Übungsserie 1: Lösungen

Aufgabe 1: Arithmetische Folgen

Bestimmen Sie das n-te Glied (a_n) der folgenden Folgen in *rekursiver*, *iterativer* und *expliziter* Form

(jeweils in der Polynom-Normalform: $a_n x^n + a_{n-1}x^{n-1} + \ldots + a_2 x^2 + a_1 x + a_0$).

Definition:

Rekursiv:	$a_n = a_{n-1} + d;\ a_1 = c$
Iterativ:	$a_n = a_1 + \sum_{i=2}^{n} d$
Explizit:	$a_n = f(n)$

Die Folgen:

(a) 1, 2, 3, 4, ...
(b) 5, 13, 21, 29, …

Lösung:

	Rekursiv:		*Iterativ:*		*Explizit:*
(a)	$a_1 = 1,\ a_n = a_{n-1} + 1$	=	$a_1 + \sum_{i=2}^{n} 1 = 1 + \sum_{i=2}^{n} 1$	=	n
(b)	$a_1 = 5,\ a_n = a_{n-1} + 8$	=	$a_1 + \sum_{i=2}^{n} 8 = 5 + \sum_{i=1}^{n-1} 8$	=	$5 + 8(n-1) = 8n - 3$

Aufgabe 2: Arithmetische Reihen

Bestimmen Sie die Summenformeln ($s_n = \sum_{i=1}^{n} a_i$) der Folgen (a) und (b) aus Aufgabe 1 in *rekursiver*, *iterativer* und *expliziter* Form

(jeweils in der Polynom-Normalform: $a_n x^n + a_{n-1}x^{n-1} + \ldots + a_2 x^2 + a_1 x + a_0$).

Hinweis: Allgemeine Summenformel: $s_n = \frac{n(a_1 + a_n)}{2}$

Lösung:

	Rekursiv:		*Iterativ:*		*Explizit:*
(a)	$s_1 = 1,\ s_n = s_{n-1} + n$	=	$\sum_{i=1}^{n} i$	=	$\frac{n(n+1)}{2} = \frac{n^2+n}{2} = \frac{1}{2}n^2 + \frac{1}{2}n$
(b)	$s_1 = 5,\ s_n = s_{n-1} + 8n - 3$	=	$\sum_{i=1}^{n} (8i - 3)$	=	$\frac{n(5 + (8n-3))}{2} = 4n^2 + n$

Aufgabe 3: Rekursion

Es soll ein Programm erstellt werden, dass rekursiv die Summe $\sum_{i=0}^{n} i$ berechnet.

(Analog zu Beispiel `factorial()` im Folien-Skript (Folie 16).

In der Ausgangslage (ILIAS: *U01_Java_AS.zip* resp. *U01_Python_AS.zip*) soll dazu die Methode ***`recursiveSum()`*** resp. ***`recursive_sum()`*** implementiert werden (siehe auch entsprechende Markierung im Source-Code: `TODO: Implement here...`).

Lösung:

siehe: ***`uebung01/ml/aufgabe03/RecursiveSum.java`***
resp.: ***`uebung01/ml/aufgabe03/recursive_sum.py`***

Aufgabe 4: Laufzeit-Analyse anhand Insertion-Sort

Es soll der *Insertion-Sort* Algorithmus gemäss Folien-Skript "In-Place Insertion-Sort" (Folie 13) implementiert werden.
In der Ausgangslage muss dazu nur die Methode *`insertionSort()`*resp. *`insertion_sort()`* erweitert werden.

Nach der Implementierung sollen die gemessenen Laufzeiten beobachtet werden.
Entsprechen diese den Erwartungen?

Hinweis für Java:
Damit der *JIT-Hotspot-Compiler* der *Virtual Machine* die Messresultate nicht verfälscht, muss diese im *Interpreter-Mode* laufen: *`-Xint`*
(z.B. in Eclipse: *Run Configurations...>(x)=Arguments>VM arguments: `-Xint`*)

Lösung:

siehe: ***`uebung01/ml/aufgabe04/InsertionSort.java`***
resp.: ***`uebung01/ml/aufgabe04/insertion_sort.py`***