7.2 Structural Sharing

a) Was steht hinter dem Prinzip des Structural Sharing in Scala? Erläutern Sie das Prinzip an einem Beispiel!

Durch structural sharing werden Daten von Unveränderlichen Datentypen (z.B. Listen) wieder verwendet, wenn sie erneut benötigt werden, anstelle neu gebaut und in einem anderen Speicherbereich abgelegt zu werden. Dadurch spart man zum einen Zeit (Konstruktionszeit) und Speicher.

```
val list_1_2 = List(1,2)
val list_1_2_3_4 = 3 :: 4 :: list_1_2 // uses the same memory for 1 & 2
like list_1_2
```

b) Welchen Vorteil bietet es? Welche Nachteile liegen vor?

Vorteile beantwortet in a: Zeit- & Speichervorteile

Strcutural Sharing funktioniert nur für unveränderliche Datentypen. Soll, z.B. eine Liste die gleichen Elemente in anderer Reihenfolge enthalten, kann strcutural sharing nicht verwendet werden und es wird neuer Speicherbereich belegt.

c) Welche Art von Datentypen profitieren besonders vom Structural Sharing?

Alle Arten von unveränderlichen Aufzählungen: Listen, Sets, Maps.