Technische Universität Chemnitz
Fakultät für Informatik
Prof. Dr. Andreas Goerdt

Chemnitz, den 13.02.2004

Algorithmen und Programmierung
Klausur zum WS 2003/2004 – Musterlösungen

1. Aufgabe: $2 + 2 + 1 = 5$ Punkte

a) Geben Sie die periodische Dualzahl $0,\overline{001}$ als Bruch im Dezimalsystem an.

Lösung:

$$\begin{aligned}
0,\overline{001} &= 2^{-3} + 2^{-6} + 2^{-9} + 2^{-12} + \ldots \\
&= 2^{-3} \cdot (2^0 + 2^{-3} + 2^{-6} + 2^{-9} + \ldots) \\
&= 2^{-3} \cdot ((2^{-3})^0 + (2^{-3})^1 + (2^{-3})^2 + (2^{-3})^3 + \ldots) \\
&= \sum_{i=0}^{\infty} 2^{-3} \cdot (2^{-3})^i \\
&= \frac{\frac{1}{8}}{1-\frac{1}{8}} \\
&= \frac{1}{7}
\end{aligned}$$

b) Transformieren Sie die Zahl $(0,35)_{10}$ in eine Dualzahl.

Lösung:

$$\begin{array}{lcll}
0,35 \cdot 2 & = & 0,7 & \rightarrow 0 \\
0,7 \cdot 2 & = & 1,4 & \rightarrow 1 \\
0,4 \cdot 2 & = & 0,8 & \rightarrow 0 \\
0,8 \cdot 2 & = & 1,6 & \rightarrow 1 \\
0,6 \cdot 2 & = & 1,2 & \rightarrow 1 \\
0,2 \cdot 2 & = & 0,4 & \rightarrow 0 \\
0,4 \cdot 2 & = & 0,8 & \rightarrow 0 \\
\vdots & & &
\end{array}$$

$$(0,35)_{10} = 0,01\overline{0110}_2$$

c) Betrachten Sie die Zweier-Komplementdarstellung auf n Bits und ergänzen Sie den Satz auf dem Lösungsblatt.

Lösung:
Ist $a = (b_{n-1} \ldots b_0)_{2Kpl}$, dann ist

$$(b_{n-1} \ldots b_0)_2 = \mathrm{MOD}(a, 2^n).$$

2. Aufgabe: $2 + 7 = 9$ Punkte

Gegeben sei folgendes Programmstück:

```
// Eingabe x>0, ganzzahlig
// Eingabe y>=0, ganzzahlig
z = 0;

while (y >= x) {
  y = y - x;
  z = z + 1;
}

// Ergebnis steht in z
```

a) Was berechnet dieses Programmstück (Wert der Variablen z nach Abarbeitung)?

Lösung:
Nach Abarbeitung des Programmstücks enthält die Variable z den Wert der ganzzahligen Division des eingelesenen Wertes für y durch den eingelesenen Wert für x.

b) Weisen Sie die Korrektheit Ihrer Aussage aus a) nach. Gehen Sie dabei gemäß dem Muster

- Aussage über die Endlichkeit der Anzahl der Schleifendurchläufe (1)
- Beweis einer geeigneten Invariante (Sie dürfen bei Bedarf die Operatoren DIV(x, y) (ganzzahlige Division von x durch y) und MOD(x,y) (Rest von DIV(x, y)) benutzen.) (4)
- Quintessenz (2)

vor.

Lösung:
Seien x_{start} und y_{start} die eingelesenen Werte für x und y sowie x_l, y_l und z_l die Werte der Variablen x, y und z nach dem l-ten Schleifendurchlauf. Dabei gilt $x_l = x_{start}$ $\forall l$, da der Wert der Variablen x während der Schleife nicht verändert wird.

1) Endlichkeit der Anzahl der Schleifendurchläufe n

   $x_{start} > 0$ und $y_{start} \geq 0$ sind endliche, ganze Zahlen. Die Schleife bricht ab, wenn der Wert von y kleiner als x_{start} ist. In jedem Schleifendurchlauf wird der Wert von y um x_{start} verringert. Folglich ist die Anzahl der Schleifendurchläufe endlich.

2) Beweis, dass DIV(y_{start}, x_{start}) berechnet wird

   Invariante:

$$\mathrm{DIV}(y_{start}, x_{start}) = z_l + \mathrm{DIV}(y_l, x_{start})$$

Induktionsanfang:
Für $l = 0$ gilt: $x_0 = x_{start}$, $y_0 = y_{start}$ und $z_0 = 0$
Überprüfung der Invariante:

$$\begin{aligned} \mathrm{DIV}(y_{start}, x_{start}) &= z_0 + \mathrm{DIV}(y_0, x_{start}) \\ &= \mathrm{DIV}(y_{start}, x_{start}) \end{aligned}$$

Die Invariante gilt.

Induktionschritt:
Sei $1 \leq l \leq n$ fest. Dann gilt für $l-1$ nach Induktionsvoraussetzung (Invariante):

$$\mathrm{DIV}(y_{start}, x_{start}) = z_{l-1} + \mathrm{DIV}(y_{l-1}, x_{start})$$

Wir betrachten nun den l-ten Schleifendurchlauf:

$$\begin{aligned} z_l &= z_{l-1} + 1 \\ y_l &= y_{l-1} - x_{start} \end{aligned}$$

Damit erhalten wir

$$\begin{aligned} z_l + \mathrm{DIV}(y_l, x_{start}) &= z_{l-1} + 1 + \mathrm{DIV}(y_{l-1} - x_{start}, x_{start}) \\ &= z_{l-1} + 1 + \mathrm{DIV}(y_{l-1}, x_{start}) - 1 \\ &= z_{l-1} + \mathrm{DIV}(y_{l-1}, x_{start}) \end{aligned}$$

und die Invariante gilt.

3) Quintessenz

Nach dem letzten (n-ten) Schleifendurchlauf ist die Schleifenbedingung nicht mehr erfüllt ($y_n < x_{start}$) und die Invariante gilt, also
$\mathrm{DIV}(y_{start}, x_{start}) = z_n + \mathrm{DIV}(y_n, x_{start}) = z_n + 0$.
Folglich ist in der Variable z nach Abarbeitung des Programmstücks der Wert $\mathrm{DIV}(y_{start}, x_{start})$ gespeichert. Die Aussage von a) ist also korrekt.

3. Aufgabe: 1 + 7 = 8 Punkte

Gegeben sei folgendes Programmstück:

```
// Eingabe y>=1, ganzzahlig
x = 1;

while (y > 0) {
    x = x * 3;
    y = y - 1;
}

// Ergebnis steht in x
```

a) Was berechnet dieses Programmstück (Wert der Variablen x nach Abarbeitung)?

Lösung:
Nach Abarbeitung des Programmstücks enthält die Variable x den Wert „3 hoch den eingelesenen Wert für y".

b) Weisen Sie die Korrektheit Ihrer Aussage aus a) nach. Gehen Sie dabei gemäß dem Muster

- Aussage über die Endlichkeit der Anzahl der Schleifendurchläufe (1)
- Beweis einer geeigneten Invariante (4)
- Quintessenz (2)

vor.

Lösung:
Sei y_{start} der eingelesene Wert für y sowie x_l und y_l die Werte der Variablen x und y nach dem l-ten Schleifendurchlauf.

1) Endlichkeit der Anzahl der Schleifendurchläufe n

y_{start} ist eine endliche, ganze Zahl ≥ 1. Die Schleife bricht ab, wenn der Wert von y gleich 0 ist. In jedem Schleifendurchlauf wird der Wert von y um 1 verringert. Folglich ist die Anzahl der Schleifendurchläufe endlich mit $n = y_{start}$.

2) Beweis, dass $3^{y_{start}}$ berechnet wird

Invariante:

$$3^{y_{start}} = x_l \cdot 3^{y_l}$$

Induktionsanfang:
Für $l = 0$ gilt: $x_0 = 1$ und $y_0 = y_{start}$
Überprüfung der Invariante:

$$\begin{aligned} 3^{y_{start}} &= x_0 \cdot 3^{y_0} \\ &= 3^{y_{start}} \end{aligned}$$

Die Invariante gilt.

Induktionschritt:
Sei $1 \leq l \leq n$ fest. Dann gilt für $l-1$ nach Induktionsvoraussetzung (Invariante):

$$3^{y_{start}} = x_{l-1} \cdot 3^{y_{l-1}}$$

Wir betrachten nun den l-ten Schleifendurchlauf:

$$\begin{array}{rcl} x_l & = & x_{l-1} \cdot 3 \\ y_l & = & y_{l-1} - 1 \end{array}$$

Damit erhalten wir

$$\begin{array}{rcl} x_l \cdot 3^{y_l} & = & x_{l-1} \cdot 3 \cdot 3^{y_{l-1}-1} \\ & = & x_{l-1} \cdot 3^{1+y_{l-1}-1} \\ & = & x_{l-1} \cdot 3^{y_{l-1}} \end{array}$$

und die Invariante gilt.

3) Quintessenz

Nach dem letzten (n-ten) Schleifendurchlauf ist die Schleifenbedingung nicht mehr erfüllt ($y_n = 0$) und die Invariante gilt, also
$3^{y_{start}} = x_n \cdot 3^{y_n} = x_n \cdot 1$.
Folglich ist in der Variable x nach Abarbeitung des Programmstücks der Wert $3^{y_{start}}$ gespeichert. Die Aussage von a) ist also korrekt.

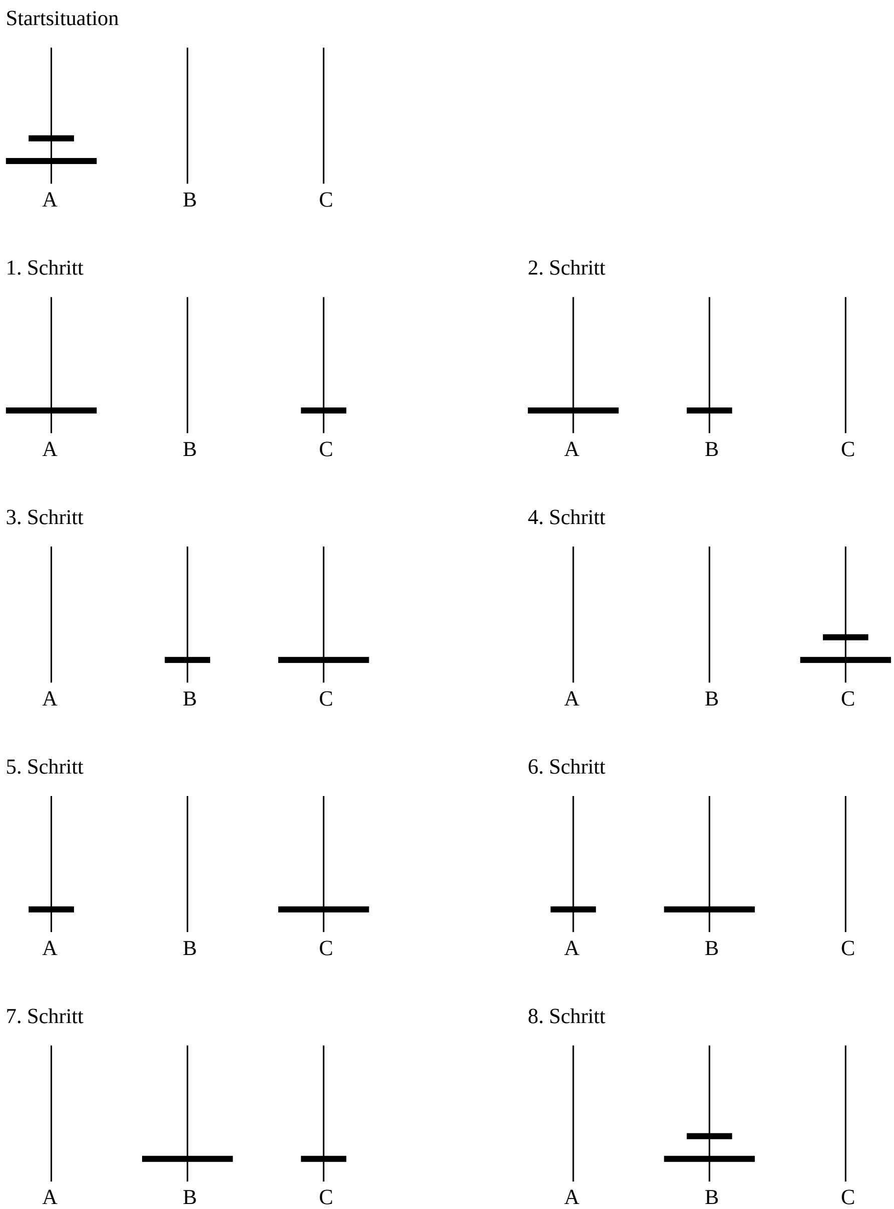
4. Aufgabe: $2 + 7 + 2 = 11$ Punkte

Betrachten Sie das Problem der „Türme von Hanoi“ mit 3 Stäben (A, B, C). Die Scheiben seien mit 1 (oberste, kleinste Scheibe) bis n (unterste, größte Scheibe) bezeichnet. Sie sollen von A nach B umgesetzt werden.

Zusätzlich muss folgende Einschränkung eingehalten werden: Das Umsetzen von Scheiben direkt von A nach B und umgekehrt von B nach A ist **nicht** erlaubt!

a) Zeichnen Sie für die vorgegebene Situation auf dem Lösungsblatt jeweils die Zwischen-Situationen nach jedem einzelnen Schritt bis zum Ende (alles auf Stab B) auf.

Lösung:



b) Tragen Sie in den Rahmen auf dem Lösungsblatt eine rekursive Methode mit **genau drei** rekursiven Aufrufen zur Ermittlung einer zulässigen Schrittfolge zur Umsetzung von n Scheiben ein. Für einen Schritt soll das Programm einen Text der Art „Scheibe i von ... nach ...“ ausgeben. Hinweis: Als Hilfsstab müssen Sie immer C benutzen.

Lösung:

```java
public class Tuerme {

  public static void hanoi(char a, char b, char c, int n){

    if (n==1) {
      System.out.println("Scheibe 1 von " + a + " nach " + c);
      System.out.println("Scheibe 1 von " + c + " nach " + b);
    }
    else {
      hanoi(a,b,c,n-1);
      System.out.println("Scheibe " + n +
          " von " + a + " nach " + c);
      hanoi(b,a,c,n-1);
      System.out.println("Scheibe " + n +
          " von " + c + " nach " + b);
      hanoi(a,b,c,n-1);
    }
  }

  public static void main(String[] args){
    char a = 'A', b = 'B', c = 'C';
    int n = IOTools.readInteger("n = ");
    hanoi(a,b,c,n);
  }
}
```

c) Geben Sie eine nicht-rekursive Formel für die Anzahl der Umsetzungen (Züge) Ihrer Methode aus b) in Abhängigkeit von n an.

Lösung:
Sei H_n die Anzahl der Züge, die die Methode ausgibt, um n Scheiben umzusetzen.
Aus der Methode lässt sich ableiten:

$$\begin{array}{rcl} H_1 & = & 2 \\ H_n & = & 3 \cdot H_{n-1} + 2 \end{array}$$

Daraus ergibt sich:

$$\begin{array}{rcl} H_n & = & 3 \cdot H_{n-1} + 2 \\ & = & 3 \cdot (3 \cdot H_{n-2} + 2) + 2 \\ & = & 3 \cdot (3 \cdot (3 \cdot H_{n-3} + 2) + 2) + 2 \\ & = & 3 \cdot \Big(\ldots \Big(3 \cdot H_{n-(n-1)} + 2 \Big) \ldots + 2 \Big) + 2 \end{array}$$

Da

$$H_{n-(n-1)} = H_1 = 2$$

ist, folgt

$$\begin{array}{rcl} H_n & = & 3^{n-1} \cdot 2 + 3^{n-2} \cdot 2 + 3^{n-3} \cdot 2 + \ldots + 3^0 \cdot 2 \\ & = & \sum_{i=0}^{n-1} 2 \cdot 3^i \\ & = & 2 \cdot \frac{3^n - 1}{3 - 1} \\ & = & 3^n - 1 \end{array}$$

5. Aufgabe: 4 Punkte

Gegeben sei ein aufsteigend sortiertes Feld f ganzer Zahlen. Betrachten Sie folgende Version der binären Suche nach einem Element x.

```
int a, b, h;

a = 0;
b = f.length - 1;

while (b > a) {
  h = (a + b) / 2;
  if (x <= f[h])
    b = h;
  if (x >= f[h])
    a = h;
}

if (f[a] == x)
  // Ausgabe: x an Position a enthalten
else
  // Ausgabe: x nicht enthalten
```

Warum ist der Algorithmus nicht für alle möglichen Eingaben von x korrekt? Begründen Sie Ihre Aussage durch ein Beispiel!

Lösung:

Der Algorithmus verursacht bei bestimmten Ausgangssituationen Endlosschleifen.

- Das größte Element des Feldes wird gesucht. Der Algorithmus „hängt" bei a = zweitgrößter Index, b = größter Index.

  Beispiel: Feld = {2,3}, gesucht wird 3
- Das gesuchte Element ist nicht enthalten, aber alle Feldelemente sind kleiner als das gesuchte. Der Algorithmus hängt wie oben.

  Beispiel: Feld = {2,3}, gesucht wird 5
- Das gesuchte Element ist nicht enthalten, und die Feldelemente sind ungünstig verteilt.

  Beispiel: Feld = {1,3}, gesucht wird 2

6. Aufgabe: 5 + 3 = 8 Punkte

a) Finden Sie alle Fehler in folgendem Programmstück:

```
int i;
boolean[] feld;
short x;

for (i = 0; i <= feld.length; i++)
  feld[i] = 1;

x = x + 1;
```

Lösung:

- `feld` ist nicht erzeugt
- Indexüberschreitung bei `feld` im letzten Schleifendurchlauf
- einer `boolean`-Variable (Elemente von `feld`) kann keine 1 zugewiesen werden
- `x` ist nicht initialisiert, Inkrementierung daher nicht möglich
- Operation + hat Ergebnistyp `int`, folglich Typecast zu `short` notwendig

b) Was gibt folgendes Programm aus?

```
public class Test{

  static int x=1, y=2, z=3;

  public static void up(int x){
    int y=5;
    System.out.println(x);
    System.out.println(y);
    System.out.println(z);
    z=4;
  }

  public static void main(String[] args){
    int y=10;
    up(y);
    System.out.println(x);
    System.out.println(y);
    System.out.println(z);
  }
}
```

Lösung:

10

5

3

1

10

4