmpty :: [Char] -> Bool

isEmpty l = null l


isElmt :: Char -> [Char] -> Bool

isElmt a l

  | isEmpty l = False

  | otherwise =

    if a==(head l) then True

    else isElmt a (tail l)
```

```
-- FUNGSI UTAMA

makeUnique :: [Char] -> [Char]

makeUnique l

  | isEmpty l = []

  | isElmt (last l) (init l) = makeUnique (init l)
  | otherwise = (makeUnique (init l)) ++ [last l]
```

Soal 4

Soal

Konkatenasi 2 List

Salinlah definisi list of character dalam file ListOfCharacter.hs.

Buatlah fungsi konkat yang menerima masukan 2 buah list of character, misalnya lc1 dan lc2, yang masing-masing mungkin kosong, dan menghasilkan list baru yang merupakan penggabungan lc1 dengan lc2 (lc1 di awal).

Contoh aplikasi dan hasil:

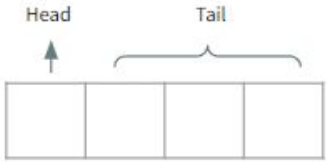
Aplikasi	**Hasil**
konkat ['s','a','y'] ['a']	"saya"
konkat [] ['a']	"a"
konkat ['a'] []	"a"
konkat [] []	""

Konkatenasi 2 List

Jawaban 1

```
module ListOfCharacter where

-- Definisi dan Spesifikasi

    konkat :: [Char] -> [Char] -> [Char]

    -- konkat(lc1, lc2) menggabungkan list lc2 di belakang list
lc1

-- Realisasi

    konkat lc1 lc2

        | isEmpty lc2 = lc1

        | otherwise = konkat (konsDot lc1 (head lc2)) (tail lc2)
```

Jawaban 2 dan 3 tetap include Definisi dan Spesifikasi yaa!




```
konsDot lc e = lc ++ [e]
```

Jawaban 2

```
-- Realisasi

konkat lc1 lc2

        | null lc2 = lc1

        | otherwise = [head lc1] ++ konkat (tail lc1) lc2
```

Jawaban 3

```
 -- Realisasi

 konkat lc1 lc2 = lc1 ++ lc2 --
 Menggabungkan list 1 dan list 2
```

Set Difference

Salinlah definisi list of integer dalam file ListOfInteger.hs.

Buatlah sebuah fungsi setDiff (definisi, spesifikasi, dan realisasi) yang menerima masukan dua buah list of integer (l1 dan l2) dengan elemen unik dan terurut membesar dan mengembalikan sebuah list of integer yang elemennya adalah semua elemen l1 yang tidak ada di l2.

Contoh aplikasi dan hasil:

Aplikasi	**Hasil**
setDiff [2,4,6,8,10] [3,4,5,6]	[2,8,10]
setDiff [] [2,3,4,5]	[]
SetDiff [4,6,8,13,26] [ ]	[4,6,8,13,26]

Set Difference

```haskell
module ListOfInteger where

-- DEFINISI DAN SPESIFIKASI

isMember:: Int -> [Int] -> Bool

{- Memeriksa apakah sebuah angka berada dalam list -}

setDiff :: [Int] -> [Int] -> [Int]

{- Menerima input 2 buah list of integer lalu mengeluarkan sebuah list of integer
yang merupakan elemen-elemen pada list pertama yang tidak ada pada list kedua. -}

-- REALISASI

isMember n l
   | isEmpty l = False
   | n == (head l) = True
   | otherwise = isMember n (tail l)

setDiff l1 l2
   | isEmpty l1 = []
   | not (isMember (head l1) l2) = konso (head l1) (setDiff (tail l1) l2)
   | otherwise = setDiff (tail l1) l2
```

ListPlus

Nama File: ListOfInteger.hs

Header: module ListOfInteger where

Salinlah definisi list of integer dalam file ListOfInteger.hs.

Buatlah fungsi **listPlus** yang menerima masukan dua buah list of integer dengan dimensi (banyaknya elemen) sama, misalnya li1 dan li2, dan menghasilkan penjumlahan kedua list, yaitu sebuah list dengan setiap elemen li1 dan li2 pada urutan yang sama dijumlahkan.

Contoh aplikasi dan hasil:

Aplikasi	Hasil
listPlus [1,2,3] [1,2,3]	[2,4,6]
listPlus [2,22,4,5] [4,0,-1,-1]	[6,22,3,4]
listPlus [] []	[]

ListPlus

```
module ListOfInteger where
-- isi file ListOfInteger.hs

listPlus :: [Int] -> [Int] -> [Int]
-- listPlus menerima 2 buah list of Integer l1, l2 dan
menghasilkan sebuah list of Integer hasil penjumlahan
keduanya

-- REALISASI
listPlus l1 l2 =
  if (isEmpty l1) then []
  else konso ((head l1) + (head l2)) (listPlus (tail l1)
(tail l2))
```

Soal 5

Fungsi Sebagai Parameter Fungsi

List Index

Tuliskan definisi, spesifikasi, dan realisasi dari fungsi listIndex yang menerima masukan:

1. Sebuah list of integer, misalnya l
2. Sebuah fungsi yang menerima masukan sebuah integer dan menghasilkan sebuah char, misal f

Fungsi listIndex akan menghasilkan sebuah list of character yang melambangkan nilai-nilai indeks dari suatu list nilai integer. Misal fungsi f akan mengembalikan nilai B untuk range nilai [70,80], maka nilai 75 akan secara otomatis diubah menjadi 'B' oleh fungsi f.

-- [80-100]: 'A'

-- [70-80): 'B'

-- [65-70): 'C'

-- [55-65): 'D'

-- [0-55): 'E'

Masukan		**Hasil listIndex l f**
l	**f**	
[75, 90, 10, 20, 100]	nilai :: Int -> Char	["BAEEA"]

List Index

```
module ListIndex where

    konso :: Char -> [Char] -> [Char]

{- konso e lc menghasilkan sebuah list of character dari e
(sebuah character)

   dan lc (list of char), dengan e sebagai elemen pertama: e
o lc -> lc' -}

-- REALISASI

    konso e lc = [e] ++ lc


    isEmpty :: [Int] -> Bool

    isEmpty x = null x
```

```
listIndex :: [Int] -> (Int -> Char) -> [Char]

listIndex i f

    | isEmpty i = []

    | otherwise = konso (f (head i)) (listIndex (tail i) f)
```

Soal 6

Offset List

Tuliskan definisi, spesifikasi, dan realisasi fungsi offsetList yang menerima masukan sebuah list of integer yang melakukan “offset” atau perubahan nilai terhadap elemen list sesuai dengan aturan tertentu (yang ditentukan oleh sebuah fungsi offset) dan menghasilkan list baru dengan elemen hasil offset.

Contoh:

1. Dengan fungsi offset plus2, akan menghasilkan list baru dengan nilai setiap elemen yang sudah bertambah 2
2. Dengan fungsi offset minus1, akan menghasilkan list baru dengan nilai setiap elemen yang sudah berkurang 1
3. Dengan fungsi offset offKond, akan menghasilkan list baru dengan nilai setiap elemen yang diubah sesuai ketentuan range tertentu

Offset List

Masukan		Hasil offsetList l offset
l	offset	
[2,9,14,1,5,6]	plus2 :: Int -> Int -- plus2 a menghasilkan a + 2	[4,11,16,3,7,8]
[2,3,4,1,5,6,10]	minus1 :: Int -> Int -- minus1 a menghasilkan a - 1	[1, 2, 3, 0, 4, 5, 9]
[-2,0,4,31,15,60]	offKond :: Int -> Int -- offKond a: -- jika a >= 0 dan a <= 40 maka menghasilkan 10 -- jika a < 0 maka menghasilkan 0 -- jika a > 40 maka menghasilkan 20	[0,10,10,10,10,20]
[]	plus2 :: Int -> Int -- plus2 a menghasilkan a + 2	[]

Offset List

```haskell
module OffsetList where
    -- DEFINISI DAN SPESIFIKASI
    isEmpty :: [Int] -> Bool
    konso :: Int -> [Int] -> [Int]
    offsetList :: [Int] -> (Int -> Int) -> [Int]

    -- REALISASI
    isEmpty l = null l
    konso e li = [e] ++ li
    offsetList l offset =
        if isEmpty l then []
        else konso (offset (head l)) (offsetList (tail l) offset)
```

Soal Bonus

Alternate Sort

Time limit	1 s
Memory limit	64 MB

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]
```

Alternate Sort

```
-- DEFINISI DAN SPESIFIKASI FUNGSI ANTARA
minElem :: [Int] -> Int
-- minElem l menghasilkan elemen terkecil dari list of integer
-- REALISASI
minElem l
    | isOneElmt l = head l
    | otherwise = if (head l)<(minElem (tail l)) then (head l) else
     (minElem (tail l))

del :: Int -> [Int] -> [Int]
-- del x l menghapus elemen x dari list l
-- REALISASI
del x l
    | isEmpty l = []
    | x == (head l) = tail l
    | otherwise = konso (head l) (del x (tail l))
```

```
    sort :: [Int] -> [Int]
-- sort l menghasilkan sebuah list of integer yang elemennya terurut dari terkecil
-- ke terbesar
-- REALISASI
sort l
    | isEmpty l = l
    | otherwise = konso (minElem l) (sort (del (minElem l) l))

-- DEFINISI DAN SPESIFIKASI PROGRAM
alternateSort :: [Int] -> [Int]
-- menerima list of integer lalu memprosesnya dengan langkah berikut
-- urutkan list, lalu bagi menjadi dua (l1 & l2)
-- ambil elemen terkecil dari l1 lalu ambil elemen terbesar dari l2, ulangi sampai
-- list kosong
-- REALISASI
alternateSort l
    | isEmpty l || isOneElmt l = l
    | otherwise = konso (head (sort l)) (konso (last(sort l))
     (alternateSort(init(tail(sort l)))))
```

Alternate Sort

Kesalahan Umum

1. Tidak dikerjain hehe
2. Hanya diurutkan dari yang terkecil ke yang terbesar

Bonus ga wajib, gapapa, have fun :D