411 Datenstrukturen und Algorithmen entwerfen und anwenden

Titel	LBV Modul 411-3 - 3 Elemente - Praktische Umsetzungsarbeit, Praktische Umsetzungsarbeit, Praktische Umsetzungsarbeit
Institution	Informatikschule Olten GmbH
Übersicht	Problemstellungen sowohl als Rekursion, als auch als Loop umsetzen. Sortier-Algorithmen kennen und einsetzen. Verkettete Liste verstehen und erweitern. Abstrakte Datentypen richtig einsetzen. Analyse von vorgegebenen Programm-Codes bzgl Funktionalität und Laufzeitkomplexität.
Ergänzung	

Teil	**1**
Gewichtung	30%
Richtzeit (Empfehlung)	2
Element-Beschreibung	1) Aufgabenstellung (z.B. Summe der ersten n positiven, ganzen Zahlen) sowohl als Rekursion, als auch als Loop umsetzen und deren Laufzeitkomplexität angeben (O-Notation). Visualisieren der Rekursion (z.B. für die Summe der ersten 3 Zahlen) auf dem Stack. 2) Vorgegebenen Pseudocode umsetzen und dessen Funktionalität erkennen (z.B. ggT). 3) Vorgegebenen Sortier-Algorithmus (z.B. InsertionSort) ohne Computer analysieren und für jeden Durchgang die Situation des Arrays aufzeigen sowie die Laufzeitkomplexität angeben.
Hilfsmittel	1) und 2) frei 3) ohne Hilfsmittel
Bewertung	1) je 3 Punkte für Rekursion und Loop, je 1 Punkt für Laufzeitkomplexität, 2 Punkte für Stack-Visualisierung = 10 Punkte 2) 4 Punkte für die Implementation des Pseudocodes inkl. Test-Programm, 1 Punkt für das Erkennen der Funktionalität = 5 Punkte 3) 4 Punkte für die korrekte Darstellung der Arrays, 1 Punkt für Laufzeitkomplexität = 5 Punkte
Praxisbezug	Vor- und Nachteile von Rekursion und Loop kennen. Umsetzen von Rekursionen und Loops. Laufzeitkomplexitäten beurteilen können (z.B. was passiert wenn statt 1'000 später 1'000'000 Elemente sortiert werden müssen). Fremde Code-Stücke analysieren können. Verhalten von Programm-Code bzgl. Stack und Heap erkennen können.

Teil	**2**
Gewichtung	40%
Richtzeit (Empfehlung)	2
Element-Beschreibung	1) Greedy-Algorithmus umsetzen und erklären (z.B. minimale Anzahl Münzen für einen vorgegebenen

	Betrag). 2) Eigene Klasse erstellen und Comparable-Interface implementieren inkl. Test-Programm. 3) Aufgabenstellung (z.B. x^y) sowohl als Rekursion, als auch als Loop umsetzen. 4) Vorgegebenen Algorithmus (z.B. Min/Max in einem Array) ohne Computer analysieren und Funktionalität erkennen und eine Zusatzfrage zum Code beantworten (z.B. Anzahl Vergleiche für ein Array der Grösse n)
Hilfsmittel	1) bis 3) frei 4) ohne Hilfsmittel
Bewertung	1) 2 Punkte für die Umsetzung, 1 Punkt für die Erklärung = 3 Punkte 2) je 1 Punkt für die Klasse, die Interface-Implementation und das Test-Programm = 3 Punkte 3) je 1 Punkte für Rekursion und Loop = 2 Punkte 4) je 1 Punkt für das Erkennen der Funktionalität und das Beantworten der Zusatzfrage = 2 Punkte
Praxisbezug	Greedy-Algorithmus kennen und einsetzen können. Interface verstehen und implementieren können. Umsetzen von Rekursionen und Loops. Fremde Code-Stücke analysieren können.

Teil	**3**
Gewichtung	30%
Richtzeit (Empfehlung)	2
Element-Beschreibung	1) Vorgegebene Implementation einer verketteten Liste erweitern und erklären (visualisieren) 2) Geeigneten abstrakten Datentyp aufgrund von vorgegebenen Daten wählen, einsetzen und erklären.
Hilfsmittel	frei
Bewertung	1) Gesamthaft 7 Punkte für z.B. 2 Erweiterungen, 3 Punkte für die Visualisierungen = 10 Punkte 2) 3 Punkte für die Wahl des geeigneten Datentyps, 3 Punkte für den korrekten Einsatz nach Vorgabe, 2 Punkte für die Erklärungen = 8 Punkte
Praxisbezug	Datenstrukturen verstehen und richtig einsetzen können.

Publiziert: 08.07.2016 13:34:34
Ablaufdatum: Kein Ablaufdatum