**Klausurvorbereitung und Belegthemen**

**Aufgabe 1 (Generische Programmierung und Exception Handling):**
Gegeben sei die folgende grundlegende Definition „Konfigurationsmap
val config=Map[String, Any]("port"->1000, "address"->"superservice@htw", "delay"→5.5)

Der Typ der Konfigurationsmap ist [String, Any], da üblicherweise einem textuellem Schlüssel ein beliebiger Wert (vom Typ her) zugeordnet wird. Werden die Daten aus der Map geholt, so muss in der Regel ein Typcast und eine Fehlerbehandlung (Schlüssel nicht vorhanden) durchgeführt werden.

Ermitteln Sie die folgenden Werte aus der Map:
a) Lassen Sie sich die Portnummer geben, in dem Sie die Map über den (...)-Operator das Element lesen. Behandeln Sie den Fehler, der auftritt, wenn das Schlüsselement nicht vorhanden ist.
val port:Int= ...

b) Lassen Sie sich die Portnummer geben, in dem Sie die Map über die Operation get auslesen. Diese Liefert eine Option zurück, die sie dann über die get-Operation auslesen. Behandeln Sie den Fehler, der auftritt, wenn das Schlüsselement nicht vorhanden ist.
val port:Int= …

c) Lassen Sie sich Portnummer und Adresse geben, in dem Sie den get-Operator und Pattern Matching verwenden.
val (port,address):(Int,String)=...

**Aufgabe 2 (Gruppierungen):**
In Entenhausen wollen nach der üppigen Weihnachtszeit, einige Bewohner ihr Gewicht reduzieren, in dem Sie dIhre Kalorienaufnahme pro Tag verringern. Dazu führen sie ein Tagebuch, bei dem sie aufschreiben, zu welcher Mahlzeit sie wie viele Kalorien zu sich genommen haben. Der entstandene Datensatz hat die folgende Struktur:

```
val calories: List[(String, String, List[(String, Int)])] =
List(("Donald Duck", "2022-01-01",List(("Frühstück",800), ("Mittag",
700), ("Snack",200), ("Abendbrot", 500))), ("Donald Duck", "2022-01-
02",List(("Frühstück",700), ("Mittag", 650), ("Abendbrot", 520))),
("Donald Duck", "2022-01-03",List(("Frühstück",800), ("Mittag", 700),
("Snack",200), ("Abendbrot", 500), ("Snack",150))),("Donald Duck",
"2022-01-04",List(("Frühstück",850), ("Mittag", 900), ("Snack",500),
("Snack", 400))),("Donald Duck", "2022-01-05",List(("Frühstück",600),
("Mittag", 700), ("Snack",200), ("Abendbrot", 100))), ("Dagobert Duck",
"2022-01-01",List(("Frühstück",300), ("Mittag", 500), ("Snack",100),
("Abendbrot", 200))), ("Dagobert Duck", "2022-01-02",
List( ("Frühstück",200), ("Mittag", 300), ("Snack",400), ("Abendbrot",
200))), ("Dagobert Duck", "2022-01-03",List(("Frühstück",800),
("Mittag", 700), ("Snack",200), ("Snack", 200))), ("Dagobert Duck",
"2022-01-04",List(("Frühstück",200), ("Mittag", 300), ("Snack",200),
("Snack", 500))), ("Dagobert Duck", "2022-01-05",List(("Frühstück",200),
("Mittag", 700), ("Abendbrot", 500))))
```

Für jeden Tag gibt es einen Eintrag der folgenden Struktur: (Person, Tag, Liste der Mahlzeiten). Die Liste der Mahlzeiten besteht aus Tupeln mit den Namen der Mahlzeit (Frühstück, Mittag, Abendbrot, Snack) sowie der aufgenommenen Kalorien.

Benutzen Sie keine Elemente der imperativen Programmierung wie veränderliche Variablen!

a) Schreiben Sie die Funktion averageCaloriesByMeal(l:List[(String, String, List[(String, Int)])]):Map[String,Int]. Sie soll berechnen, wie viel Kalorien durchschnittlich bei den einzelnen angegebenen Mahlzeiten aufgenommen werden. Ergebnis soll eine Map sein, dessen Schlüssel die Mahlzeit und dessen Wert die Kalorien sind. Verwenden Sie dabei den Gruppierungsoperator für die Aggregation der Map.

b) b) Schreiben Sie eine Funktion averageCaloriesMealPerPerson(l: List[(String, String, List[(String, Int)])]):Map[(String,String),Double], die ermittelt, wie viele Kalorien pro Mahlzeittyp („Snack“,...) von den einzelnen Einwohnern zu sich genommen wurden. Ergebnis soll eine Map sein, dessen Schlüssel der Name und der Essenstyp und dessen Wert der Durchschnitt ist. Verwenden Sie dabei den Gruppierungsoperator für die Aggregation der Map.

c) Schreiben Sie eine Funktion averageCaloriesPerDayAndPerson(l: List[(String, String, List[(String, Int)])]):Map[String,Double], die ermittelt, wie viele Kalorien pro Tag von den einzelnen Einwohnern im Durchschnitt zu sich genommen wurde. Ergebnis soll eine Map sein, dessen Schlüssel der Name und dessen Wert der Durschschnitt ist. Verwenden Sie dabei den Gruppierungsoperator für die Aggregation der Map.