

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

```

1 object UniqueRowsCounter {
2   def main(args: Array[String]): Unit = {
3     val sc = new java.util.Scanner(System.in)
4     // Baca nilai M dan N
5     val M = sc.nextInt()
6     val N = sc.nextInt()
7     // Fungsi untuk memeriksa apakah sebuah list memiliki elemen duplikat
8     def hasDuplicates(row: List[Int]): Boolean = {
9       row.distinct.length != row.length
10    }
11    // Membaca M baris dan menghitung jumlah baris yang tidak memiliki duplikat
12    var countUniqueRows = 0
13    for (_ <- 1 to M) {
14      val row = (1 to N).map(_ => sc.nextInt()).toList
15      if (!hasDuplicates(row)) {
16        countUniqueRows += 1
17      }
18    }
19    // Cetak hasil
20    println(countUniqueRows)
21  }
22 }
23

```

STDIN

```

3 5
1 5 7 5 1
2 3 13 3 4

```

Output:

```
1
```

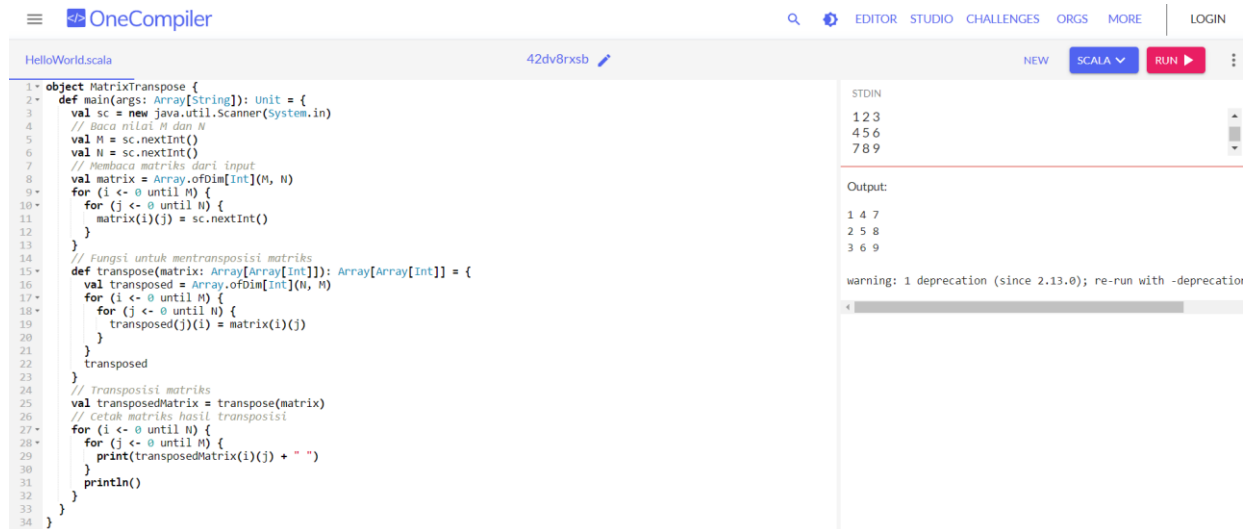
2. Diberikan bilangan bulat M , N . M baris berikutnya berisikan N bilangan bulat yang merepresentasikan matriks berdimensi $M \times N$. Buat program untuk mentranspose matriks tersebut.

Contoh Input

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

Contoh Output

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



```
1 object MatrixTranspose {
2   def main(args: Array[String]): Unit = {
3     val sc = new java.util.Scanner(System.in)
4     // Baca nilai M dan N
5     val M = sc.nextInt()
6     val N = sc.nextInt()
7     // Membaca matriks dari input
8     val matrix = Array.ofDim[Int](M, N)
9     for (i <- 0 until M) {
10      for (j <- 0 until N) {
11        matrix(i)(j) = sc.nextInt()
12      }
13    }
14    // Fungsi untuk mentransposisi matriks
15    def transpose(matrix: Array[Array[Int]]): Array[Array[Int]] = {
16      val transposed = Array.ofDim[Int](N, M)
17      for (i <- 0 until M) {
18        for (j <- 0 until N) {
19          transposed(j)(i) = matrix(i)(j)
20        }
21      }
22      transposed
23    }
24    // Transposisi matriks
25    val transposedMatrix = transpose(matrix)
26    // cetak matriks hasil transposisi
27    for (i <- 0 until N) {
28      for (j <- 0 until M) {
29        print(transposedMatrix(i)(j) + " ")
30      }
31      println()
32    }
33  }
34 }
```

STDIN

123
456
789

Output:

1 4 7
2 5 8
3 6 9

warning: 1 deprecation (since 2.13.0); re-run with -deprecation

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
```

```
import scala.collection.mutable
```

```
object Main extends App {
```

```
// Fungsi untuk mengonversi huruf mutu menjadi nilai
def convertGradeToPoints(grade: String): Double = grade 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
  case _ => 0.0 // default jika tidak ada huruf mutu yang sesuai
}
```

```
// Fungsi untuk menghitung IPK
def calculateIPK(data: List[String]): List[(String, Double)] = {
  val studentRecords = mutable.Map[String, (String, Double, Int)]()
```

```
for (entry <- data) {
  val Array(nim, name, _, grade) = entry.split(", ") // Mengabaikan mata kuliah
  val points = convertGradeToPoints(grade)
```

```
  if (studentRecords.contains(nim)) {
    val (existingName, totalPoints, courses) = studentRecords(nim)
    studentRecords(nim) = (existingName, totalPoints + points, courses + 1)
  } else {
```

```
    studentRecords(nim) = (name, points, 1)
  }
}

studentRecords.map { case (nim, (name, totalPoints, courses)) =>
  (name, totalPoints / courses)
}.toList
}

// Input data (sample input)
val input_data = List(
  "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"
)

// Hitung IPK
val results = calculateIPK(input_data)

// Output hasil
if (results.isEmpty) {
  println("No data provided.")
} else {
  results.sortBy(_._1).foreach { case (name, ipk) =>
    println(s"$name ${"% .2f".format(ipk)}")
  }
}
}
```

Output:

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