Übung 7 – Aufgabe

Aufgabe 1: Implementieren Sie die folgenden Aufgabenstellungen:

a) Schreiben Sie eine Funktion moduloMap(l:List[Int], mod_value:Int):Map[Int,List[Int]], die aus einer Liste von Zahlen, eine Map erzeugt, deren Schlüssel ein Int-Wert ist, der sich aus der Modulo-Rechnung des Listenwertes mit mod_value ergibt. Zu den Schlüsselwerten werden dann alle Ints der Ausgangsliste innerhalb einer Liste gespeichert: z.B.:

val l= List(1,4,5,7,8,9)
moduloMap(l,3) ergibt dann:

Map(1 -> List(7, 4, 1), 2 -> List(8, 5), 0 -> List(9))

Benutzen Sie dafür nur einen Aggregationsoperator!

b) Gegeben sei eine Liste von Wörtern. Schreiben Sie eine Funktion countLetters(l:List[String]):Map[Int,Int], die aus der Liste von Wörtern eine Map generiert, in der gespeichert wird, wie viele Wörter es mit einer entsprechenden Buchstabenzahl (Schlüssel) gibt: z.B.:

val w=List("Hallo","das","sind","ein","paar", "Wörter")
countLetters(w)

ergibt: Map(5 -> 1, 3 -> 2, 4 -> 2, 6 -> 1)

Benutzen Sie dafür nur eine Aggregationsfunktion.

c) Wandeln Sie die Funktion so um, dass nicht die Anzahl der Wörter gespeichert wird, sondern die Wörter selbst. Benutzen Sie nur eine Aggregationsfunktion.

Aufgabe 2: Implementieren Sie die folgenden Funktionen:

a) Schreiben Sie eine Funktion avgNumbers(l:List[Int]):Map[Boolean, Double]. Die Funktion soll aus der Liste die Durchschnittswerte der geraden und der ungeraden Zahlen bilden. Der Schlüsselwert soll dabei ein Boolean sein, der bei true alle geraden Werte zusammenfasst und false bei allen ungeraden: z.B.:

val l2= List(1,4,5,7,8,9)

avgNumbers(l) ergibt:

Map(false -> 5.5, true -> 6.0)

Aufgabe 3: Gegeben sei die folgende Liste von Tupeln:

val stundenProtokoll:List[(String, Int, List[Int])]= List(("Hans",1,List(7,9,4,12,8)),
("Hans",2,List(8,2,10,12,12)), ("Hans",3,List(8,8,8,7,9)),("Hans",4,List(8,9,10,9,8)),
("Monika",1,List(6,9,8,7,8)),("Monika",2,List(7,9,8,6,9)), ("Monika",3,List(12,9,12,8,7)),
("Monika",4,List(6,9)),("Kevin",1,List(6,9,8,7,8)),("Kevin",2,List(7,8,8,7,9)),
("Kevin",3,List(12,3,12,3,2)),("Kevin",4,List(12,3)))

In der Liste wird aufgeführt, welche Mitarbeiter (Stelle 1), in welcher Kalenderwoche (Stelle 2), wie viele Stunden (Stelle 2) gearbeitet hat. Die Stunden sind dabei als Liste repräsentiert

– jeder Tag wird als Listeneintrag aufgeführt. Der Name identifiziert einen Mitarbeiter eindeutig, d.h. die Kombination mit Kalenderwoche

Extrahieren Sie die folgenden Informationen aus der Liste. Benutzen Sie dabei nur Higher Order Functions.

a) Schreiben Sie eine Funktion maxWorkPerWeek(l:List[(String, Int, List[Int])]):(String,Int,Int). Sie soll aus der Liste extrahieren, welcher Mitarbeiter in welcher Woche am meisten gearbeitet hat. Ergebnis soll ein Tripel sein, bestehend aus dem Namen, der Kalenderwoche und der Stundenanzahl.

b) Schreiben Sie eine Funktion maxWork(l:List[(String, Int, List[Int])]):(String,Int). Diese Funktion soll berechnen, wer insgesamt am meisten gearbeitet hat. Ergebnis soll ein Tupel sein, dass aus dem Namen und der Stundenanzahl besteht.

c) Schreiben sie eine Funktion daylyMax(l:List[(String, Int, List[Int])]):(Int, List[String]), die ermittelt, was die maximale Stundenzahl ist, die an einem Tag gearbeitet wurde und wer alles diese maximale Stundenzahl gearbeitet hat. Ergebnis soll ein Tupel sein, dessen erstes Element die Stundenzahl ist und dessen zweites die Liste von Namen, die diese Stundenzahl an einem Tag gearbeitet haben.