Latihan Awal

Kuliah Pemdek Semester Genap 2015/2016

Fakultas Ilmu Komputer
Universitas Indonesia

Alfan

Latihan

Definisikan fungsi :

```
min3 a b c : Int -> Int -> Int -> Int
```

Latihan

Definisikan fungsi **isTriple** yang menerima 3 buah bilangan bulat positif, dan mengembalikan True jika 3 buah bilangan tersebut adalah ***Pythagorean Triple***.

```
> isTriple 3 4 5
True

> isTriple 2 2 2
False
```

Latihan

Tahun kabisat

Definisikan fungsi **`isLeapYear`** yang menerima sebuah integer yang merepresentasikan tahun. Kemudian, fungsi ini akan mengembalikan **True** jika tahun tersebut adalah kabisat, dan **False** jika sebaliknya.

Tahun kabisat berdasarkan *Proleptic Gregorian Calendar*.

Latihan

Diberikan definisi **threeEqual**

```
threeEqual :: Int -> Int -> Int -> Bool
threeEqual m n p = (m == n) && (n == p)
```

Gunakan **threeEqual** untuk mendefinisikan **fourEqual**:

```
forEqual :: Int -> Int -> Int -> Int -> Bool
```

Latihan

Diberikan definisi **threeDifferent**

```
threeDifferent :: Int -> Int -> Int -> Bool
```

Fungsi ini akan mengembalikan **True** jika ketiga argumen semuanya berbeda !

Latihan

Definisikan sebuah fungsi yang mengembalikan nilai -1 jika sebuah argumen bernilai negative, 1 jika argumen bernilai positif, dan 0 jika argumen bernilai 0.

Gunakan **if-then-else**, dan setelah itu, gunakan **guards**.

Latihan

Definisikan fungsi **charToNum** yang melakukan konversi dari character seperti **‘4’** ke integer **4**.

Jika bukan digit, fungsi akan mengembalikan **0**.

```
charToNum :: Char -> Int
```

Latihan

Definisikan fungsi **jenisHuruf** yang menerima sebuah nilai bertipe **Char**. Fungsi tersebut mengembalikan String “lower case” jika huruf masukan merupakan huruf kecil, dan mengembalikan String “upper case” jika huruf masukan merupakan huruf besar, “bukan huruf” jika karakter tersebut bukan huruf ({A, B, …, Z, a, b, …, z}).

Latihan

Definisikan sebuah fungsi **nor** yang menerima dua buah Boolean, dan mengembalikan **True** jika dua-duanya **bukan True**. Untuk kasus yang lain, fungsi **nor** akan mengembalikan **False**.

Buatlah beberapa definisi ! Dan perlu ada versi yang menggunakan **Pattern Matching**.

Latihan

Definisikan sebuah fungsi yang mengembalikan rata-rata dari 3 buah nilai:

```
averageThree :: Int -> Int -> Int -> Float
```

Latihan

Gunakan definisi **averageThree** untuk mendefinisikan fungsi yang bisa menghitung ada berapa banyak argumen yang nilainya **lebih besar dari rata-ratanya**.

```
howManyAboveAvg :: Int -> Int -> Int -> Int
```

Latihan

Buatlah sebuah fungsi **nextDetik: (Int, Int, Int) -> (Int, Int, Int)** yang menerima waktu dalam tiga buah nilai yaitu **jam**, **menit**, **detik**, dan mengembalikan waktu **satu detik kedepan** dalam representasi yang sama. Nilai jam mempunyai range 0 – 23, nilai menit mempunyai range 0 – 59, dan nilai detik mempunyai range 0 – 59.

- nextDetik (2, 3, 45) -> (2, 3, 46)
- nextDetik (22, 59, 59) -> (23, 0, 0)
- nextDetik (23, 59, 59) -> (0, 0, 0)

Latihan

Definisikan fungsi **`rangeProduct m n`**

Yang mengembalikan:

```
m * (m+1) * ... * (n-1) * n
```

m dan **n** adalah bilangan natural 0, 1, 2, ...

Jika **n < m**, fungsi akan mengembalikan 0.

Latihan

Definisikan fungsi **`factorial`** menggunakan definisi fungsi **`rangeProduct m n`** yang sudah ada !

Latihan

Definisikan fungsi **`mult a b`** yang menghitung perkalian bilangan **a** dan **b** dengan *primitive recursion definition* !

a dan **b** adalah bilangan bulat non-negative (natural numbers).

```
mult 4 0 = 0
mult 4 3 = 12
mult 4 5 = 20
```

Latihan

Definisikan fungsi **`remainder m n`** yang mengembalikan **sisa** dari pembagian **m** dengan **n**.

m dan **n** adalah bilangan bulat positif.

```
remainder 37 10 = 7
remainder 6 4 = 2
```

Apa yang terjadi kalau n = 0 ?

Latihan

Definisikan fungsi **`rpm m n`** yang melakukan simulasi perkalian **m** dan **n** dengan aturan **Russian Peasant Multiplication**.

m dan **n** adalah bilangan bulat non-negative.

Latihan

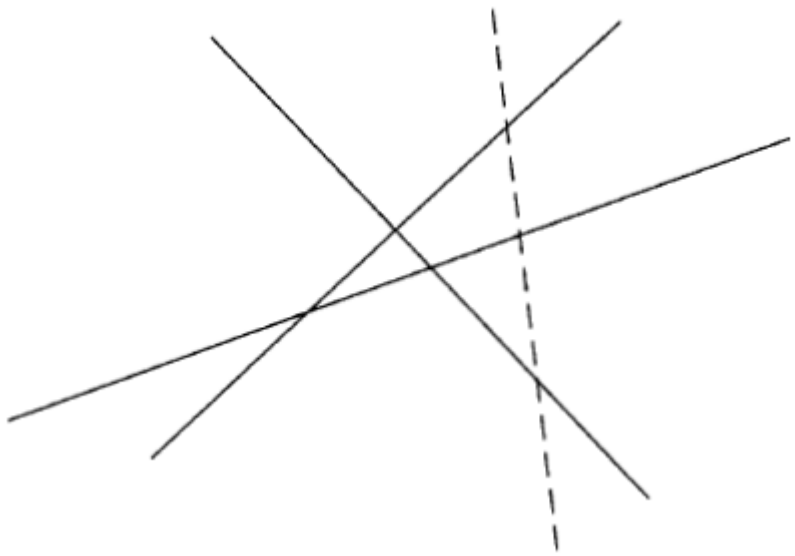
Definisikan fungsi **`power a n`** yang menghitung $\mathbf{a^n}$ dengan cara rekursif.

a dan **n** adalah bilangan bulat non-negative.

Latihan

Geometrical Problem

Definisikan fungsi **`regions n`** yang mampu menghitung berapa **banyak maksimal potongan kertas** yang dapat dihasilkan jika kita menggunakan **n buah garis potong** ?



Latihan

Definisikan fungsi **`isr n`** yang menghitung ***integer square root*** dari sebuah bilangan bulat positif **`n`**.

Integer square root dari **`n`** adalah sebuah bilangan bulat terbesar yang kuadratnya **`<= n`**.

```
isr 16 = 4
isr 15 = 3
```

Latihan

Definisikan fungsi **`sum : int -> int`**

sum (n) = 1 + 2 + 3 + … + (n-1) + n

Latihan

Definisikan fungsi **`fibonacci : int -> int`**

F(0) = 0,

F(1) = 1,

F(n) = F(n-1) + F(n-2)

Latihan

Definisikan fungsi **`kpk : int * int -> int`**

*Kelipatan persekutuan terkecil dari 2 bilangan bulat

Latihan

Definisikan fungsi **`sum : int * int -> int`**

sum (m, n) = m + (m + 1) + (m + 2) + … + (m + (n - 1)) + (m + n)

Latihan – List Comprehension

Definisikan fungsi **`capitalizeLetters str`** yang mengubah semua huruf kecil menjadi huruf besar dan membuang karakter non-huruf.

```
capitalizeLetters :: String -> String
```

`capitalizeLetters “aBcD34eF” = “ABCDEF”`

Anda boleh menggunakan fungsi yang sudah didefinisikan di **Char.hs**

```
isLetter :: Char -> Bool
toUpper :: Char -> Char
```

Latihan – List Comprehension

Definisikan fungsi **`partition p ls`** yang melakukan partisi sebuah list, dimana list hasil partisi yang pertama berisi elemen **≤ p**, dan yang kedua berisi elemen **> p**.

```
partition :: Int -> [Int] -> ([Int], [Int])
```

```
partition 3 [1,2,3,4,5] = ([1,2,3], [4,5])
```

Latihan – List Comprehension

Definisikan fungsi **`divisors n`** yang menampilkan daftar **pembagi** dari sebuah **bilangan bulat positif n**.

```
divisors :: Int -> [Int]
```

```
divisors 12 = [1,2,3,4,6,12]
divisors 34 = [1,2,17,34]
```

Latihan – List Comprehension

Gunakan definisi dari fungsi **`divisors n`** untuk mendefinisikan fungsi **`isPrime n`** !

```
isPrime :: Int -> Bool
```

```
isPrime 0 = False
isPrime (-2) = False
isPrime 2 = True
isPrime 7 = True
```

Hint: n adalah bilangan prima jika pembaginya hanya ada dua yaitu **1 dan n**.

Latihan – List Comprehension

Definisikan fungsi **`matches n ls`** yang mengambil semua kemunculan **`n`** pada list **`ls`**.

```
matches :: Int -> [Int] -> [Int]
```

```
matches 1 [1,2,1,4,5,1] = [1,1,1]
matches 2 [1,2,3] = [2]
matches 5 [1,2,3] = []
```

Latihan – List Comprehension

Gunakan definisi dari fungsi **`matches n ls`** untuk mendefinisikan fungsi **`isElem n ls`** yang mengembalikan **`True`** jika **`n`** merupakan anggota dari list **`ls`**.

```
isElem :: Int -> [Int] -> Bool
```

```
isElem 1 [1,2,1,4,5,1] = True
isElem 2 [1,2,3] = True
isElem 5 [1,2,3] = False
```

Latihan – List Comprehension

Definisikan fungsi **`double ls`** yang menerima list of list of Integer, dan mengembalikan list of list of integer dengan setiap elemen terdalam sudah dikalikan dengan 2.

```
double :: [[Int]] -> [[Int]]
```

```
double [[1,2],[2],[3,5]] = [[2,4],[4],[6,10]]
```

Latihan – List Comprehension (medium)

Definisikan fungsi **`flatten ls`** yang menerima list of list of Integer, dan mengembalikan list of integer.

```
flatten :: [[Int]] -> [Int]
```

```
flatten [[1,2],[2],[3,5]] = [1,2,2,3,5]
```