Informatik II 2. August 2004

gelesen von Prof. Zitterbart

Durchfallquote:
Hauptklausur: 42 %

Zum Bestehen waren 30 von 60 Punkten erforderlich.

Umbruch: Micha Lenk, Timo Weingärtner
Herausgeber: Fachschaft Mathematik/Informatik

Für die Wiedereinführung der echten verfassten Studierendenschaft

Klausur Informatik II vom 2. August 2004 (Prof. Zitterbart)

Aufgabe 1 (10P):
Wissensaufgaben: Beantworten Sie die folgenden Fragen kurz aber präzise.

1. Gegeben seien ein Anweisungsblock S und eine Nachbedingung Q. Warum muss für jedes P, für das {P} S {Q} gilt, auch P => wp(S, Q) gelten? (2P)
2. Was bedeutet in Java das Schlüsselwort **`protected`** unmittelbar vor einer Variablen-Vereinbarung? (1P)
3. Gegeben sei folgender Teil eines Java-Programms: (2P)

```
int a=2;
int b=3;
System.out.println(--a << 2 % b++);
System.out.println(b);
```

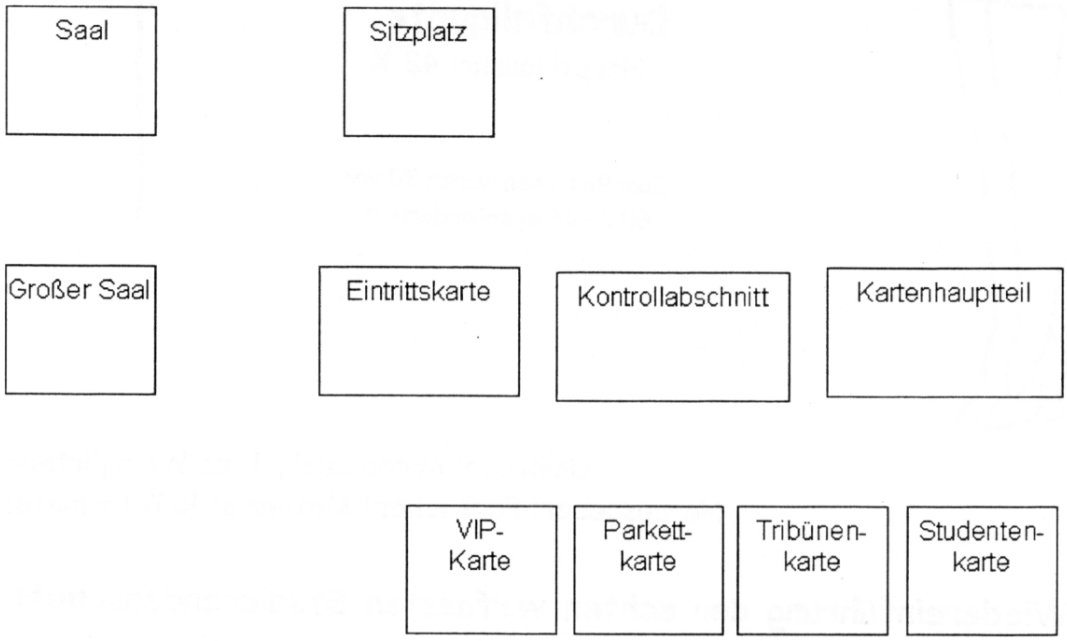
   Wie lautet die Ausgabe dieses Teil-Programms?
4. Gibt es bipartite Graphen, die keine Bäume sind? Wenn ja – warum? Wenn nein – warum nicht? (1P)
5. Geben Sie die Rechen-Komplexität von Counting-Sort möglichst genau an. (1P)
6. In der Übung wurde gezeigt, dass $\Omega(n \log n)$ eine untere Schranke für Sortierverfahren darstellt. Wieso widerspricht Counting-Sort dieser Schranke nicht? (2P)
7. Wann sind implizite Casts zwischen Typen verschiedener Wertebereiche (z.B. zwischen **`int`** und **`double`**) möglich? (1P)

Aufgabe 2 (10P):
Das Badische Staatstheater beauftragt Sie, ein neues Kartenreservierungssystem zu erstellen. Sie stellen folgende Situation fest:

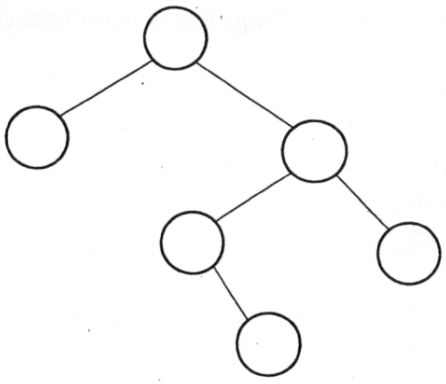
Es existieren verschiedene Säle mit jeweils unterschiedlicher Anzahl an Sitzplätzen. Der Große Saal hat 1000 Sitzplätze. Zu jedem Sitzplatz gehört genau eine Eintrittskarte. Es gibt 4 verschiedene Arten von Eintrittskarten für den Großen Saal : Studentenkarten (125 Stück), Tribünenkarten (400 Stück), Parkett-Karten (375 Stück) und VIP-Karten (100 Stück). Die Eintrittskarten bestehen jeweils aus einem Kontrollabschnitt (zum Abreißen) und einem Kartenhauptteil. Kartenhauptteil und Kontrollabschnitt verfügen jeweils über dieselbe ID. Auf jeder Eintrittskarte ist eine Werbenachricht untergebracht. Diese ist für alle Eintrittskarten gleich.

Vervollständigen Sie das unten aufgeführte Klassendiagramm.

Hinweis: Fügen Sie keine eigenen Klassen hinzu. Verwenden Sie alle angegebenen Klassen.



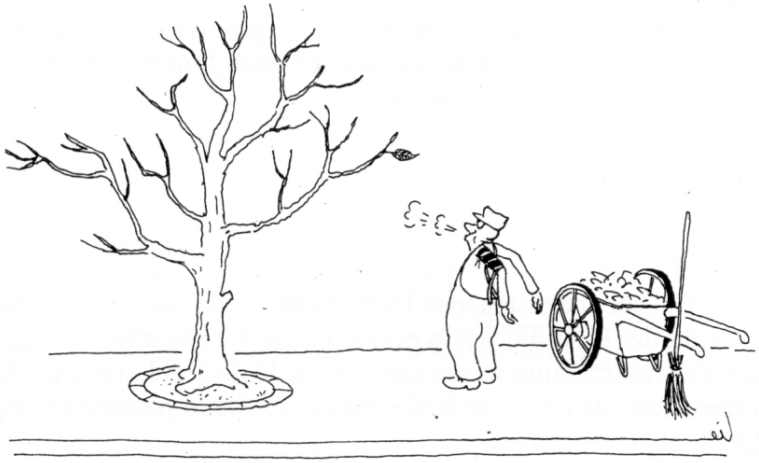
Aufgabe 3 (10P):
Gegeben sei nebenstehender Baum mit "leeren" Knoten:



a) (1P)
Fügen Sie die Ziffern 1, 2, 3, 4, 5, 6 so in die Knoten ein, dass der daraus resultierende Baum die Eigenschaft eines binären Suchbaums besitzt.

b) (3P)
Erfüllt dieser Baum die AVL-Eigenschaft?
Wenn ja – wieso? Wenn nein – zeichnen Sie den Baum nach einer geeigneten, die AVL-Eigenschaft herstellenden Rotation.

c) (6P)
Zeigen Sie, dass ein AVL-Baum der Höhe h aus *mindestens* F_{h+3}-1 Knoten besteht. Dabei steht die n-te Fibonacci-Zahl (0, 1, 1, 2, ...) für F_n.



Aufgabe 4 (10P):
a) (3P)
Gegeben sei das folgende Java-Programm:

```
class Magic {

     static int i = 5;

     public Magic (int j) {
          i = j;
          System.out.println(i);
     }

     public static void main (String argv[]) {
          System.out.println(i);
          Magic c1 = new Magic(++i);
          System.out.println(i);
          Magic c2 = new Magic(i++);
          System.out.println(i);
     }
}
```

Wie lautet seine Ausgabe?

b) 1. (3,5P) Schreiben Sie eine *rekursive* Java-Funktion **`int fak (int n)`**, die zu einer Zahl $n \geq 0$ deren Fakultät berechnet und zurückgibt.

2. (3,5P) Schreiben Sie nun eine *iterative* Java-Funktion **`int fak (int n)`**, die zu einer Zahl n dessen Fakultät berechnet und zurückgibt.

Aufgabe 5 (10P):
Das folgende Code-Fragment **S** berechnet die n-te Fibonacci-Zahl F_n.

```
S:
int m=1;
int a=1;
int b=0;
int i=2;

while (i<n) {
      b=a;
      a=m;
      m=a+b;
      i++;
}
```

Die Vorbedingung vor diesem Code-Fragment lautet P: {n≥2}, wobei n die Zahl ist, für die F_n ausgerechnet werden soll. Als Nachbedingung gilt Q:={m=F_m}.

a) (9P)
Beweisen Sie die partielle Korrektheit von **S** bzgl. Q un P mit Hilfe des WP-Kalküls.

b) (1P)
Zeigen Sie, dass die while-Schleife terminiert.

Aufgabe 6 (5P):
Gegeben seien die drei Funktionen f(n), g(n) und h(n). Nun gelte $f(n) = \Theta(g(n))$ und außerdem gelte $g(n) = \Theta(h(n))$. Gilt dann automatisch auch $f(n) = \Theta(h(n))$?
Beweisen Sie Ihre Behauptung.

Aufgabe 7 (5P):
Gegeben sei nebenstehende Hash-Tabelle, die durch Einfügen der Elemente 1, 3, 4, 6 mit der Hash-Funktion h(x) = x mod 4 entstanden ist. Zur Kollisionsauflösung werde eine zweite Hash-Funktion g(p)=(p^2 + 1) mod 11 verwendet, die bei einer Kollision an der Stelle p versucht, den Schlüssel x an der Stelle g(p) einzufügen.

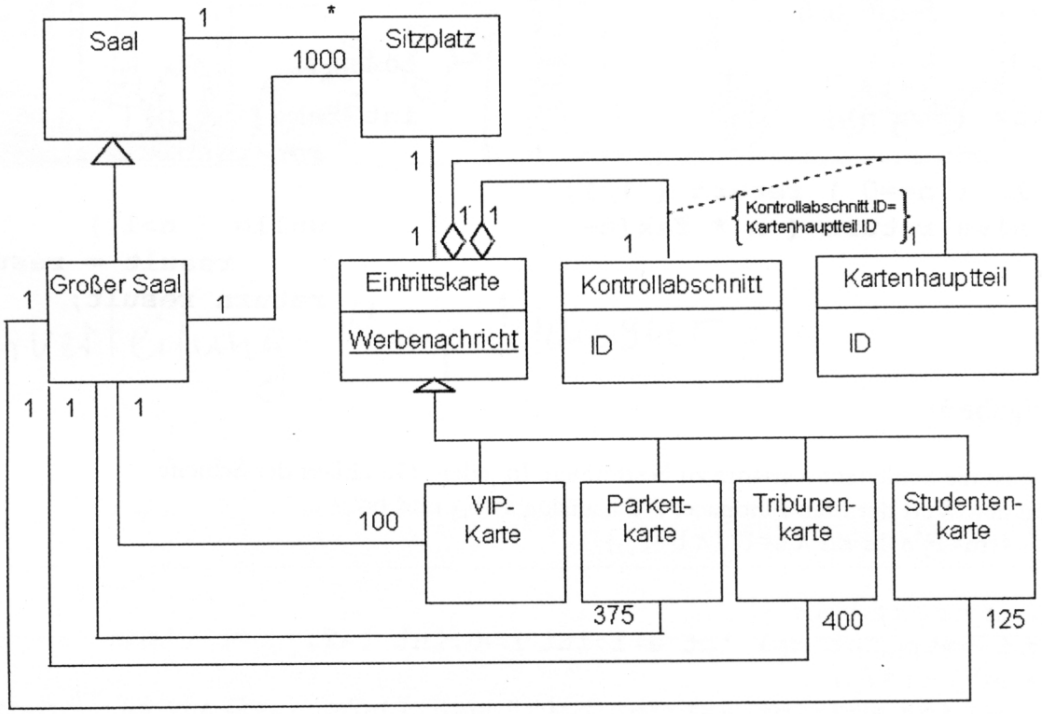
Fügen Sie mit dieser Kollisions-Strategie zusätzlich die Schlüssel 8, 9, 7 in dieser Reihenfolge in die Tabelle ein.

Schlüssel	Zelle
4	0
1	1
6	2
3	3
	4
	5
	6
	7
	8
	9
	10

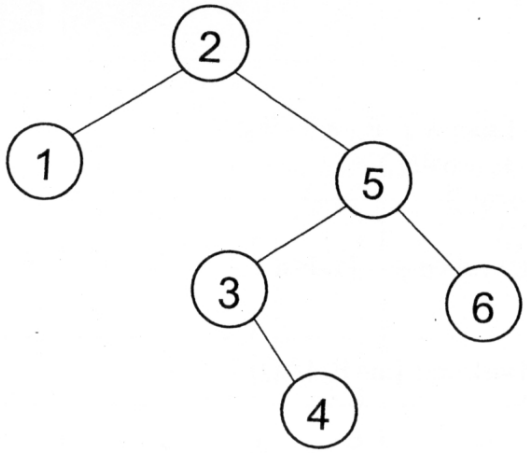
Lösung zu Aufgabe 1:

1. Antwort: wp(S, Q) ist die schwächste Vorbedingung, die erfüllt sein muss, damit nach S die Nachbedingung Q erreicht wird. Jedes P muss demnach wp(S, Q) erfüllen.
2. Antwort: Dieses Schlüsselwort bedeutet, dass die Variable nur sichtbar ist innerhalb der eigenen Klasse, allen Klassen der eigenen **`package`**, sowie allen vererbten Klassen.
3. Antwort: 4 und 4
4. Antwort: Ja, z.B. bipartite Graphen, die einen Zyklus enthalten.
5. Antwort: Aufwand von $\Theta(n)$
6. Antwort: Die Schranke gilt für O(1) Speicher. Counting-Sort benötigt in Abhängigkeit der Größe der zu sortierenden Zahlen jedoch zusätzlichen Speicher.
7. Antwort: Dies gilt nur bei *Widening*.

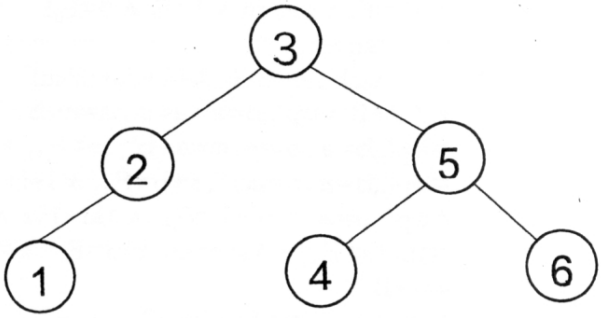
Lösung zu Aufgabe 2:



Lösung zu Aufgabe 3a:



Lösung zu Aufgabe 3b:



Lösung zu Aufgabe 3c:
Beweis durch vollständige Induktion
Bezeichne Anzahl der Knoten im Baum der Höhe h mit n_h.

Induktionsanfang:
h => Baum besteht nur aus einem Knoten: $F_3-1 = 1 = n_h$, ok

Induktionsschluss:
Die Induktionsvoraussetzung sei für alle h' mit $1 \leq h' \leq h$ bis zu einem h bewiesen, d.h. ein Baum der Höhe h' habe mindestens $F_{h'+3}-1$ Knoten: $n_{h'} \geq F_{h'+3}-1$

Jetzt zu zeigen: h+1
Für jeden AVL-Baum der Höhe n_{h+1} gilt: $n_{h+1} \geq h+n_{h-1}+1$,
da der Baum aus Wurzelknoten + linker Teilbaum + rechter Teilbaum besteht.
Der Balance-Faktor des AVL-Baums erlaubt dabei einen Höhenunterschied von "1" zwischen linkem und rechtem Teilbaum.

nach I.V. => $n_{h+1} \geq (F_{h+3}-1)+(F_{h+2}-1)+1 = F_{h+4}-1$ q.e.d.

Lösung zu Aufgabe 4:

a) Lösung: 5, 6, 6, 6, 6

b) Lösung 1:

```
int fak (int n){

     if ( n==0 ) return ( 1 );
     else return ( n * fak(n-
1) );

}
```

Lösung 2:

```
int fak (int n){
     int result = n;

     while ( n>1 )
          result = result*(--n);
     return result;
}
```

Lösung zu Aufgabe 5:

a) Zunächst ist die Schleifeninvariante zu bestimmen. In jedem Durchlauf der Schleife enthält m den Wert der i-ten Fibonacci-Zahl, analog $a=F_{i-1}$ und $b=F_{i-2}$.
$I:=\{m=F_i \wedge 1 \leq i \leq n \wedge a=F_{i-1} \wedge b=F_{i-2}\}$

1) I gilt vor der Schleife
z.Z. P⇒wp(„**int m=1;int a=1;int b=0;int i=2;**“, $m=F_i \wedge 1 \leq i \leq n \wedge a=F_{i-1} \wedge b=F_{i-2}$)
= wp(„**int m=1;int a=1;int b=0;**“, $m=F_2 \wedge 1 \leq 2 \leq n \wedge a=F_1 \wedge b=F_0$)
= wp(„**int m=1;int a=1;**“, $m=F_2 \wedge 1 \leq 2 \leq n \wedge a=F_1 \wedge 0=F_0$)
= wp(„**int m=1;**“, $m=F_2 \wedge 1 \leq 2 \leq n \wedge 1=F_1 \wedge 0=F_0$)
= $\{1=F_2 \wedge 1 \leq 2 \leq n \wedge 1=F_1 \wedge 0=F_0\}$
= $1 \leq 2 \leq n$ ok
2) I gilt nach einem Schleifendurchlauf
z.Z. $I \wedge B \Rightarrow$wp(„**b=a;a=m;m=a+b;i++**“, $m=F_i \wedge 1 \leq i \leq n \wedge a=F_{i-1} \wedge b=F_{i-2}$)
=wp(„**b=a;a=m;m=a+b**“, $m=F_{i+1} \wedge 1 \leq i+1 \leq n \wedge a=F_i \wedge b=F_{i-1}$)
=wp(„**b=a;a=m;**“, $a+b=F_{i+1} \wedge 1 \leq i+1 \leq n \wedge a=F_i \wedge b=F_{i-1}$)
=wp(„**b=a;**“, $m+b=F_{i+1} \wedge 1 \leq i+1 \leq n \wedge m=F_i \wedge b=F_{i-1}$)
=$\{m+a=F_{i+1} \wedge 1 \leq i+1 \leq n \wedge m=F_i \wedge a=F_{i-1}\}$ = $\{TRUE \wedge 1 \leq i < n\}$ = $\{1 \leq i < n\}$
$\Leftarrow I \wedge B$
3) Es muss gelten: $I \wedge \neg B \Rightarrow Q$
$I \wedge \neg B = \{ m=F_i \wedge 1 \leq i \leq n \wedge a=F_{i-1} \wedge b=F_{i-2} \wedge i \geq n\} \Rightarrow \{i=n\}$ und $\{m=F_n\}=Q$

b) Antwort: Die while-Schleife terminiert, da i streng monoton wächst und durch n begrenzt ist.

Lösung zu Aufgabe 6:

Ja, auch hier gilt die Transitivität!

Aus $f(n) = \Theta(g(n))$ folgt: $\exists c_1, c_2 > 0, \exists n_0 \in N, \forall n \geq n_0 : 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$,
aus $g(n) = \Theta(h(n))$ folgt: $\exists c_3, c_4 > 0, \exists n_0 \in N, \forall n \geq n_0 : 0 \leq c_3 h(n) \leq g(n) \leq c_4 h(n)$
$\Rightarrow 0 \leq c_3 c_2 h(n) \leq c_2 g(n) \leq c_4 c_2 h(n)$
$\Rightarrow f(n) \leq c_2 g(n) \leq c_4 c_2 h(n)$
und umgekehrt
$0 \leq c_1 c_3 h(n) \leq c_1 g(n) \leq c_1 c_4 h(n)$
$\Rightarrow 0 \leq c_1 c_3 h(n) \leq c_1 g(n) \leq f(n)$
$\Rightarrow \exists c'_1, c'_2 > 0, \exists n_0 \in N, \forall n \geq n_0 : 0 \leq c'_1 h(n) \leq f(n) \leq c'_2 h(n)$, mit $c'_1 = c_1 c_3, c'_2 = c_2 c_4$
$\Rightarrow f(n) = \Theta(h(n))$

Lösung zu Aufgabe 7

Schlüssel	Zelle
4	0
1	1
6	2
3	3
9	4
8	5
	6
	7
	8
	9
7	10

Notenverteilung Hauptklausur: