7.3 Streams

a) Was unterscheidet Streams von Listen?

Streams sind unveränderlich und theoretisch in der Größe nicht begrenzt. Ströme sind allerdings Spezialisierungen von Listen! Streams brauchen oft auch keinen Speicherplatz bis sie evaluiert werden. Beim konstruieren wird eine Funktion festgelegt mit der die Evaluierung erfolgen soll.

b) Nennen Sie drei mögliche Anwendungsfälle (unabhängig von Zahlfolgen), bei denen Sie die Anwendung von Streams für sinnvoll erachten!

1. Bei potentiell sehr großen Listen.
2. Bei jeder Art von Filterlogik.
3. Wenn zip verwendet werden soll, spricht wenn z.B. ein Koordinaten-Tuple aus zwei Streams gebaut werden soll.

c) Erklären sie anhand von Streams (und einem selbst gewählten Beispiel) wie das sliding funktioniert.

Sliding schneidet Listen (und damit auch Streams) in eine Liste von Sublisten, bzw. in einen Iterator der über die Sublisten läuft.

```
val stream: Stream[Int] = 1 #:: 2 #:: 3 #:: 4 #:: s;

// sliding(number) schneidet den stream in Sublisten der Länge number,
wobei das erste Element jeder Liste auch das letzte Element der vorherigen
Liste ist.
// toList verwandelt den Iterator in eine Liste.
val stream_slide_1 = stream.sliding(2).toList //
List(List(1,2),List(2,3),List(3,4))

// der zweite Paramter gibt einen Abstand zwischen den Elementen an. 1 ist
default, 2 ist genau der Abstand der so slidet, dass jedes Element in
genau einer Subliste auftauch. Alles größer 2 bewirkt, dass Elemente in
den Sublisten ausgelassen werden.
val stream_slide_2 = stream.sliding(2,2).toList //
List(List(1,2),List(3,4))
```