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

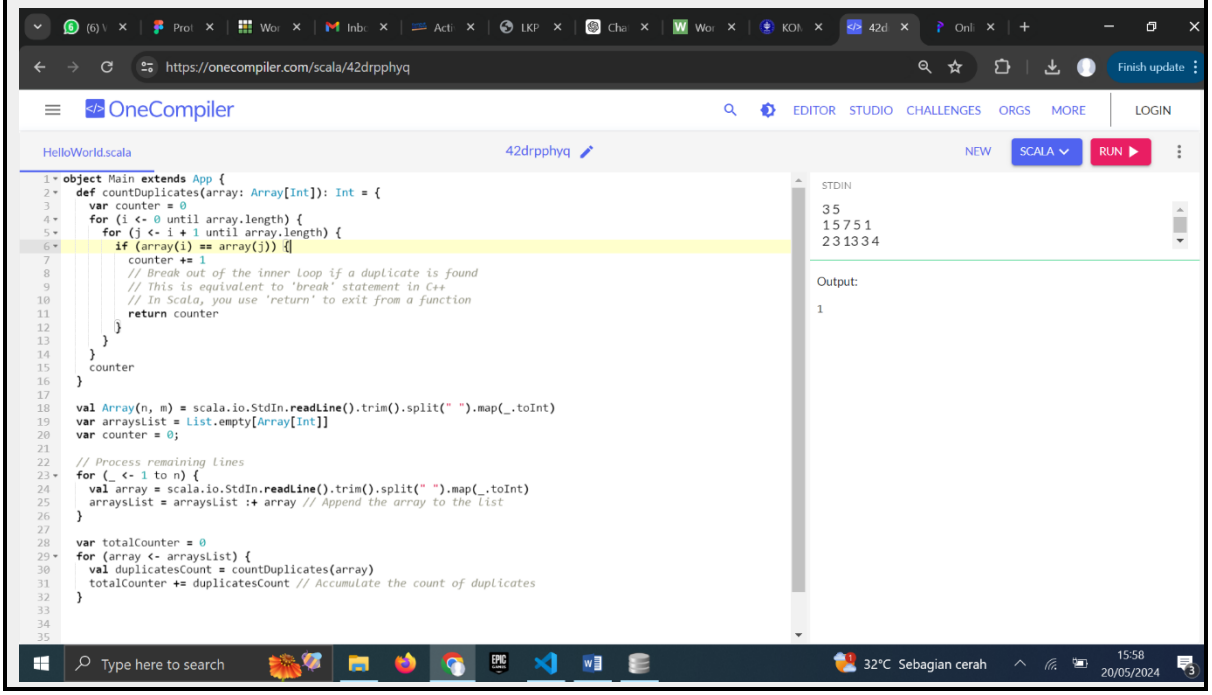
Kelompok bilangan baris pertama dan kedua memiliki duplikat, sedangkan kelompok bilangan ketiga tidak memiliki duplikat sehingga jumlah kelompok tidak memiliki duplikat ada 1

Kode:

```
object Main extends App {
  def countDuplicates(array: Array[Int]): Int = {
    var counter = 0
    for (i <- 0 until array.length) {
      for (j <- i + 1 until array.length) {
        if (array(i) == array(j)) {
          counter += 1
          // Break out of the inner loop if a duplicate is found
          // This is equivalent to 'break' statement in C++
          // In Scala, you use 'return' to exit from a function
          return counter
        }
      }
    }
    counter
  }

  val Array(n, m) = scala.io.StdIn.readLine().trim().split(" ").map(_.toInt)
  var arraysList = List.empty[Array[Int]]
  var counter = 0;

  // Process remaining lines
```

```
for (_ <- 1 to n) {
  val array = scala.io.StdIn.readLine().trim().split(" ").map(_.toInt)
  arraysList = arraysList :+ array // Append the array to the list
}

var totalCounter = 0
for (array <- arraysList) {
  val duplicatesCount = countDuplicates(array)
  totalCounter += duplicatesCount // Accumulate the count of duplicates
}
```

Dokumentasi kode dan hasil:



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.

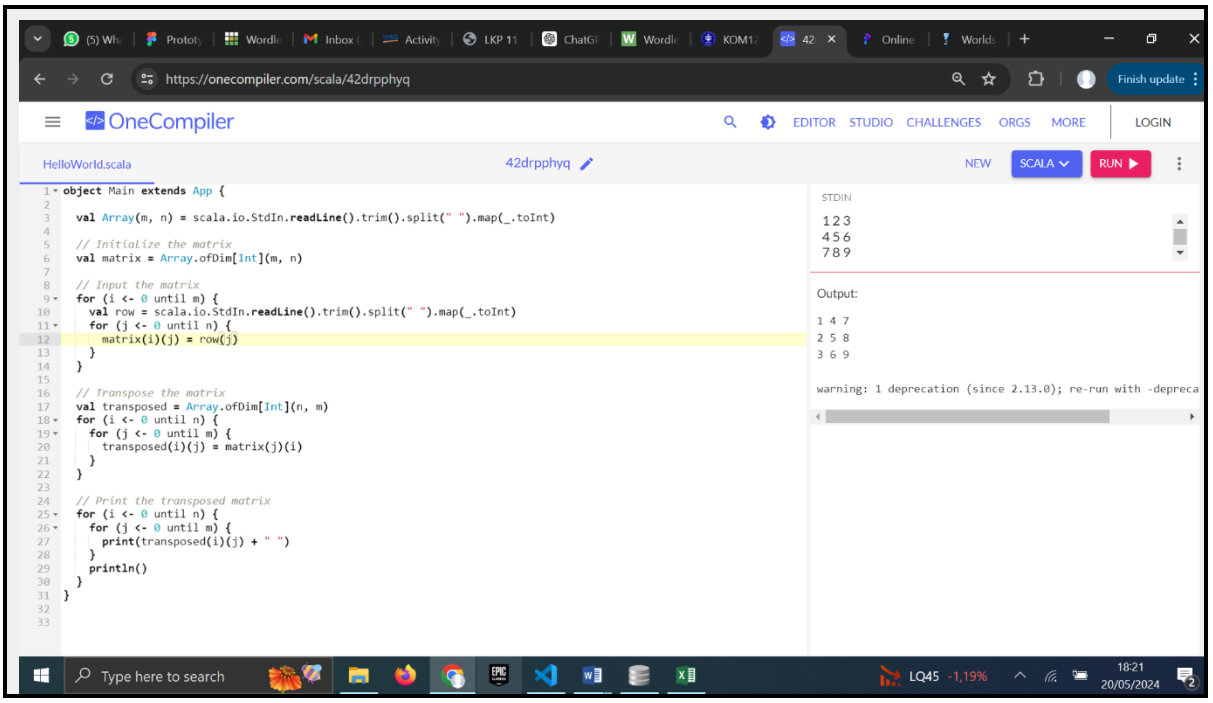
Contoh Input
3 3 1 2 3 4 5 6 7 8 9
Contoh Output

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

Kode:

```
object Main extends App {

  val Array(m, n) = scala.io.StdIn.readLine().trim().split("
").map(_.toInt)

  // Initialize the matrix
  val matrix = Array.ofDim[Int](m, n)

  // Input the matrix
  for (i <- 0 until m) {
    val row = scala.io.StdIn.readLine().trim().split("
").map(_.toInt)
    for (j <- 0 until n) {
      matrix(i)(j) = row(j)
    }
  }

  // Transpose the matrix
  val transposed = Array.ofDim[Int](n, m)
  for (i <- 0 until n) {
    for (j <- 0 until m) {
      transposed(i)(j) = matrix(j)(i)
    }
  }

  // Print the transposed matrix
  for (i <- 0 until n) {
    for (j <- 0 until m) {
      print(transposed(i)(j) + " ")
    }
    println()
  }
}
```

Dokumentasi kode dan hasil:



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
Kode:
object Main extends App {
  val n = scala.io.StdIn.readLine().trim().toInt
  // Initialize an empty map to store cumulative grades for each
NIM
  var cumulativeGrades: Map[String, Double] =
Map().withDefaultValue(0.0)
  var nimCounts: Map[String, Int] = Map().withDefaultValue(0)
  var studentNames: Map[String, String] = Map() // Map to store
names for each NIM

  // Loop through each row and read it
  for (_ <- 1 to n) {
    val rowString = scala.io.StdIn.readLine().trim()
    val Array(nim, name, _, grade) =
rowString.split(",").map(_.trim())
    // Accumulate grades for each distinct NIM
    val numericGrade = grade match {
      case "A"  => 4.0
      case "AB" => 3.5
      case "B"  => 3.0
      case "BC" => 2.5
      case "C"  => 2.0
```

```
      case "D"  => 1.5
      case "E"  => 1.0
      case _    => 0.0 // Handle unknown grades here
    }
    cumulativeGrades += (nim -> (cumulativeGrades(nim) +
numericGrade))
    nimCounts += (nim -> (nimCounts(nim) + 1))
    studentNames += (nim -> name) // Store the name for the
corresponding NIM
  }

  // Print the cumulative grades along with student names
  cumulativeGrades.foreach { case (nim, grade) =>
    val name = studentNames(nim) // Get the name corresponding to
the NIM
    val hasil = grade / nimCounts(nim)
    println(s"$name ${"%.2f".format(hasil)}") // Format hasil to
two decimal places
  }
}
```

Dokumentasi kode dan hasil: