

Klausur Programmierparadigmen — Beispiellösung

SS2016, 22. September 2016, 11:00 – 13:00 Uhr

Zugelassene Hilfsmittel: Papierbasierte Quellen (Vorlesungsfolien, Übungsblätter, eigene Aufzeichnungen, Bücher, ...)

Die Verwendung von elektronischen Geräten ist verboten.

Bearbeitungszeit: 120 min

Aufgabe 1 (Haskell)

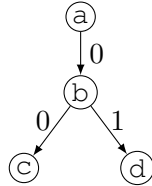
[26 Punkte]

Gegeben ist der Typ für Mehrwegbäume

```
data Tree a = Node a [Tree a]
```

sowie zwei Beispielbäume

```
t1 = Node "a" [
  Node "b" [
    Node "c" [],
    Node "d" []
  ]
]
```



```
t2 = Node "Z" []
```

ⓧ

- (a) Der Weg von der Wurzel eines Baumes zu einem Teilbaum kann durch eine Liste natürlicher Zahlen beschrieben werden. Jedes Element solch einer (*gültigen*) *Position* steht dabei für den Index einer ausgehenden Kante (gezählt ab 0). Alle anderen Listen nennen wir *ungültige Positionen*. [5 Punkte]

Implementieren Sie die Funktion

```
treeIndex :: Tree a -> [Int] -> Tree a
```

sodass der Aufruf `treeIndex t p` den durch die Position `p` angegebenen Teilbaum von `t` zurückgibt. Falls `p` ungültig ist, beenden Sie die Ausführung durch Aufruf von `error`.

Beispiel:

```
t1          =>+ Node "a" [Node "b" [Node "c" [], Node "d" []]]
treeIndex t1 []      =>+ Node "a" [Node "b" [Node "c" [], Node "d" []]]
treeIndex t1 [0,1]   =>+ Node "d" []
treeIndex t1 [1]     =>+ ⊥    Exception: invalid position
treeIndex t1 [0,1,0] =>+ ⊥    Exception: invalid position
```

- (b) Implementieren Sie eine Funktion [6 Punkte]

```
treePositions :: Tree a -> [[Int]]
```

die eine Liste aller im obigen Sinne gültigen Positionen eines Baumes berechnet.

Beispiel:

```
treePositions t1 == [ [], [0], [0,0], [0,1] ]
treePositions t2 == [ [] ]
```

Hinweis: Sie können List Comprehensions verwenden.

- (c) Implementieren Sie eine Funktion [10 Punkte]

```
changeTree :: (Tree a -> Tree a) -> [Int] -> Tree a -> Tree a
```

Der Aufruf `changeTree f p t` ändert in `t` den durch `p` beschriebenen Teilbaum ab, indem er `f` auf diesen Teilbaum anwendet. Für ungültiges `p` wird `t` unverändert zurückgegeben.

Beispiel:

```
let f (Node _ xs) = Node "Z" xs
t1 =>+ Node "a" [Node "b" [Node "c" [], Node "d" []]]
changeTree f []      t1 =>+ Node "Z" [Node "b" [Node "c" [], Node "d" []]]
changeTree f [0]     t1 =>+ Node "a" [Node "Z" [Node "c" [], Node "d" []]]
changeTree f [0,1]   t1 =>+ Node "a" [Node "b" [Node "c" [], Node "Z" []]]
changeTree f [0,1,0] t1 =>+ Node "a" [Node "b" [Node "c" [], Node "d" []]]
```

(d) Verwenden Sie nun `changeTree` zur Implementierung einer Funktion

[5 Punkte]

```
overrideTree :: Tree a -> [Int] -> Tree a -> Tree a
```

Der Aufruf `overrideTree t' p t` ersetzt in `t` den durch `p` beschriebenen Teilbaum durch `t'`. Für ungültiges `p` wird `t` unverändert zurückgegeben.

Beispiel:

```

                                t1 =>+ Node "a" [Node "b" [Node "c" [], Node "d" []]]
overrideTree t2 []             t1 =>+ Node "Z" []
overrideTree t2 [0]            t1 =>+ Node "a" [Node "Z" []]
overrideTree t2 [0,1,0] t1 =>+ Node "a" [Node "b" [Node "c" [], Node "d" []]]

```

Beispiellösung:

```

ith :: [a] -> Int -> a
ith [] i = error "invalid index"
ith (x:xs) 0 = x
ith (x:xs) i = ith xs (i-1)

treeIndex :: Tree a -> [Int] -> Tree a
treeIndex t [] = t
treeIndex (Node a ts) (i:is)
  | i < 0 = error "invalid position"
  | i >= length ts = error "invalid position"
  | otherwise = treeIndex (ith ts i) is

treePositions :: Tree a -> [[Int]]
treePositions (Node _ ts) =
  [] : [ (i : is) | i <- [0..(length ts - 1)],
          is <- treePositions (ith ts i) ]

changeListAt :: (a -> a) -> Int -> [a] -> [a]
changeListAt f _ [] = []
changeListAt f 0 (x:xs) = (f x : xs)
changeListAt f n (x:xs) = x : changeListAt f (n-1) xs

changeTree :: (Tree a -> Tree a) -> [Int] -> Tree a -> Tree a
changeTree f [] t = f t
changeTree f (i:is) t@(Node a ts)
  | i < 0 = t
  | i >= length ts = t
  | otherwise = Node a (changeListAt (changeTree f is) i ts)

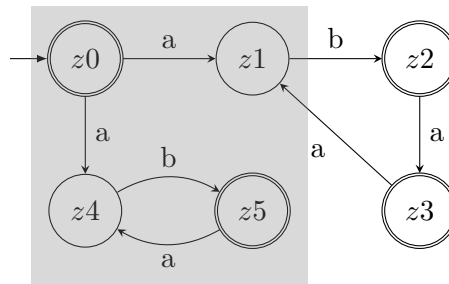
overrideTree :: Tree a -> [Int] -> Tree a -> Tree a
overrideTree t = changeTree (\x -> t)

```

Aufgabe 2 (Prolog, nichtdeterministische endliche Automaten)

[9 Punkte]

Gegeben sei folgender endlicher nichtdeterministischer Automat:



- (a) Spezifizieren Sie den Automaten in Prolog. Definieren Sie hierzu die Übergangsrelation als Prolog-Fakten `delta(Z1, X, Z2)`, sowie Fakten `final(Z)`, die die Endzustände charakterisieren. [3 Punkte]
Es reicht hierbei, wenn Sie sich auf den markierten Bereich beschränken.
- (b) Implementieren Sie ein Prolog-Prädikat `run(Z, In, Zs)`, das den Automaten in Startzustand `Z` laufen lässt mit Eingabe `In`. `In` ist dabei eine Liste von Atomen. Falls der Automat akzeptiert, soll `Zs` die durchlaufene Zustandsfolge als Liste enthalten. `run` soll reerfüllbar sein, falls durch Nichtdeterminismus verschiedene Akzeptanzpfade möglich sind. [4 Punkte]

Beispiel:

```
?run(z1, [b, a], Zs).  
Zs = [z1, z2, z3];  
false.
```

```
?run(z0, [a, b], Zs).  
Zs = [z0, z1, z2];  
Zs = [z0, z4, z5];  
false;
```

```
?run(z0, [a], Zs).  
false.
```

- (c) Implementieren Sie nun ein Prädikat `accept(In)`, das genau dann erfüllbar ist, falls der Automat die Eingabe `In` akzeptiert. [2 Punkte]

Beispiellösung:

(a) `delta(z0,a,z1).`
`delta(z0,a,z4).`
`delta(z4,b,z5).`
`delta(z5,a,z4).`

`final(z0).`
`final(z5).`

(b) `run(Z,[],[Z]):- final(Z).`
`run(Z,[X|R],[Z|ZL]):- delta(Z,X,Z2), run(Z2,R,ZL).`

(c) `accept(In) :- run(z0, In, _).`

oder, falls der Automat einfach austauschbar sein soll,
`start(z0).`

`accept(In) :- start(S), run(S, In, _).`

Aufgabe 3 (Prolog, Datenbank)

[8 Punkte]

Wir modellieren das Campus-Verwaltungssystem des KIT in Prolog. Zur Vereinfachung identifizieren wir einen Studenten über seine Matrikelnummer. Den Datenbestand stellen wir als Prolog-Fakten dar:

- Einen Fakt `note(M, V, N)` für jede Note N , die der Student mit Matrikelnummer M in Veranstaltung V erhalten hat.
Beispiel: `note(10001, 24030, 1.7)`.
- Einen Fakt `voraussetzungen(V, Vs)` für jede Veranstaltung V , sodass Vs eine Liste der Veranstaltungsnummern ist, die der Student bereits bestanden haben muss, um die Prüfung von V antreten zu dürfen.
Beispiel: `voraussetzungen(24030, [24005])`.

Implementieren Sie folgende Funktionalität:

- (a) Ein Prädikat `bestanden(M, V)`, [1 Punkt]
das genau dann erfüllt ist, wenn der Student in der Vorlesung V eine Note von 4.0 oder besser bekommen hat.
- (b) Ein Prädikat `darf_pruefen(M, V)`, [4 Punkte]
das genau dann erfüllt ist, wenn Student mit Matrikelnummer M die Prüfung mit Nummer V antreten darf. Dies ist der Fall, wenn der Student alle vorausgesetzten Klausuren bestanden hat.
- (c) Ein Prädikat `tv(V, V2)` (für “transitive Voraussetzung”), [3 Punkte]
das genau dann erfüllt ist, wenn $V2$ direkt oder indirekt Voraussetzung für V ist. Ist $V2$ uninstantiiert, sollen alle solchen $V2$ generiert werden.
Hinweis: Sie können das Prädikat `member` aus der Vorlesung verwenden.

Beispiellösung:

- (a) `bestanden(M, V) :- note(M, V, N), N =< 4.0.`
- (b) `alle_bestanden(_, []).`
`alle_bestanden(M, [V|Vs]) :- bestanden(M, V), alle_bestanden(M, Vs).`

`darf_pruefen(M, V) :- voraussetzungen(V, Vs), alle_bestanden(M, Vs).`
- (c) `tv(V, V2) :- voraussetzungen(V, Vs), member(V2, Vs).`
`tv(V, V2) :- voraussetzungen(V, Vs), member(V1, Vs), tv(V1, V2).`

Aufgabe 4 (Zähler-Objekte im λ -Kalkül, β -Reduktion)

[16 Punkte]

Zähler-Objekte reagieren auf die Nachrichten *inc*, *add*, *get*. In Java ließe sich definieren:

```
class Zähler {
    int x;
    Zähler(int x) { this.x = x; }
    public Zähler inc()      { return new Zähler(x + 1); }
    public Zähler add(int y) { return new Zähler(x + y); }
    public int   get()      { return x; }
}
```

Im λ -Kalkül lassen sich Nachrichten – ähnlich wie Church-Booleans – darstellen als:

$$\begin{aligned}\text{inc} &= \lambda i. \lambda a. \lambda g. i \\ \text{add} &= \lambda i. \lambda a. \lambda g. a \\ \text{get} &= \lambda i. \lambda a. \lambda g. g\end{aligned}$$

Weiterhin sei definiert:

$$\begin{aligned}\text{react} &= \lambda n. \lambda x. \lambda m. m \ (n \ (\text{succ } x)) \\ &\quad (\lambda y. n \ (\text{plus } x \ y)) \\ &\quad x\end{aligned}$$

Mit *succ* und *plus* wie bekannt: $\text{succ } c_x \Rightarrow^* c_{x+1}$ und $\text{plus } c_x \ c_y \Rightarrow^* c_{x+y}$ für Church-Zahlen c_x, c_y .

- (a) Zeigen Sie, dass für Church-Zahlen c_x, c_y und beliebige λ -Terme n :

[10 Punkte]

$$\begin{aligned}\text{react } n \ c_x \ \text{inc} &\Rightarrow^* n \ c_{x+1} \\ \text{react } n \ c_x \ \text{add } c_y &\Rightarrow^* n \ c_{x+y} \\ \text{react } n \ c_x \ \text{get} &\Rightarrow^* c_x\end{aligned}$$

- (b) Hätten wir einen λ -Term *new*, der neue Zähler-Objekte konstruiert, so ließe sich ein Zähler mit Stand c_x einfach darstellen als:

[3 Punkte]

$$z_x = \text{react } \text{new } c_x$$

Geben Sie also einen λ -Term *new* an, sodass¹ für Church-Zahlen c_x :

$$\text{new } c_x \Rightarrow^* \text{react } \text{new } c_x$$

Zur Erinnerung: Der λ -Kalkül erlaubt keine rekursiven Definitionen. Verwenden Sie stattdessen einen Fixpunktkombinator wie *Y*.

- (c) Zeigen Sie, dass mit solchem *new*:

[3 Punkte]

$$\begin{aligned}z_x \ \text{inc} &\Rightarrow^* z_{x+1} \\ z_x \ \text{add } c_y &\Rightarrow^* z_{x+y} \\ z_x \ \text{get} &\Rightarrow^* c_x\end{aligned}$$

Hinweis: Verwenden Sie das in (a) und (b) Gezeigte.

¹Es reicht auch, wenn $\text{new } c_x \Rightarrow^* X \Leftarrow^* \text{react } \text{new } c_x$ für einen Term X

Beispiellösung:

(a)

$$\begin{aligned}
\text{react } n \ c_x \ \text{inc} &\Rightarrow^3 \text{inc } (n \ (\text{succ } c_x)) \ (\lambda Y. n \ (\text{plus } c_x \ Y)) \ c_x \\
&\Rightarrow^3 n \ (\text{succ } c_x) \\
&\Rightarrow^* n \ c_{x+1} \\
\text{react } n \ c_x \ \text{add } c_y &\Rightarrow^3 \text{add } (n \ (\text{succ } c_x)) \ (\lambda Y. n \ (\text{plus } c_x \ Y)) \ c_x \ c_y \\
&\Rightarrow^3 (\lambda Y. n \ (\text{plus } c_x \ Y)) \ c_y \\
&\Rightarrow n \ (\text{plus } c_x \ c_y) \\
&\Rightarrow^* n \ c_{x+y} \\
\text{react } n \ c_x \ \text{get} &\Rightarrow^3 \text{get } (n \ (\text{succ } c_x)) \ (\lambda Y. n \ (\text{plus } c_x \ Y)) \ c_x \\
&\Rightarrow^3 c_x
\end{aligned}$$

(b) Sei

$$\text{New} = \lambda n. \lambda x. \text{react } n \ x$$

Dann wähle new als Resultat der Reduktion:

$$Y \ \text{New} \Rightarrow \underbrace{(\lambda x. \text{New } (x \ x)) \ (\lambda x. \text{New } (x \ x))}_{=: \text{new}}$$

Alternativ: Für das Verhalten wie in Fußnote¹ reicht schon:

$$\text{new} = Y \ \text{New}$$

oder einfach:

$$\text{new} = Y \ \text{react}$$

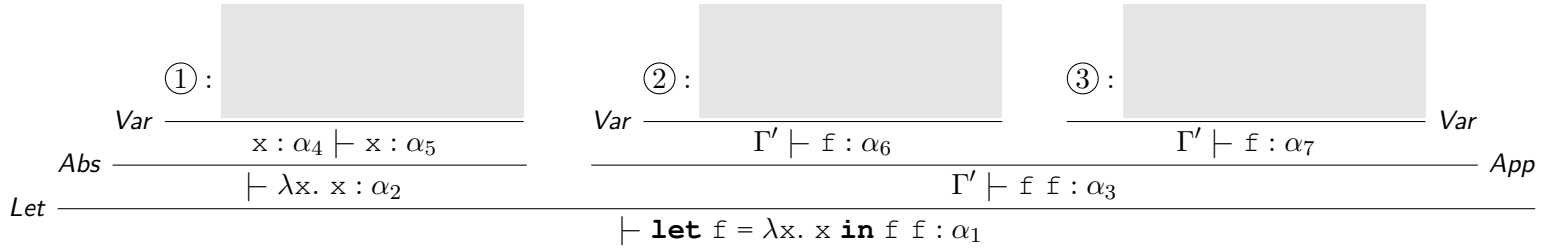
(c)

$$\begin{aligned}
z_x \ \text{inc} &\underbrace{\Rightarrow^*}_{\text{per a)}} \text{new } c_{x+1} \underbrace{\Rightarrow^*}_{\text{per b)}} z_{x+1} \\
z_x \ \text{add } c_y &\underbrace{\Rightarrow^*}_{\text{per a)}} \text{new } c_{x+y} \underbrace{\Rightarrow^*}_{\text{per b)}} z_{x+y} \\
z_x \ \text{get} &\underbrace{\Rightarrow^*}_{\text{per a)}} c_x
\end{aligned}$$

Aufgabe 5 (Typinferenz)

[21 Punkte]

Betrachten Sie das folgende Skelett einer Typherleitung für den Lambda-Term **let** $f = \lambda x. x$ **in** $f f$.



- (a) Vervollständigen Sie Box ① in der oben gegebenen Typherleitung.

[2 Punkte]

Beispiellösung: ① : $(x : \alpha_4)x = \alpha_5$

- (b) Bestimmen Sie die Typgleichungssysteme C_0 und C_{let} (vgl. Skript „Typinferenz für let“, Folie 324).

[3 Punkte]

Beispiellösung:

$$\begin{aligned}
 C_0 &= \{\alpha_1 = \alpha_3\} \\
 C_{let} &= \{\alpha_2 = \alpha_4 \rightarrow \alpha_5, \alpha_4 = \alpha_5\}
 \end{aligned}$$

Verwenden Sie im Folgenden $\sigma_{let} = [\alpha_2 \dot{\rightarrow} \alpha_5 \rightarrow \alpha_5, \alpha_4 \dot{\rightarrow} \alpha_5]$.

- (c) Berechnen Sie nun Γ' und vervollständigen Sie die Boxen ② und ③ in der gegebenen Typherleitung.

[6 Punkte]

Beispiellösung:

$$\begin{aligned}
 \Gamma' &= \sigma_{let}(\emptyset), f : ta(\sigma_{let}(\alpha_2), \sigma_{let}(\emptyset)) \\
 &= f : ta(\alpha_5 \rightarrow \alpha_5, \emptyset) \\
 &= f : \forall \alpha_5. \alpha_5 \rightarrow \alpha_5 \\
 \textcircled{2} : \Gamma'(f) &= \forall \alpha_5. \alpha_5 \rightarrow \alpha_5 \succeq \alpha_8 \rightarrow \alpha_8 \\
 \textcircled{3} : \Gamma'(f) &= \forall \alpha_5. \alpha_5 \rightarrow \alpha_5 \succeq \alpha_9 \rightarrow \alpha_9
 \end{aligned}$$

- (d) Bestimmen Sie das vollständige Typgleichungssystem C und berechnen Sie daraus einen allgemeinsten Unifikator σ .

[10 Punkte]

Beispiellösung:

$$\begin{aligned}
C &= C_0 \cup C'_{let} \cup \{ \alpha_6 = \alpha_7 \rightarrow \alpha_3, \\
&\quad \alpha_6 = \alpha_8 \rightarrow \alpha_8, \\
&\quad \alpha_7 = \alpha_9 \rightarrow \alpha_9 \\
&\quad \} \\
&= \{ \alpha_1 = \alpha_3, \\
&\quad \alpha_2 = \alpha_5 \rightarrow \alpha_5, \\
&\quad \alpha_4 = \alpha_5, \\
&\quad \alpha_6 = \alpha_7 \rightarrow \alpha_3, \\
&\quad \alpha_6 = \alpha_8 \rightarrow \alpha_8, \\
&\quad \alpha_7 = \alpha_9 \rightarrow \alpha_9 \\
&\quad \} \\
\sigma &= \left[\begin{array}{c} \alpha_1 \dot{\vdash} \alpha_9 \rightarrow \alpha_9, \\ \alpha_2 \dot{\vdash} \alpha_5 \rightarrow \alpha_5, \\ \alpha_3 \dot{\vdash} \alpha_9 \rightarrow \alpha_9, \\ \alpha_4 \dot{\vdash} \alpha_5, \\ \alpha_6 \dot{\vdash} (\alpha_9 \rightarrow \alpha_9) \rightarrow (\alpha_9 \rightarrow \alpha_9), \\ \alpha_7 \dot{\vdash} \alpha_9 \rightarrow \alpha_9, \\ \alpha_8 \dot{\vdash} \alpha_9 \rightarrow \alpha_9 \end{array} \right]
\end{aligned}$$

Der vollständige Typinferenzbaum:

$$\begin{array}{c}
\begin{array}{c} \Gamma'(\mathbf{f}) = \forall \alpha_5. \alpha_5 \rightarrow \alpha_5 \quad \Gamma'(\mathbf{f}) = \forall \alpha_5. \alpha_5 \rightarrow \alpha_5 \\ \text{Var} \frac{(x : \alpha_4) x = \alpha_5}{x : \alpha_4 \vdash x : \alpha_5} \quad \text{Var} \frac{\succeq \alpha_8 \rightarrow \alpha_8}{\Gamma' \vdash \mathbf{f} : \alpha_6} \quad \frac{\succeq \alpha_9 \rightarrow \alpha_9}{\Gamma' \vdash \mathbf{f} : \alpha_7} \text{Var} \\ \text{Abs} \frac{}{\vdash \lambda x. x : \alpha_2} \quad \frac{}{\Gamma' \vdash \mathbf{f} \mathbf{f} : \alpha_3} \text{App} \\ \text{Let} \frac{}{\vdash \mathbf{let} \mathbf{f} = \lambda x. x \mathbf{in} \mathbf{f} \mathbf{f} : \alpha_1} \end{array}
\end{array}$$

Aufgabe 6 (Java: puffernder asynchroner Schreiber)

[16 Punkte]

Ein `BufferedAsyncWriter` ist ein `Thread`, welcher Zeilen (`Strings`) zwischenspeichert und diese irgendwann in eine Textdatei schreibt. Die Motivation hinter einem `BufferedAsyncWriter` ist es zu vermeiden, dass aufrufende `Threads` blockierende I/O-Operationen durchführen. Ihre Aufgabe ist es die fehlenden Methodenrumpfe der Klasse `BufferedAsyncWriter` zu implementieren.

Ein anderer `Thread` kann **write** aufrufen um dem `BufferedAsyncWriter` eine Zeile zu übergeben. Die Zeile wird zuerst nur in der `ArrayList` `buffer` gepuffert. Sobald der Inhalt von `buffer` 100 Einträge erreicht, soll der `BufferedAsyncWriter` aktiv werden. Verwenden Sie hierzu die Signalisierungsmethoden der Klasse `Object`.

Das Verhalten eines `BufferedAsyncWriter` wird in der Methode **run** implementiert. In der privaten **flushBuffer**-Methode wird mithilfe von `writer` der Inhalt von `buffer` in eine Textdatei geschrieben und im Anschluss geleert. Der `BufferedAsyncWriter` soll (zusätzlich zum 100-Einträge-Auslöser) alle 1000 Millisekunden `flushBuffer` aufrufen. Verwenden Sie hierfür die Signalisierungsmethode mit `Timeout` `wait(milliseconds)` von `Object`.

Durch Aufrufen der `interrupt`-Methode von `BufferedAsyncWriter` (die Methode wird von `Thread` geerbt) kann dem `BufferedAsyncWriter` signalisiert werden, dass er sich beenden soll. Verwenden Sie hierzu in `run` die Methode `isInterrupted`¹ und bedenken Sie, dass `wait` eine `InterruptedException` werfen kann.

Sie werden die folgenden Methoden brauchen:

- `ArrayList.size()`: liefert die Anzahl an Elementen in der `ArrayList`.
- `ArrayList.add(Object)`: fügt ein Element zur `ArrayList` hinzu.
- `ArrayList.clear()`: leert die `ArrayList`.
- `PrintWriter.println(String)`: schreibt den `String` in die Textdatei.

Beispiellösung:

```
public class BufferedAsyncWriter extends Thread {
    private ArrayList<String> buffer = new ArrayList<String>();
    private PrintWriter writer;

    public BufferedAsyncWriter(String textFilePath) {
        try { writer = new PrintWriter(textFilePath, "UTF-8"); }
        catch (Exception e) { ... }
    }
}
```

¹siehe Java-Folien 13–15

```
public void write(String line) {
    synchronized (this) { //sync auf buffer geht auch
        buffer.add(line);
        if (buffer.size() >= 100) {
            notifyAll();
            //notify ist auch möglich
        }
    }
}

public void run() {
    do {
        synchronized (this) {
            try {
                wait(1000);
                //muss im sync sein
            } catch (InterruptedException e) {
                interrupt();
                //oder andere korrekter Abbruch
                //flushBuffer muss vorher nochmal aufgerufen werden
            }
            flushBuffer();
            //entweder dieser Aufruf oder der Inhalt
            //von flushBuffer muss synchronized sein
        }
    } while (!isInterrupted());
    // do while ist nötig um mindestens einen Schleifendurchlauf zu erzwingen
    // bei write() gleich gefolgt von interrupt() würde sonst close() aufgerufen
    // hierfür gibt es allerdings keinen Punktabzug

    writer.close();
}

private void flushBuffer() {
    for (String line : buffer) {
        writer.println(line);
    }
    buffer.clear();
}
}
```

Aufgabe 7 (Java: ArrayList)

[7 Punkte]

Gegeben ist die Implementierung einer ArrayList ähnlichen Klasse. Sobald die Kapazität des internen Arrays überschritten wird, wird es durch ein neues Array von doppelter Größe ersetzt.

```
1 public class MyArrayList {
2     private Object[] array = new Object[1];
3     private int count = 0; // count also points to the next free index
4
5     public void add(Object o) {
6         // array full?
7         if (count == array.length) {
8             // double size
9             Object[] temp = new Object[array.length * 2];
10
11             // array copy
12             for (int i = 0; i < array.length; i++) {
13                 temp[i] = array[i];
14             }
15
16             temp[count] = o;
17
18             // use new, bigger array
19             array = temp;
20         } else {
21             array[count] = o;
22         }
23
24         count++;
25     }
26
27     public synchronized Object remove() {
28         count--;
29         Object o = array[count];
30         array[count] = null;
31         return o;
32     }
33 }
```

Nehmen Sie nun an, dass ein Objekt von `MyArrayList` durch mehrere Threads verwendet wird.

- (a) Kann es dabei zu Deadlocks kommen? Falls ja, geben Sie eine Abarbeitungsfolge [1.5 Punkte] an, die dazu führt. Falls nein, begründen Sie Ihre Antwort.

Beispiellösung: Nein kann es nicht. Es gibt nur ein `synchronized`. Aus diesem Block gibt es keine ausgehenden Aufrufe, die in ein anderes `synchronized` führen.

- (b) Kann es bei mehreren nebenläufigen Aufrufen von `add` dazu kommen, dass ein [2.5 Punkte] übergebenes Objekt danach nicht im Array enthalten ist? Falls ja, geben Sie eine Abarbeitungsfolge an, die dazu führt. Falls nein, begründen Sie Ihre Antwort.

Beispiellösung: Ja.

Thread 1 und 2 betreten "gleichzeitig" die Methode.

(Das Array hat noch genug Kapazität.)

Thread 1 führt Zeile 21 aus.

Thread 2 führt Zeile 21 aus. (das Objekt von Thread 1 geht verloren)

Erst danach wird `count` von beiden Threads erhöht.

- (c) Kann es sein, dass ein durch `remove` zurückgegebenes Objekt weiterhin im Array [3 Punkte] enthalten ist? Falls ja, geben Sie eine Abarbeitungsfolge an, die dazu führt. Falls nein, begründen Sie Ihre Antwort.

Beispiellösung: Ja

Thread 1 betritt `add`. array ist voll und muss erweitert werden.

Thread 1 kopiert das alte array, wird aber bei Zeile 16 unterbrochen.

Thread 2 betritt `remove` und entfernt ein Objekt aus dem aktuellen array und verlässt die Methode.

Thread 1 führt den Rest von `add` aus.

(Wegen Zeile 19 ist das entfernte Element im vergrößerten array wieder vorhanden.)

Aufgabe 8 (MPI: Parallelisierung)

[7 Punkte]

Gegeben ist der nachfolgende C-Code. Die Methode `processData` wendet auf jedes Element eines Arrays die (möglicherweise sehr rechenaufwendige) Methode `func` an. Danach wird die Summe der Ergebnisse berechnet und in der ersten Zelle des Arrays gespeichert. Der Parameter `n` gibt die Anzahl an Elementen im Array an.

```
void processData(int data[], int n) {
    int sum = 0;
    for (int i = 0; i < n; i++) {
        sum += func(data[i]);
    }
    data[0] = sum;
}
```

Ihre Aufgabe ist es im nachfolgenden Code die Lücken so auszufüllen, dass die oben gegebene Funktionalität mit MPI parallel bearbeitet wird. Sie können davon ausgehen, dass `n` ohne Rest durch die Anzahl an MPI-Prozessen teilbar ist. `MPI_Init` wurde vor einem Aufruf von `processData` bereits aufgerufen. Der Parameter `data` enthält nur für den Root-Prozess die zu verarbeitenden Daten.

```
void processData(int data[], int n, int root) {
    int numProcs, rank;
    int temp[n];

    MPI_Comm_size(MPI_COMM_WORLD, &numProcs);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);

    int sendcount = n / numProcs;

    // distribute data to processes
    MPI_Scatter(data, sendcount, MPI_INT, temp, sendcount, MPI_INT,
               root, MPI_COMM_WORLD);

    // apply function and sum up
    int sum = 0;
    for (int i = 0; i < sendcount; i++) {
        sum += func(temp[i]);
    }
    temp[0] = sum;

    // collect and sum up data
    MPI_Reduce(temp, data, 1, MPI_INT, MPI_Sum, root, MPI_COMM_WORLD);
}
```


Name:

Matrikelnummer:

Aufgabe 9 (Java-Bytecode)

[10 Punkte]

Übersetzen Sie folgenden Java-Programmausschnitt in Java-Bytecode:

```
if ((a < b) || !(a < c) || (c < b)) && !(c < 0) {  
    c = b + a;  
}
```

Gehen Sie dabei von folgenden Rahmenbedingungen aus:

- Bei a, b und c handelt es sich um `int`-Variablen.
- a, b und c sind als lokale Variablen 0, 1 und 2 verfügbar.

Hinweise:

- Sie können das aus der Vorlesung bekannte Schema zur Codeerzeugung für bedingte Sprünge verwenden (Folien 422ff.).
- Um eine Bedingung der Form `!cond` zu übersetzen, genügt es, `cond` zu übersetzen und die Sprungziele geeignet anzupassen.

Beispiellösung:

```
    iload_0  
    iload_1  
    if_icmplt condition4  
    iload_0  
    iload_2  
    if_icmplt afterif  
    iload_2  
    iload_1  
    if_icmplt afterif  
    goto condition4 // optional  
condition4:  
    iload_2  
    iconst_0  
    if_icmplt afterif  
    goto thenbranch // optional  
thenbranch:  
    iload_0  
    iload_1  
    iadd  
    istore_2  
afterif:
```

Name:

Matrikelnummer:
