

INFORMATIK I

Inoffizielle Probeklausur
Lösungsvorschlag

15. JANUAR 2005

1. Verständnis- und Wissensfragen (8 Punkte)

1.1 Wahr-/Falsch-Fragen (6 Punkte)

Kreuzen Sie an, ob die Aussage wahr (*W*) oder falsch (*F*) ist.

Hinweis: Jedes korrekte Kreuz zählt 0,5 Punkte, jedes falsche Kreuz bewirkt 0,5 Punkte Abzug! Die Teilaufgabe wird mindestens mit 0 Punkten bewertet.

- | | | |
|----------------------------------------------|----------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <input type="checkbox"/> <i>W</i> | <input checked="" type="checkbox"/> <i>F</i> | Quicksort hat im schlimmsten Fall eine Theoretische Effizienz von $O(n \log n)$. |
| <input type="checkbox"/> <i>W</i> | <input checked="" type="checkbox"/> <i>F</i> | Ein Algorithmus mit einer Theoretischen Effizienz von $O(n^2)$ ist für alle Eingaben schneller als ein Algorithmus mit einer Theoretischen Effizienz von $O(n^3)$. |
| <input type="checkbox"/> <i>W</i> | <input checked="" type="checkbox"/> <i>F</i> | Die Taktfrequenz des Prozessors in einem Von-Neumann-Rechner stellt den sogenannten „Flaschenhals“ dar. |
| <input type="checkbox"/> <i>W</i> | <input checked="" type="checkbox"/> <i>F</i> | Signalparameter unterliegen keinerlei zeitlichen Veränderungen. |
| <input checked="" type="checkbox"/> <i>W</i> | <input type="checkbox"/> <i>F</i> | Folgender Java-Code liefert keinen Fehler: <code>for(;;) {;}</code> |
| <input type="checkbox"/> <i>W</i> | <input checked="" type="checkbox"/> <i>F</i> | In einem Hamiltonschen Kreis ist jede Ecke eines ungerichteten Graphen mindestens einmal oder mehrmals enthalten. |
| <input type="checkbox"/> <i>W</i> | <input checked="" type="checkbox"/> <i>F</i> | Ein Graph ist genau dann azyklisch, wenn er keinen Eulerschen Zyklus enthält. |
| <input checked="" type="checkbox"/> <i>W</i> | <input type="checkbox"/> <i>F</i> | Ein gerichteter Baum hat genau eine Ecke e mit $ \bullet e = 0$. |
| <input checked="" type="checkbox"/> <i>W</i> | <input type="checkbox"/> <i>F</i> | Der Euklidische Algorithmus berechnet den größten gemeinsamen Teiler zweier natürlicher Zahlen. |
| <input type="checkbox"/> <i>W</i> | <input checked="" type="checkbox"/> <i>F</i> | Klassennamen werden in einem UML-Klassendiagramm unterstrichen dargestellt. |
| <input checked="" type="checkbox"/> <i>W</i> | <input type="checkbox"/> <i>F</i> | Die Sonne steuert das Wachstum der Pflanzen. |
| <input type="checkbox"/> <i>W</i> | <input checked="" type="checkbox"/> <i>F</i> | Der Lautstärkeregel des Radios regelt dessen Lautstärke. |

1.2 Wissensfragen (2 Punkte)

a) Definieren Sie Information im Sinne der Vorlesung. (1 Punkt)

Lösung:

Interpretation einer Nachricht auf der Grundlage eines Bezugssystems.

b) Definieren Sie System im Sinne der Vorlesung. (1 Punkt)

Lösung:

Kollektion von Gegenständen, die in einem inneren Zusammenhang stehen, samt den Beziehungen zwischen diesen Gegenständen.

2. Algebren (10 Punkte)

2.1 Kantorowitsch-Baum (5 Punkte)

Gegeben sei folgender arithmetischer Ausdruck in Infix-Form:

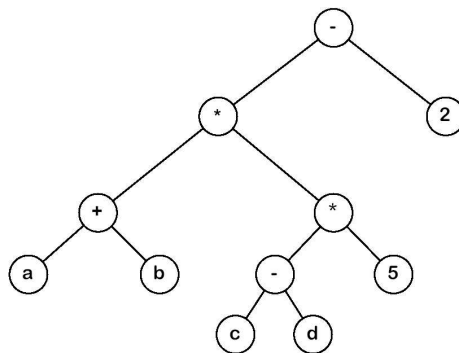
$$(a + b) * ((c - d) * 5) - 2.$$

Hinweis: Beachten Sie die übliche Vorrangregelung der Operatoren.

- Zeichnen Sie den zugehörigen Kantorowitsch-Baum. (3 Punkte)
- Bestimmen Sie die Postfix-Form des Ausdrucks. (2 Punkte)

Lösung:

a)



b) Postfix-Form: $ab+cd-5**2-$

2.2 Boolesche Algebra (5 Punkte)

Wandeln Sie den Ausdruck

$$\left(((a \wedge b) \vee (\neg a \wedge \neg b)) \wedge \neg(a \vee c) \right) \vee (b \wedge c)$$

unter Verwendung der Gesetze der Booleschen Algebra in konjunktive Normalform (KNF) um.

Lösung:

$$\begin{aligned} & \left(((a \wedge b) \vee (\neg a \wedge \neg b)) \wedge \neg(a \vee c) \right) \vee (b \wedge c) \\ &= \left(((a \wedge b) \vee (\neg a \wedge \neg b)) \wedge \neg a \wedge \neg c \right) \vee (b \wedge c) \\ &= \left((a \vee (\neg a \wedge \neg b)) \wedge (b \vee (\neg a \wedge \neg b)) \wedge \neg a \wedge \neg c \right) \vee (b \wedge c) \\ &= \left(((a \vee \neg a) \wedge (a \vee \neg b)) \wedge ((b \vee \neg a) \wedge (b \vee \neg b)) \wedge \neg a \wedge \neg c \right) \vee (b \wedge c) \\ &= \left((a \vee \neg b) \wedge (b \vee \neg a) \wedge \neg a \wedge \neg c \right) \vee (b \wedge c) \\ &= \left((a \vee \neg b) \wedge \neg a \wedge \neg c \right) \vee (b \wedge c) \\ &= \left((a \wedge \neg a \wedge \neg c) \vee (\neg b \wedge \neg a \wedge \neg c) \right) \vee (b \wedge c) \\ &= \left(\neg b \wedge \neg a \wedge \neg c \right) \vee (b \wedge c) \\ &= (b \vee (\neg b \wedge \neg a \wedge \neg c)) \wedge (c \vee (\neg b \wedge \neg a \wedge \neg c)) \\ &= (b \vee \neg b) \wedge (b \vee \neg a) \wedge (b \vee \neg c) \wedge (c \vee \neg b) \wedge (c \vee \neg a) \wedge (c \vee \neg c) \\ &= (b \vee \neg a) \wedge (b \vee \neg c) \wedge (c \vee \neg b) \wedge (c \vee \neg a) \\ &= (\neg a \vee b \vee c) \wedge (\neg a \vee b \vee \neg c) \wedge (a \vee b \vee \neg c) \wedge (\neg a \vee b \vee \neg c) \wedge (a \vee \neg b \vee c) \wedge (\neg a \vee \neg b \vee c) \wedge (\neg a \vee \neg b \vee c) \\ &= (\neg a \vee b \vee c) \wedge (\neg a \vee b \vee \neg c) \wedge (a \vee b \vee \neg c) \wedge (a \vee \neg b \vee c) \wedge (\neg a \vee \neg b \vee c) \end{aligned}$$

3. Relationen (12 Punkte)

3.1 Halbgruppen & Monoide (4 Punkte)

Überprüfen Sie, ob es sich bei den folgenden Strukturen um Halbgruppen, Monoide oder keines von beidem handelt. Beweisen Sie Ihre Aussage.

- Die ganzen Zahlen \mathbb{Z} mit der Verknüpfung $\star: a \star b := a + 3b$. (2 Punkte)
- Die Zahlen vom Typ `int` in Java mit dem Plus-Operator `+`. (2 Punkte)

Lösung:

- \mathbb{Z} mit der Verknüpfung \star ist weder Halbgruppe noch Monoid, da das Assoziativgesetz verletzt ist:

$$\begin{aligned}(a \star b) \star c &= (a + 3b) \star c = (a + 3b) + 3c = a + 3b + 3c \\ a \star (b \star c) &= a \star (b + 3c) = a + 3 \cdot (b + 3c) = a + 3b + 9c\end{aligned}$$

$$\Rightarrow (a \star b) \star c \neq a \star (b \star c).$$

- Die Zahlen vom Typ `int` in Java mit dem Plus-Operator `+` sind ein Monoid.

- Das Assoziativgesetz ist erfüllt, da für alle `a`, `b`, `c` vom Typ `int` gilt:

$$a + (b + c) = (a + b) + c.$$

- Neutrales Element ist die 0, denn: $0 + a = a = a + 0$
- Die Menge aller Werte vom Typ `int` ist abgeschlossen, insbesondere gilt:

$$\begin{aligned}(2^{31} - 1) + 1 &= 2^{31} \in \text{int} \\ \text{und } (-2^{31} + 1) - 1 &= -2^{31} \in \text{int}\end{aligned}$$

3.2 Graphen (8 Punkte)

Gegeben sei folgender Graph $G = (E, K)$ mit $E := \{1, 2, 3, 4, 5\}$ und

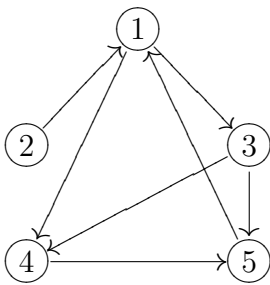
K : 1 : [3, 4]
 2 : [1]
 3 : [4, 5]
 4 : [5]
 5 : [1]

a) Wie nennt man diese Darstellungsform? (1 Punkt)

Lösung:
 Adjazenzliste

b) Zeichnen Sie den zugehörigen Graphen. (1 Punkt)

Lösung:



c) Führen Sie den Floyd-Warshall-Algorithmus zur Berechnung der reflexiven transitiven Hülle für die durch den Graphen gegebene Relation durch. Tragen Sie dazu die Ursprungsrelation in das erste angegebene Feld ein und führen Sie dort den ersten Schritt des Algorithmus aus. Führen Sie dann den Algorithmus der Reihe nach für die Spalten 1,2,3,4 und 5 durch und tragen Sie die jeweiligen Zwischenergebnisse in die dafür vorgesehenen Felder ein. Markieren Sie jeweils die hinzugekommene(n) Kante(n) durch Einkreisen! (6 Punkte)

	1	2	3	4	5
1	1		1	1	
2	1	1			
3			1	1	1
4				1	1
5	1				1

~>

	1	2	3	4	5
1	1		1	1	
2	1	1	1	1	
3			1	1	1
4				1	1
5	1		1	1	1

~>

	1	2	3	4	5
1	1		1	1	
2	1	1	1	1	
3			1	1	1
4				1	1
5	1		1	1	1

~>

	1	2	3	4	5
1	1		1	1	1
2	1	1	1	1	1
3			1	1	1
4				1	1
5	1		1	1	1

~>

	1	2	3	4	5
1	1		1	1	1
2	1	1	1	1	1
3	1		1	1	1
4	1		1	1	1
5	1		1	1	1

~>

	1	2	3	4	5
1	1		1	1	1
2	1	1	1	1	1
3	1		1	1	1
4	1		1	1	1
5	1		1	1	1

4. Java (22 Punkte)

4.1 Suche in einer sortierten Liste (10 Punkte)

Schreiben Sie eine Methode

```
static boolean search(int[] list, int n)
```

die in einer *aufsteigend sortierten* Liste (hier als Array `list` übergeben) aus positiven ganzen Zahlen das Element `n` sucht. Falls `n` gefunden wird, soll `true` zurückgegeben werden; falls nicht, `false`.

Die Methode soll eine Theoretische Effizienz von $O(\log n)$ haben.

Hinweis: `list.length` liefert die Länge der Liste bzw. des Arrays `list`.

Implementierungen mit einer Theoretischen Effizienz von $O(n)$ und höher werden mit 0 Punkten bewertet.

Rekursive Lösung:

```
static boolean search(int[] list, int n) {
    int l = 0;
    int r = list.length - 1;
    return searchRek(list, n, l, r);
}

static boolean searchRek(int[] list, int n, int l, int r) {
    int p = (r - l)/2 + 1;           // pivot index
    if (r - l <= 0)                 // List empty?
        return false;              // Then n is not in the list
    else if (n == list[p])          // n found?
        return true;               // yepeeee!
    else if (n < list[p])           // can n be in the left half of list?
        return searchRek(list, n, l, p - 1); // search left half (p - 1 because n != list[p])
    else                             // n can only be in the right half of list
        return searchRek(list, n, p, r + 1); // search right half (p + 1 because n != list[p])
}
}
```

Iterative Lösung:

```
static boolean search(int[] list, int n) {
    int l = 0;                      // left border
    int r = list.length - 1;        // right border
    int p;                           // This will be our pivot
    boolean found = false;           // this will be true if n is in the list
    while (r - l > 0 && !found) { // Loop while elements left and n is not found yet
        p = (r - l)/2 + 1;           // our pivot index
        if (n == list[p])            // n found?
            found = true;            // yepeeee!
        else if (n < list[p])         // can n be in the left half of list?
            r = p - 1;                // search the left half (p - 1 because n != list[p])
        else                          // n can only be in the right half of list...
            l = p + 1;                // search the right half (p + 1 because n != list[p])
    }
    return found;                    // "found" knows whether "n inside" or not...
}
```

4.2 Quelltext-Analyse (12 Punkte)

Gegeben sei folgendes Java-Programm:

```
class Program1{
    public static void main(String args[]) {
        int a1 = 0, a2 = 0, a3 = 0, a4 = 0, a5 = 0, a6 = 0;
        int number;
        for (int i = 0; i < 600; i++) {
            number = (i % 6) + 1;          /* 1 */
            switch(number) {
                case 1 : a1++;             /* 2 */
                case 2 : a2++;             /* 3 */
                case 3 : a3++;             /* 4 */
                case 4 : a4++;             /* 5 */
                case 5 : a5++;             /* 6 */
                case 6 : a6++;             /* 7 */
            }
        }
        int sum = a1 + a2 + a3 + a4 + a5 + a6;
        Out.println("Variable 1 : " + a1);
        Out.println("Variable 2 : " + a2);
        Out.println("Variable 3 : " + a3);
        Out.println("Variable 4 : " + a4);
        Out.println("Variable 5 : " + a5);
        Out.println("Variable 6 : " + a6);
        Out.println("Insgesamt : " + sum);
    }
}
```

a) Welche Ausgabe ist zu erwarten? (3,5 Punkte)

Variable 1: 100

Variable 2: 200

Variable 3: 300

Variable 4: 400

Variable 5: 500

Variable 6: 600

Insgesamt : 2100

b) Begründen Sie Ihre Aussage. (4 Punkte)

Lösung:

Die `case`-Anweisung im `switch`-Block setzt nur Sprungmarken (labels). Trifft der Fall einer `case`-Anweisung zu, so springt der Programmzeiger an die markierte Stelle und ignoriert weitere Sprungmarken.

Trifft nun `case 1` zu, ist also `number == 1`, so springt das Programm an die angegebene Stelle und inkrementiert `a1`. Das Programm läuft aber weiter und bricht den `switch`-Block nicht ab. Demnach werden nun noch `a2`, `a3`, `a4`, `a5` und `a6` erhöht.

Ist `number == 4`, so werden nur `a4`, `a5` und `a6` erhöht, da das Programm zum Label `case 4` springt. `a1`, `a2` und `a3` werden übergangen. `a6` wird demnach bei jedem Schleifen-Durchgang erhöht.

Die Variable `number` nimmt während der `for`-Schleife periodisch 1, 2, 3, 4, 5 und 6 an. Sie ist also genau 100 mal 1, 100 mal 2, usw.

Um an einer Stelle den `switch`-Block abubrechen, benötigt man die Anweisung `break` (siehe Aufgabe c).

c) Wie müsste das Programm verändert bzw. erweitert werden, damit es eine Simulation eines 6-seitigen Würfels mit gleicher Verteilung ergibt? Beschränken Sie sich dabei nur auf die Zeilen, die mit einem Kommentar `/**/` markiert sind! (3,5 Punkte)

`/* 1 */` Ersetzen durch: `number = (int)(Math.random()*6) + 1;`

`/* 2 */` Erweitern mit: `break;`

`/* 3 */` Erweitern mit: `break;`

`/* 4 */` Erweitern mit: `break;`

`/* 5 */` Erweitern mit: `break;`

`/* 6 */` Erweitern mit: `break;`

`/* 7 */` Erweitern mit: `break;` oder Nicht verändern

d) Welche Funktion erfüllt `default` in einem `switch`-Block? (1 Punkt)

Lösung:

Oft ist es nicht nötig oder möglich alle Fälle bei einem `switch` durch `case` abzudecken. Trifft nun keiner der durch `case` abgedeckten Fälle ein, so wird zum Label `default` gesprungen. Ist dieses nicht gesetzt, wird zum Ende des `switch`-Blocks gesprungen. Der Block wird nicht ausgeführt.

5. UML (8 Punkte)

Homer J. Simpson hat sich entschlossen zu studieren. Da er so vergesslich ist, musste ihm Marge ein Aktivitätsdiagramm seines Tages anfertigen, damit er sich auch an der Universität zurechtfindet.

Sein Tagesablauf sieht folgendermassen aus: Zunächst geht er zum Frühstückstisch und isst solange bis er nicht mehr hungrig ist. Danach hat er mehrere Möglichkeiten zur Universität von Springfield zu gelangen, die er allerdings vom Wetter abhängig macht: Wenn es nicht regnet, geht er zu Fuß; wenn es regnet, aber nicht glatt ist, fährt er mit dem Auto; wenn es regnet und glatt ist, fährt er mit dem Bus.

Wenn er zu Fuß geht, macht er zwischendurch einen Zwischenstop bei Moe's und kippt 2 Bierchen, macht sich dann direkt auf den Weg zur Universität; Wenn er mit dem Bus oder dem Auto fährt, fährt er direkt zur Universität.

An der Universität hört er Montags, Mittwochs und Donnerstags den ganzen Tag „Atomkraftwerksicherheit für Fortgeschrittene“ (abgekürzt: AKS II). Dienstags hat er Tutorium in „Effizient ohne Überarbeitung“ (abgekürzt: EÜ), Freitags hätte er „Körperliche Fitness für Anfänger“ (abgekürzt: KF I), wo er aber nur hinget, wenn eine Prüfung stattfindet. Nach den Veranstaltungen fährt er nach Hause.

Da Marge leider im Moment auf Schulung ist und Homer eine ganze Flasche Ketchup über sein altes UML-Aktivitätsdiagramm geschüttet hat, müssen Sie nun ein neues erstellen.

Hinweis: Sie können davon ausgehen, dass die Klassen `Homer`, `Terminkalender` & `Wetter` bereits existieren und mit den benötigten Methoden ausgestattet sind.

Lösung:

