Lembar Kerja Praktikum KOM120C Pemrograman
11: Functional Programming IV

PETUNJUK PRAKTIKUM

Review HOF

- HOF untuk pengurutan dalam list: sorted, sortBy, dan sortWith.
- HOF untuk tipe data string: split, length, reverse, dll.
- HOF groupBy untuk mengelompokkan list berdasarkan ketentuan tertentu, menghasilkan kembalian berupa map.
- Compiler Scala Online: https://onecompiler.com/scala

Struktur data dan Container pada Scala

1. **Array**
   - Memiliki ukuran **tetap**

```
val myArray = Array(1, 2, 3, 4, 5)
val zeros = Array.fill(5)(0) // Array(0, 0, 0, 0, 0)
```

   - **Mutable** (elemennya dapat diubah)

```
myArray(2) = 10
println(myArray) // Array(1,2,10,4,5)
```

   - Setiap elemen **bertipe sama**
   - Mendukung penggunaan **indeks** untuk mengakses elemen (indeks pertama nol)

```
val thirdElement = myArray(2)
println("Third element: " + thirdElement)
```

2. **List**
   - **Immutable** (elemennya tidak bisa diubah)

```
val myList = List(1, 2, 3, 4, 5)
myList(2) = 10 // Error
```

   - Setiap elemen **bertipe sama**
   - Mendukung operasi menambah elemen list atau menggabungkan dua list

```
val list3 = list1 ++ list2
// atau
```

```
val list3 = list1 :+ list2
// atau
val list3 = list1.concat(list2)
```

- Mendukung penggunaan **head dan tail** untuk mengakses elemen (biasanya diolah dengan algoritma rekursi)

```
// Fungsi rekursif untuk menjumlahkan semua elemen
dalam list integer
def sumList(list: List[Int]): Int = list match {
  case Nil => 0 // Basis kasus: list kosong, jumlahnya
adalah 0
  case head::tail => head + sumList(tail) // Rekursi:
jumlahkan kepala list dengan jumlah dari sisa list
}

// Contoh penggunaan
val myList = List(1, 2, 3, 4, 5)
val total = sumList(myList)
println("Total sum: " + total)
```

3. **Set**
   - **Immutable** (elemennya tidak bisa diubah)
   - Elemennya harus **unik** (tidak duplikat)

```
val mySet = Set(1, 2, 3, 4, 5, 1, 2)
println(mySet) // HashSet(5, 1, 2, 3, 4)
```

4. **Map**
   - Menyimpan pasangan **kunci-nilai**
   - kunci harus **unik**
   - **Immutable** (pasangan kunci-nilai tidak bisa diubah)
   - Diakses berdasarkan **kunci**

5. **Tuple**
   - Setiap elemen bisa memiliki **tipe data berbeda**
   - **Immutable** (elemennya tidak bisa diubah)

TUGAS

1. Diberikan bilangan bulat M, N. M baris berikutnya berisikan N bilangan bulat. Buat program untuk menghitung berapa banyak dari kelompok bilangan setiap baris yang tidak ada duplikatnya,

Contoh Input
3 5 1 5 7 5 1 2 3 13 3 4 1 4 6 8 10
Contoh Output
1
Penjelasan
Kelompok bilangan baris pertama dan kedua memiliki duplikat, sedangkan kelompok bilangan ketiga tidak memiliki duplikat sehingga jumlah kelompok tidak memiliki duplikat ada 1

2. Diberikan bilangan bulat M, N. M baris berikutnya berisikan N bilangan bulat yang merepresentasikan matriks berdimensi M*N. Buat program untuk mentranspose matriks tersebut.

Contoh Input
3 3 1 2 3 4 5 6 7 8 9
Contoh Output
1 4 7 2 5 8 3 6 9

3. Zantos ingin membuat sebuah sistem untuk mencari nilai IPK. terdapat input sebanyak N baris yang berisikan NIM, Nama mahasiswa, Nama mata Kuliah, dan Huruf Mutu yang dipisahkan dengan tanda koma plus spasi (,). Nilai Dari setiap huruf mutu adalah:

A : 4.0 AB: 3.5 B : 3.0 BC: 2.5

```
C : 2.0
D : 1.0
E : 0
```

Anggap input sudah merupakan semua mata kuliah yang diambil oleh setiap mahasiswa. Buatlah program untuk mencari IPK mahasiswa yang ada di input (output IPK dibulatkan menjadi 2 angka dibelakang koma). Output berformat nama IPK dan diurutkan berdasarkan NIM. Dipastikan tidak ada input yang memiliki id dan mata kuliah yang sama.

Contoh Input

```
5
G0401221001, Rangga Rafif, Blockchain, A
G0401221002, Raja Iblis, Blockchain, AB
G0401221003, Zantos Zantoso, Forensik, B
G0401221001, Rangga Rafif, PWN, BC
G0401221001, Rangga Rafif, Kriptografi, A
```

Contoh Output

```
Rangga Rafif 3.50
Raja Iblis 3.50
Zantos Zantoso 3.0
```

Jawab :

1.

```
def input(x : Int) : List[String] = {
  def base(y : Int, daftar : List[String]) : List[String] = {
    if (y < 1) daftar
    else base(y-1, scala.io.StdIn.readLine :: daftar)
  }
  base(x, List())
}

def cek(x : List[Int]) : Int = {
  def ulang(a : List[Int], c : Boolean) : Boolean = {
    if(a == Nil) return c
    else if(a.tail.contains(a.head)) return true
    else return ulang(a.tail, c)
  }
  if(ulang(x, false)) return 0
  else return 1
}

var a = scala.io.StdIn.readLine.split(" ").toList.map(_.toInt)
var M = a.head
var N = a.last
var plat = (0 to M-1).map(input(N).reverse)
var baru = plat.map(x => x.split(" ").toList.map(_.toInt))
var ans = baru.map(x => cek(x)).reduce(_ + _)
println(ans)
```

STDIN

```
35
15751
231334
```

Output:

```
1
```

2.

```
def input(w : Int, x : Int) : Array[Array[Int]] ={
  def ulangan(w : Int, x : Int, y : Int, ar : Array[Array[Int]])
: Array[Array[Int]] ={
    if(y == w) ar
    else{
      var baris = scala.io.StdIn.readLine.split("
").toList.map(_.toInt).toArray
      ar(y) = baris
      ulangan(w, x, y+1, ar)
    }
  }
  var tmp = Array.ofDim[Int](w*x, 0)
  ulangan(w, x, 0, tmp)
}

def tran(m : Int, n : Int, ar : Array[Array[Int]]) :
Array[Array[Int]] ={
  def ulang(a : Int, b : Int, c : Int, d : Int, br :
Array[Array[Int]], cr : Array[Int]): Array[Array[Int]] ={
    if(c == a) br
    else if(d == b){
      br(c) = cr
      ulang(a, b, c+1, 0, br, Array.ofDim[Int](n))
    }
    else{
      cr(d) = ar(d)(c)
      ulang(a, b, c, d+1, br, cr)
    }
  }
  var tmp = Array.ofDim[Int](m*n, 0)
  ulang(m, n, 0, 0, tmp, Array.ofDim[Int](n))
}

var input1 = scala.io.StdIn.readLine.split(" ").toList
var n = input1.head.toInt
var m = input1.tail.head.toInt

var arr = input(n, m)

var transpose = tran(m, n, arr)
transpose.map(x => println(x.mkString(" ")))
```

```
3 3
1 2 3
4 5 6
7 8 9
```

Output:

```
1 4 7
2 5 8
3 6 9
```

3.

```
def penilaian(x : String) : Double = {
  x match {
    case "A" => 4.0
    case "AB" => 3.5
    case "B" => 3.0
    case "BC" => 2.5
    case "C" => 2.0
    case "D" => 1.0
    case "E" => 0.0
  }
}
def redundant(x : List[(String, String)]) : List[(String, String)] = {
    x match{
      case Nil => Nil
      case head::Nil => List(head)
      case head::tail => {
        if(tail.contains(head)) redundant(tail)
        else head::redundant(tail)
      }
    }
}
def masukan(x : Int) : List[String] ={
  def ulangan (x : Int, y : Int, z : List[String]) : List[String] ={
    if(y != x){
      var tmp = scala.io.StdIn.readLine
      ulangan(x, y+1, tmp :: z)
    }
    else z
  }
  ulangan(x, 0, List())
}
def pilih(x : String, y : String) : String ={
  if(x.length < y.length) x
  else y
}
def cariNama(x : String, y : List[(String, String)]) : String ={
  var z = y.filter(_._1 == x)
  z.head._2
}

var n = scala.io.StdIn.readLine.toInt //baca n
var datdat = masukan(n).reverse //baca input n baris dalam string
var baru = datdat.map(x => x.split(", ").toList)
var simpan = baru.map(x => x match {  case List(a, b, c, d) => (a, b)  })
var daftar = redundant(simpan) //daftar nama tanpa perulangan
var findIPK = baru.map(x => x match {  case List(a, b, c, d) => List(a, d)  })
                  .groupBy({case List(x,y) => x})
var hitung = findIPK.map({case (key, value) => (key, value.map(_.reduce((x, y) =>
pilih(x, y)))) })
var IPK = hitung.map({case (key, value) => (key, (value.map(penilaian).reduce(_+_)
/ value.size))})
                .toSeq.sortBy({case (key, value) => key})
IPK.map({case (key, value) => println(""+ cariNama(key, daftar) + " " +
f"$value%.2f")})
```

```
5
G0401221001, Rangga Rafif, Blockchain, A
G0401221002, Raja Iblis, Blockchain, AB
G0401221003, Zantos Zantoso, Forensik, B
G0401221001, Rangga Rafif, PWN, BC
G0401221001, Rangga Rafif, Kriptografi, A
```

Output:

```
Rangga Rafif 3.50
Raja Iblis 3.50
Zantos Zantoso 3.00
```