Responsi

Praktikum 1 - S2

made with love by labpro

Common Mistakes

Tidak di compile di lokal

Kebanyakan praktikan langsung melakukan submit ke olympia tanpa mengetahui programnya dapat dijalankan secara benar atau tidak

Kesalahan sintaks

Dikarenakan sintaks haskell yang baru, banyak praktikan yang tidak mengetahui sintaks yang digunakan oleh haskell

Tidak menuliskan definisi

Walaupun hanya merupakan best practice, mendefinisikan fungsi yang dibuat dapat mempermudah penggambaran soal

Tidak sempat di grade

Praktikan melakukan grading soal secara bersamaan, tidak bertahap. Sehingga terdapat waktu menunggu grading

Common Mistakes

Urutan If-else

Kebanyakan praktikan yang tidak mendapatkan nilai penuh disebabkan oleh urutan If-else yang tidak tepat

Perhatikan Soal

Kebanyakan praktikan tidak membaca soal dengan baik, seperti penamaan modul, permintaan soal, rumus yang digunakan dalam soal, dsb.

Tidak teliti

Kebanyakan praktikan tidak mendapatkan nilai penuh karena tidak teliti dalam penggunaan variabelnya. Ada yang membuat *infinite recursion* dan variabel tertukar dalam kodenya.

Kelengkapan If-else

Beberapa praktikan tidak memiliki case kondisional yang lengkap, sehingga tidak mencakup semua syarat yang ada pada soal

Soal 1 (Luas Segitiga)

Nama File: LuasSegitiga.hs

Header: module LuasSegitiga where

Buatlah definisi, spesifikasi, dan realisasi fungsi **luasSegitiga** yang menerima masukan 2 buah bilangan real (float) a dan t dengan a = alas segitiga dan t = tinggi segitiga (asumsikan: a > 0, t > 0) dan menghasilkan luas segitiga berdasarkan rumus: luas = ½ * a * t

Contoh aplikasi fungsi dan hasilnya:

```
> luasSegitiga 3 4
6.0
```

Soal 1 (Luas Segitiga)

```
module LuasSegitiga where

-- DEFINISI DAN SPESIFIKASI
luasSegitiga :: Float -> Float -> Float
{- luasSegitiga(a, t) adalah fungsi yang menerima masukan 2 buah bilangan
real (float) a dan t dengan a = alas segitiga dan t = tinggi segitiga
(asumsikan: a > 0, t > 0) dan menghasilkan luas segitiga berdasarkan
rumus: luas = ½ * a * t -}
-- a, t merupakan float (bilangan real)

-- REALISASI
luasSegitiga a t = 0.5 * a * t

-- APLIKASI
-- luasSegitiga 2 3
```

Soal 2 (Max3)

Nama File: Max3.hs

Header: module Max3 where

Tuliskanlah fungsi **max3** yang menerima 3 buah bilangan integer yang berbeda, dan menuliskan nilai terbesar di antara ketiganya. Tidak boleh membuat fungsi antara. Berikut adalah definisi dan spesifikasi fungsi dalam notasi fungsional:

```
max3 : 3 integer -> integer
{ max3 (a,b,c) mengirimkan nilai yang paling besar di antara a, b, dan c. Asumsi: a, b, c bilangan berbeda }
```

Contoh aplikasi dan hasil:

No	Aplikasi	Hasil	Keterangan
1.	max3 1 3 5	5	Bilangan terbesar adalah 5
2.	max3 (-10) 0 (-5)	0	Bilangan terbesar adalah 0
3.	max3 20 34 33	34	Bilangan terbesar adalah 34

Soal 2 (Max3)

```
module Max3 where

-- DEFINISI DAN SPESIFIKASI
max3:: Int -> Int -> Int -> Int
{- max3(a,b,c) mengirimkan nilai yang paling besar di antara a, b, dan c.
Asumsi: a, b, c bilangan berbeda -}
-- a, b, c, dan d merupakan integer

-- REALISASI
max3 a b c
      | c > b && c > a = c
      | b > a = b
      | otherwise = a

-- APLIKASI
-- max3 4 8 1
```

Soal 3 (Lama Tidur)

Nama File : LamaTidur.hs

Nama Modul : LamaTidur

Hari ini para panitia Arkavidia bekerja keras siang malam mempersiapkan acara besok hari. Namun, panitia jugalah seorang manusia yang pastinya butuh tidur yang cukup. Besok acara dimulai pada pukul 05.00 pagi, dan saat mereka selesai mempersiapkan acara besok. Para panitia merasa lelah dan langsung tidur di tempat. Para panitia ingin mengetahui apakah mereka dapat tidur dengan cukup (>= 6jam) dan menghitung lama mereka bisa tidur.

Bantulah panitia dengan membuatlah sebuah fungsi dengan nama **lamaTidur** yang dapat menerima input 3 integer yang merupakan jam (0..23), menit(0..59), detik(0..59) waktu mereka selesai mempersiapkan acara dan dapat mengeluarkan output tuple berisi lama waktu bisa tidur dalam format jam, menit, detik (selisih waktu input dari pukul 05.00 pagi).

Asumsi: input selalu valid.

Contoh:
> lamaTidur 1 0 0
(False, 4,0,0)

Soal 3 (Lama Tidur)

```
module LamaTidur where

-- DEFINISI DAN SPESIFIKASI
lamaTidur :: Int -> Int -> Int -> (Bool, Int, Int, Int)
{- Menentukan apakah tidur dengan cukup (>= 6 jam), dan menghitung lama tidur sampai pukul 05.00 -}

-- APLIKASI
-- lamaTidur 23 0 1
```

Soal 3 (Lama Tidur)

```haskell
-- REALISASI
lamaTidur j m d =
    let
        waktu_1 = j * 3600 + m * 60 + d
        waktu_2 = 5 * 3600 -- pukul 05.00

        waktu_tidur
            | j < 5 = waktu_2 - waktu_1                 -- hari yang sama
            | otherwise = waktu_2 - waktu_1 + 24 * 3600 -- hari kemarin

        -- convert hasil ke dalam satuan waktu
        jam = div waktu_tidur 3600
        menit = div (mod waktu_tidur 3600) 60
        detik = mod waktu_tidur 60

        -- tentukan apakah tidur cukup atau tidak
        tidur_cukup = jam >= 6
    in
        (tidur_cukup, jam, menit, detik)
```

Soal 4 (UkuranBaju)

Nama File: UkuranBaju.hs

Header: module UkuranBaju where

Panitia Gemastik ITB ingin mengotomatisasi penentuan ukuran baju panitia dengan memanfaatkan tinggi dalam cm (integer positif) dan berat badan dalam kg (integer positif) dengan ketentuan sebagai berikut:

- Baju ukuran M diberikan kepada mereka yang tingginya <= 150 cm (berapa pun berat badannya).
- Baju ukuran XL diberikan ke mereka yang tingginya > 170 cm dan berat badannya > 60 kg tapi masih <= 80 kg.
- Jika seseorang tingginya > 150 cm, tapi masih <= 170 cm dan berat badannya <= 80 kg, maka orang ini mendapatkan baju ukuran L.
- Jika seseorang tingginya > 150 cm, tapi masih <= 170 cm dan berat badannya > 80 kg, dia mendapatkan baju ukuran XL.
- Orang yang tingginya > 170 cm dan berat badannya <= 60 kg, mendapat baju ukuran L.
- Karena keterbatasan pembuat kaos, tidak ada kaos lain selain M, L, dan XL sehingga untuk yang tidak memenuhi kategori di atas tidak akan mendapatkan kaos. Dalam hal ini, untuk yang bersangkutan diberikan kategori khusus yaitu 4.

Buatlah fungsi **ukuranBaju** yang menerima masukan 2 buah integer positif, misalnya **t** (tinggi badan dalam cm) dan **b** (berat badan dalam kg) dan menghasilkan kode ukuran baju (1 adalah M, 2 adalah L, 3 adalah XL) atau kode 4 adalah untuk yang tidak mendapatkan kaos.

Contoh aplikasi dan hasilnya:

No	Aplikasi	Hasil	Keterangan
1.	ukuranBaju 160 75	2	Ukuran L
2.	ukuranBaju 145 45	1	Ukuran M
3.	ukuranBaju 190 85	4	Tidak mendapatkan kaos

Soal 4 (UkuranBaju)

```
module UkuranBaju where

-- DEFINISI DAN SPESIFIKASI
ukuranBaju :: Int -> Int -> Int
{- ukuranBaju(t, b) menerima masukan 2 buah integer positif, misalnya t (tinggi badan dalam cm) dan
b (berat badan dalam kg) dan menghasilkan kode ukuran baju (1 adalah M, 2 adalah L, 3 adalah XL)
atau kode 4 adalah untuk yang tidak mendapatkan kaos -}

-- REALISASI
-- Coba bagi kasus sehingga logika kondisional sesimpel mungkin, bisa dicari dari batas terbawah
atau batas teratas
```

Soal 4 (UkuranBaju)

```
-- Contoh: memulai dari batas tinggi terbawah hingga teratas
ukuranBaju t b
  -- tinggi <= 150 maka M
  | t <= 150 = 1
  -- tinggi <= 170, jika berat <= 80 maka L, sebaliknya XL
  | t <= 170 = if b <= 80 then 2 else 3
  -- tinggi > 170
  -- berat <= 60 maka L
  | b <= 60 = 2
  -- 60 < berat <= 80 maka XL
  | b <= 80 = 3
  -- tidak memenuhi kategori apapun
  | otherwise = 4

-- APLIKASI
-- ukuranBaju 160 75 = 2
```

Soal 5 (Fibonacci)

Nama File : Fibonacci.hs

Nama Modul : Fibonacci

Buatlah definisi, spesifikasi, dan realisasi fungsi fibonacci yang menerima masukan 1 nilai integer nonnegatif yang merupakan urutan n dalam barisan Fibonacci. Barisan Fibonacci yang dimaksud memiliki suku pertama dan kedua berturut-turut bernilai 1 dan 1.
Realisasi fungsi **fibonacci** menghasilkan nilai dari suku ke n**.**

Contoh:

```
> fibonacci 10
 55
```

Soal 5 (Fibonacci)

```
module Fibonacci where

-- DEFINISI DAN SPESIFIKASI
fibonacci :: Int -> Int
{-
    Menerima masukan 1 nilai integer nonnegatif yang merupakan
    urutan n dalam barisan Fibonacci
-}
```

Soal 5 (Fibonacci)

```
-- REALISASI
fibonacci n
  | n <= 1 = n -- basis
  {-
    Kesalahan umum: Tidak menambahkan n = 0 pada basis.
    Perhatikan bahwa masukan berupa nilai integer non negatif
  -}
  | otherwise = fibonacci (n - 1) + fibonacci (n - 2) -- rekursif

-- APLIKASI
-- fibonacci 10
```

Soal 6 (SumKelipatanX)

Nama File: SumKelipatanX.hs

Header: module SumKelipatanX where

Buatlah definisi, spesifikasi, dan realisasi fungsi **sumKelipatanX** yang menerima masukan dua buah integer positif (integer > 0), misalnya **m** dan **n**, serta sebuah integer positif lain, yaitu **x**, dan menghasilkan jumlah total bilangan kelipatan x di antara m dan n (m dan n termasuk) dengan menggunakan ekspresi rekursif.

Bilangan y disebut kelipatan bilangan x, jika y habis dibagi dengan x.

Prekondisi/syarat/asumsi yang berlaku adalah m <= n dan x <= n.

Contoh aplikasi fungsi dan hasilnya:

No	Aplikasi	Hasil	Keterangan
1.	sumKelipatanX 1 1 1	1	Kelipatan 1 di antara [1..1] adalah 1
2.	sumKelipatanX 1 10 2	30	Kelipatan 2 di antara [1..10] adalah 2, 4, 6, 8, 10 2+4+6+8+10 = 30
3.	sumKelipatanX 5 14 3	27	Kelipatan 3 di antara [5..14] adalah 6, 9, 12 6+9+12 = 27

Haskell

Soal 6 (SumKelipatanX)

```haskell
module SumKelipatanX where

-- DEFINISI DAN SPESIFIKASI
sumKelipatanX :: Int -> Int -> Int -> Int
{-
    Menerima masukan dua bilangan integer positif m dan n, serta
    integer positif lain x.

    Mengembalikan jumlah total bilangan kelipatan x di antara m dan n (m dan n termasuk)
-}
```

Soal 6 (SumKelipatanX)

```
-- REALISASI
sumKelipatanX m n x
  | (m > n) = 0 -- basis
  | (m <= n) && (mod m x == 0) = m + sumKelipatanX (m+1) n x -- rekurens jika m merupakan kelipatan
x
  | (m <= n) && (mod m x /= 0) = sumKelipatanX (m+1) n x -- rekurens jika m bukan kelipatan x

-- APLIKASI
-- sumKelipatanX 1 10 2
```

Soal 7 (Hitung Bensin)

Nama File : HitungBensin.hs

Nama Modul : HitungBensin

Setelah setahun tidak pulang kampung, akhirnya Tuan Vin pun memberanikan diri untuk meminta cuti kepada bosnya. Bosnya sebenarnya ingin langsung menyetujui cuti Tuan Vin. Akan tetapi, dia ingin Tuan Vin membereskan pekerjaannya terlebih dahulu. Pekerjaan Tuan Vin sebenarnya cukup mudah. Dia hanya perlu menyiapkan bensin untuk seluruh kendaraan perusahaannya.

Kendaraan perusahaan Tuan Vin memiliki rute yang sangat unik. Awalnya, kendaraan tersebut akan terletak pada posisi X. Kemudian jika X adalah bilangan genap, kendaraan tersebut akan bergerak ke titik X/2. Jika X adalah bilangan ganjil, kendaraan tersebut akan bergerak ke posisi (3X + 1). Hal ini terus dilakukan sampai kendaraan tersebut sampai ke kantor pusat yang terletak pada posisi 1. Untuk setiap perpindahan posisi, kendaraan tersebut akan menghabiskan bensin sebanyak 1 unit. Jika pada awalnya suatu kendaraan terletak pada posisi 11, kendaraan tersebut akan berpindah ke (11*3 + 1) = 34. Kemudian, kendaraan tersebut kemudian berpindah ke posisi 17,52,26,13,40,20,10,5,16,8,4,2 dan berakhir pada posisi 1 sehingga kendaraan tersebut menghabiskan bensin sebanyak 14 unit.

Tiap harinya, akan ada kendaraan yang berangkat dari posisi A sampai dengan posisi B. Tuan Vin kemudian menjumlahkan banyaknya bensin yang dibutuhkan untuk tiap-tiap kendaraan dari posisi A sampai dengan posisi B. Tuan Vin takut dia tidak sempat menyelesaikan kalkulasinya sebelum hari cutinya tiba. Sebagai teman baik Tuan Vin, Anda pun ingin membantu Tuan Vin dengan membuat sebuah fungsi untuk menghitung bensin yang perlu disiapkan. Fungsi **hitungBensin** menerima 2 buah bilangan bulat, A dan B (A <= B). Fungsi ini kemudian mengeluarkan sebuah bilangan bulat yang menunjukkan konsumsi bensin dari tiap-tiap kendaraan dari A sampai B. Tentu saja sebagai programmer yang baik, Anda harus membuat definisi, spesifikasi, dan realisasi fungsi ini.

Contoh :

> hitungBensin 11 11

14

> hitungBensin 1 10

67

Keterangan:

Pada contoh kedua, 67 didapatkan dengan menjumlahkan bensin yang diperlukan untuk mobil yang mulai pada posisi ke-1, posisi ke-2, posisi ke-3, hingga posisi ke-10

Soal 7 (Hitung Bensin)

```haskell
module HitungBensin where

-- DEFINISI DAN SPESIFIKASI
hitungBensin :: Int -> Int -> Int
{-
    Menghitung hasil jumlah hitungBensin1Rute A, hitungBensin1Rute A + 1, ..., hitungBensin1Rute B
-}

hitungBensin1Rute :: Int -> Int
{-
    Menghitung total bensin yang dibutuhkan untuk bergerak dari posisi X ke posisi 1, dengan
    aturan sebagai berikut:
    Keterangan: N adalah posisi saat ini
    1. Apabila N tidak habis dibagi 2 maka berpindah posisi ke (3 * N) + 1
    2. Apabila N habis dibagi 2 maka berpindah posisi ke N / 2
-}
```

Soal 7 (Hitung Bensin)

```
-- REALISASI
hitungBensin1Rute n
    | n == 1 = 0
    | (mod n 2) == 0 = 1 + (hitungBensin1Rute (div n 2))
    | otherwise = 1 + (hitungBensin1Rute (3 * n + 1))

hitungBensin a b
    | a == b = hitungBensin1Rute a
    | otherwise = hitungBensin1Rute a + (hitungBensin (a + 1) b)


-- APLIKASI
-- hitungBensin 1 10
```

Mudah kan? :D
Ada pertanyaan?