

Klausur Informatik II

8.10.2001

Vorname: _____

Name: _____

Matrikelnummer: _____

Aufgabe	0	1	2	3	4	5	6	7	8	9
Punkte										
von	1	5	7	6	7	5	6	8	7	8

Summe

Zum Bestehen der Klausur
sind 20 Punkte hinreichend.

Note:

Die Klausur ist komplett und geheftet abzugeben.
Für die Vollständigkeit nicht gehefteter Klausuren
wird keine Verantwortung übernommen.

Sie dürfen auch die Rückseiten der Aufgabenblätter beschreiben.
Verwenden Sie kein eigenes Papier. Konzeptpapier gibt es von der Aufsicht.

Zur Klausur sind keine Hilfsmittel zugelassen.

Aufgabe 0

(1 Punkt)

Schreiben Sie auf jedes Blatt auch auf das Konzeptpapier Ihren Namen und Ihre Matrikelnummer.

Aufgabe 1 Objektorientierung

(5 Punkte)

1+4

Gegeben sei folgendes JAVA-Programm

(die erste Spalte gehört nicht zum Programm und dient nur der Numerierung der Zeilen):

```

01  class K1
02  {
03      int i; int j;
04      public int f() { return i; }
05  }
06
07  class K2 { class K3 { int i; } int j; }
08
09  class K4 extends K1
10  {
11      public int f(int i) { return i+j; }
12
13      public K4 (int i) { this.i = i; j = 2*i; }
14
15      public static void main (String args[])
16      {
17          K4 k = new K4(2);
18          System.out.println(k.f());
19          System.out.println(k.f(3));
20      }
21  }

```

- a) Welche beiden Zahlen druckt das Programm nach Aufruf von "java K4" aus?

2 7

- b) Kreuzen Sie bei den folgenden Aussagen die jeweils korrekte Vervollständigung an.
(Falsche Kreuze geben Punktabzug.)

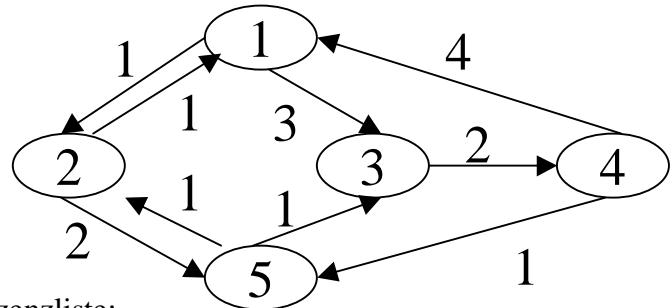
K3 ist eine Unterklasse von	<input type="checkbox"/> K1	<input type="checkbox"/> K2	<input type="checkbox"/> K4	<input checked="" type="checkbox"/> nichts davon	
K4 ist eine Unterklasse von	<input checked="" type="checkbox"/> K1	<input type="checkbox"/> K2	<input type="checkbox"/> K3	<input type="checkbox"/> nichts davon	
Der Bezeichner i in Zeile 4 ist	<input type="checkbox"/> Klassenvariable	<input checked="" type="checkbox"/> Instanzvariable	<input type="checkbox"/> Parameter		
Der Bezeichner K4 in Zeile 13 ist	<input type="checkbox"/> Klasse	<input type="checkbox"/> Parameter	<input checked="" type="checkbox"/> Konstruktor	<input type="checkbox"/> Attribut	
Der Bezeichner K3 in Zeile 7 ist	<input checked="" type="checkbox"/> Klasse	<input type="checkbox"/> Interface	<input type="checkbox"/> statisch	<input type="checkbox"/> nichts davon	
Eine abstrakte Klasse ist	<input type="checkbox"/> K1	<input type="checkbox"/> K2	<input type="checkbox"/> K3	<input type="checkbox"/> K4	<input checked="" type="checkbox"/> keine davon
Die Klasse K2 hat keine(n)	<input checked="" type="checkbox"/> Unterklasse	<input type="checkbox"/> Konstruktor	<input type="checkbox"/> Instanzvariablen		
Klassenvariablen sind	<input type="checkbox"/> j in Zeile 7	<input type="checkbox"/> k in Zeile 17	<input checked="" type="checkbox"/> keine davon		

Aufgabe 2 Graphenalgorithmen

(7 Punkte)

2+2+3

Gegeben sei der folgende gerichtete Graph mit gewichteten Kanten:



Er ist implementiert mit Hilfe der folgenden Adjazenzliste:

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

6	8	10	11	13	2	3	1	5	4	1	5	2	3	0
---	---	----	----	----	---	---	---	---	---	---	---	---	---	---

(0 markiert Listenende)

Alle Algorithmen bearbeiten die Nachfolger eines Knotens in der Reihenfolge, in der sie in der Adjazenzliste auftreten.

- a) Geben Sie die Reihenfolge der Knoten an, in denen der Tiefensuchenalgorithmus, angefangen bei Knoten 1, die Knoten besucht.

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

- b) Geben Sie die Reihenfolge der Knoten an, in denen der Breitensuchenalgorithmus, angefangen bei Knoten 1, die Knoten besucht.

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

- c) Geben Sie in der untenstehenden Tabelle alle, zwischenzeitlich als kürzeste angenommenen Pfade (Knotenfolge) von Knoten 4 zum Knoten 3 mit ihrer Länge an, in der Reihenfolge, in der sie der Ford-Fulkerson-Algorithmus findet.

Pfad	Länge
4 1 2 5 3	8
4 1 3	7
4 5 2 1 3	6
4 5 3	2

Aufgabe 3 Aufwandsschätzung

(6 Punkte)
3+3

- a.) Zeigen oder widerlegen Sie die Aussage: $f(n) = O(n) \Rightarrow \frac{n^2 + f(n)}{n} = O(n)$

$$f(n) = O(n) \Rightarrow \exists N_1, k_1 : \forall n > N_1 : f(n) \leq k_1 \cdot n$$

also

$$\frac{n^2 + f(n)}{n} \leq \frac{n^2 + k_1 \cdot n}{n} = n + k_1$$

wir wählen jetzt $N_2 = k_1$ und $k_2 = 2$ dann ist
 $\forall n \geq N_2 : n + k_1 < k_2 \cdot n$ also $n + k_1 = O(n)$

oder

$$\forall k > 1, n > 1 : \frac{n^2 + f(n)}{n} \leq \frac{kn^2 + kn^2}{n} \leq 2kn = O(n)$$

- b.) Zeigen oder widerlegen Sie die Aussage:
 Wenn $f = O(r(n))$ und $g = O(s(n))$, dann ist $f \cdot g = O((r \cdot s)(n))$.

laut Voraussetzung: $\exists k_1, N_1 : \forall n > N_1 : f(n) < k_1 r(n)$
 und: $\exists k_2, N_2 : \forall n > N_2 : g(n) < k_2 s(n)$

also ist $f(n) \cdot g(n) < k_1 r(n) \cdot k_2 s(n)$

Wir wählen nun $k_3 = k_1 \cdot k_2$ und $N_3 = \max(N_1, N_2)$,

dann ist $\forall n > N_3 : f(n) \cdot g(n) < k_3 (r(n) \cdot s(n))$
 somit $(f \cdot g)(n) = O(r \cdot s)(n)$

Aufgabe 4 Verifikation

(7 Punkte)
1+2+4

Berechnen Sie mit Hilfe des Weakest-Precondition-Kalküls (wp-Kalküls) die folgenden Ausdrücke:
Überspringen Sie dabei keine Kalkül-Regeln und vereinfachen Sie die Ergebnisterme.

a) $\text{wp}(\mathbf{x=y+1;}, \mathbf{x>y})$

$$= \mathbf{y+1 > y} = \mathbf{true}$$

b) $\text{wp}(\mathbf{x=a;}, \mathbf{false})$

$$= \mathbf{false}$$

c) $\text{wp}(\mathbf{x=a; if (x>b) x=b;}, \mathbf{x=b})$

$$= \mathbf{wp(x=a, wp (if(x>b) x=b , x=b))}$$

$$= \mathbf{wp(x=a, ((x>b) \wedge wp(x=b, x=b)) \vee (\neg (x>b) \wedge x=b))}$$

$$= \mathbf{wp(x=a, (x>b \wedge true) \vee (x\leq b \wedge x=b))}$$

$$= \mathbf{wp(x=a, x > b \vee x = b)}$$

$$= \mathbf{wp(x=a, x \geq b) = a \geq b}$$

Aufgabe 5 Suchen/Sortieren

(5 Punkte)
1+3+1

Gegeben sei der folgende Sortieralgorithmus für natürliche Zahlen. Für die n zu sortierenden Zahlen $z[i]$ gilt $0 < z[i] \leq m$. Die Felder $k[]$ und $z[]$ seien ausreichend groß alloziert.

```
void sort ()
{
    for (int a=1; a<=m; a++)    k[a] = 0;

    for (int a=1; a<=n; a++)    k[z[a]]++;

    int x=1;

    for (int a=1; a<=m; a++)
        while (k[a]>0) z[x++] = a;
}
```

- a) Geben Sie die Laufzeit des Algorithmus in Abhängigkeit von n und m im O-Kalkül an.

$O(n+m)$

- b) Für welche Werte von m und n würden Sie aus Laufzeitgründen diesen Algorithmus dem Heapsort-Verfahren (Sortieren mit Halde) vorziehen? Begründen Sie Ihre Antwort.

**für $n+m = O(n \log n)$ da Heapsort in $O(n \log n)$
also für $m = O(n \log n)$**

**Bemerkung: auch Rechnen mit exakten Werten
statt O-Kalkül ist okay (Konstante egal).**

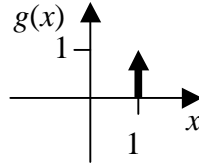
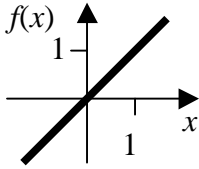
- c) Kreuzen Sie an, um was für ein Sortierverfahren es sich bei dem obigen Algorithmus handelt.

- Bubble Sort (Sortieren durch Austauschen, Blasensortieren)
- Quick Sort
- Merge Sort (Sortieren durch Mischen)
- Bucket Sort
- Insertion Sort (Sortieren durch Einordnen)

Aufgabe 6 Signale

(6 Punkte)
3+3

- a) Gegeben sind die Funktionen $f(x) = x$ und $g(x) = \delta_1(x)$:



berechnen Sie die Faltung $(f * g)(x)$ von $f(x)$ und $g(x)$.

$$\begin{aligned}
 (f * g)(x) &= \int_{-\infty}^{+\infty} f(x-t)g(t) dt \\
 &= \int_{-\infty}^{+\infty} (x-t)\delta_0(t-1) dt \\
 &\stackrel{s:=t-1}{=} \int_{-\infty}^{+\infty} (x-s-1)\delta_0(s) ds \\
 &\stackrel{\int f(x)\delta(t) = f(0)}{=} x - 1
 \end{aligned}$$

- b) Reduzieren Sie die Multiplikation zweier Polynome $P = \sum_{i=0}^n p_i x^i$ und $Q = \sum_{i=0}^n q_i x^i$ auf die diskrete Faltung $c = a * b$ zweier Folgen a und b .

setze $a[i] = p_i$ und $b[i] = q_i$

wenn nun $R = PQ = \sum_{i=0}^{2n} r_i$

dann ist $r_i = c[i]$

Aufgabe 7 Komplexität

(8 Punkte)

1+1+1+1+1+1+1

Kreuzen Sie im folgenden an, ob die Aussagen wahr (so: f) oder falsch (so: w) sind. Die Schreibweise $A \leq_p B$ bezeichne den Sachverhalt "A ist polynomial auf B reduzierbar".

(Falsche Antworten geben Punktabzug, nicht bearbeitete Teilaufgaben geben 0 Punkte.)

a) $Y \leq_p X$ und $X \in NP \Rightarrow Y \in NP$.

 w f

b) U ist NP-vollständig \Rightarrow U ist NP-hart.

 w f

c) Q ist NP-hart $\Rightarrow Q \leq_p 3SAT$

 w f

d) $3SAT \in P \Rightarrow P = NP$

 w f

e) $E \in P$ und $C \leq_p D$ und $D \leq_p E \Rightarrow C \in P$

 w f

f) Wenn der Algorithmus F Formeln aus der konjunktiven Normalform in die disjunktive wandelt, dann gilt: $F \in P \Rightarrow P = NP$.

 w f

g) $G \in NP \Rightarrow$ es gibt einen Las-Vegas-Algorithmus, der G in polynomialer Zeit löst.

 w f

h) H_1 ist P-hart, H_2 ist P-hart und $H_1 \leq_p H_2$, dann ist $H_2 \leq_p H_1$

 w f

Aufgabe 8 Parallelität

(7 Punkte)
1+1+1+4

In einer Liste mit n Objekten sei jedes Objekt entweder rot oder blau gefärbt. Im folgenden ist ein paralleler Algorithmus für eine EREW-Maschine gegeben, der auf der Liste L eine Verkettungsstruktur für die roten Elemente anlegt. Der Algorithmus benötigt n Prozessoren, wobei jeder Prozessor einem Listenelement zugeordnet ist. Der Anfang der Liste wird mit **head** und das Ende mit **tail** bezeichnet. Das dem i -ten Element folgende wird in **next[i]** gespeichert. Der Anfang der Verkettungsstruktur mit den roten Elementen wird mit **red_head** bezeichnet. Zum Aufbauen der Verkettungsstruktur dienen die Variablen **next_red** und **next_red_found**.

```

for i=0..n in parallel { // Initialisierung
    next_red[i] = next[i];
    next_red_found[i] = false;
}
for i=0..n in parallel { // Anlegen der Verkettungsstruktur
    while ((next_red_found[i]==false) && (next_red[i]!=tail)) {
        if (color[next_red[i]]==red) next_red_found[i] = true;
        else next_red[i] = next_red[next_red[i]];
    }
}
if (color[head]==red) red_head = head;
else red_head = next_red[head]

```

- a) Geben Sie die Laufzeit des Algorithmus in Abhängigkeit von n im O-Kalkül an.

$$T(n, n) = O(\log n)$$

- b) Geben Sie den Speedup des Algorithmus an.

$$S(n, n) = T(n, 1) / T(n, n) = O(n / \log n)$$

- c) Geben Sie die Effizienz des Algorithmus an.

$$E(n, n) = T(n, 1) / nT(n, n) = O(n / n \log n) = O(1 / \log n)$$

- d) Wenden Sie den Satz von Brent an und geben Sie an, ob und warum der Algorithmus (nicht) modifizierbar ist, so daß er bei gleichem zeitlichen Aufwand eine höhere Effizienz hat?

Anzahl Schritte $s = O(n \log n)$, Laufzeit $t = O(\log n)$
 Nach Brent ex. Algorithmus mit $T(n, s/t) = O(\log n)$
 $= T(n, n \log n / \log n) = T(n, n)$ d.h. genauso viele
 Prozessoren, also kein Effizienzgewinn.

Aufgabe 9 Assembler

(8 Punkte)

4+2+2

Betrachten Sie das folgende Programmfragment in 68000er Assembler:

START	MOVE.L	#0,D1	Register D1 leeren
	MOVE.W	3,D0	D0 initialisieren
LOOP1	MOVE.W	D0,-(SP)	D0 auf Stapel legen
	JSR	UNTER	Unterprogramm aufrufen
	MOVE.W	(SP)+,D0	Stapel aufräumen
	BNE	LOOP1	verzweige wenn D0-1 != 0
	MOVE.L	#1,D0	Produkt mit 1 initialisieren
LOOP2	MULU	D1,D0	D0 = D0 * D1
	ADDI.W	#-1,D1	D1 = D1 + (-1)
	BNE	LOOP2	Schleife drehen
	JMP	FIN	Schluß jetzt
UNTER	ADD.W	4(SP),D1	D1 = D1 + Übergabeparameter
	ADDI.W	#-1,4(SP)	Parameter dekrementieren
	RTS		ReTurn from Subroutine
FIN	_WRITE(" 6! = ")		String ausgeben
	_WRITELONG(D0)		Inhalt von D0 dezimal ausgeben

- a) Ergänzen Sie die grau unterlegten Felder mit Assemblercode, so daß das Programm am Schluß $6! = 720$ ausgibt.
- b) Geben Sie für den folgenden Ausdruck in Infix-Form den entsprechenden Ausdruck in Postfix-Form an, bei dem die Reihenfolge der Operanden erhalten bleibt.

$$2 + (7-b) \cdot a$$

$$2 \ 7 \ b \ - \ a \ \cdot \ +$$

- c) Geben Sie für den folgenden Ausdruck in Präfix-Form den entsprechenden Ausdruck in Infix-Form an, bei dem die Reihenfolge der Operanden erhalten bleibt.

$$\cdot - c \cdot 4 \ d \ e$$

$$(c - 4 \cdot d) \cdot e$$