Prüfungsaufgaben für das 1. Staatsexamen im Fach Informatik (vertieft studiert)

Inhaltsverzeichnis

1 Theoretische Informatik

1.1 Herbst 1989

Aufgabe 1

Hinweis:
Verwenden Sie zur Beschreibung der in dieser Aufgabe zu entwickelnden Algorithmen eine konkrete Programmiersprache wie PASCAL, MODULA o.ä. oder einen diesen Sprachen verwandten, in einschlägigen Vorlesungen oder Büchern üblicherweise benutzten "Pseudocode".

Gegeben sei der Zeichenvorrat A = {a, b, c}. M sei die Menge aller Zeichenreihen x über A mit der Eigenschaft, dass die Anzahl, wie oft das Zeichen a in x vorkommt, eine gerade Zahl ist.

1. Die Grammatik Γ habe A als Menge der Terminalzeichen, die Nichtterminalzeichen G und U, das Axiom G und die Produktionsregeln

   $G \rightarrow \varepsilon$ $\quad U \rightarrow bU$
   $G \rightarrow bG$ $\quad U \rightarrow cU$
   $G \rightarrow cG$ $\quad U \rightarrow aG$
   $G \rightarrow aU$
   (ε bezeichne die leere Zeichenreihe.)

   Beweisen Sie, dass für den Sprachschatz $L(\Gamma)$ von Γ gilt: $L(\Gamma) = M$.

2. Geben Sie einen endlichen Automaten an, der genau die Zeichenreihen von M akzeptiert.

3. Geben Sie eine kontextfreie Grammatik $L(\Gamma_1)$ an, für die ebenfalls $L(\Gamma_1) = M$ gilt, die jedoch weniger Produktionsregeln als Γ hat.

4. Formulieren Sie in Anlehnung an Γ einen rekursiven Algorithmus

   ```
   function TESTM(string s):boolean;
        ...
   ```

   der testet, ob eine Zeichenreihe s aus M ist oder nicht. Beweisen Sie, dass der Algorithmus für alle Eingaben s der Sorte string terminiert.

   Hinweis:
   Nehmen Sie dabei an, dass die Datenstruktur string (der Zeichenreihen über einem Zeichenvorrat, der die Zeichen von A enthält) mit den Grundoperationen *isempty*, *first* und *rest* zur Verfügung steht. Dabei ist für eine Zeichenreihe $x=x_1 x_2 ... x_n$:

   $$\textit{isempty}(x) = \text{true} \Leftrightarrow x=\varepsilon$$

   und, falls $x \neq \varepsilon$:

   $$\text{first}(x_1 x_2 \ldots x_n) = x_1$$
   $$\text{rest}(x_1 x_2 \ldots x_n) = x_2 \ldots x_n.$$

Für $n \in N_0$ sei nun M_n die Menge aller Zeichenreihen der Länge n aus M. Die Abbildung $h: N_0 \to N_0$ sei definiert als:

$$h(n) = \text{Anzahl der Elemente von } M_n.$$

5. Beweisen Sie, dass für h gilt:

$$h(n) = \begin{cases} 1 & \text{falls } n = 0 \\ 3^{n-1} + h(n-1) & \text{sonst} \end{cases}$$

   Hinweis: Es gibt genau 3^n Zeichenreihen der Länge n über A.

6. Formulieren Sie anhand von 5. einen rekursiven Algorithmus zur Berechnung von h(n) für gegebenes $n \varepsilon N_0$. Nehmen Sie dabei an, dass Addition, Subtraktion und Multiplikation, nicht jedoch die Potenzierung als arithmetische Grundoperationen zur Verfügung stehen. Bestimmen Sie (in Abhängigkeit von n) die Anzahl der Additionen, Subtraktionen und Multiplikationen. die gemäß diesem Algorithmus bei der Berechnung von h(n) durchgeführt werden.

7. Beweisen Sie, dass für $n > 0$ auch gilt:

   $h(n) = 3*h(n-1)-1.$

   Was bedeutet diese Beziehung im Hinblick auf die Komplexität der Berechnung von h gemäß 6. ?

8. Für die Entrekursivierung der Berechnung von h wird durch die Beziehung aus 7. eine Einbettung von h in die Abbildung g: $N_0^3 \to Z$ mit

   $g(n,k,m) = k \bullet h(n) - m$

   nahegelegt.

   Entwickeln Sie zunächst einen repetetiv rekursiven Algorithmus zur Berechnung von g(n,k.m) für gegebene $n,k,m \in N_0$ und daraus einen iterativen Algorithmus zur Berechnung von h(n) für gegebenes $n \in N_0$.

Aufgabe 2

Die Abbildungen $null : N_0 \to N_0$ und $succ : N_0 \to N_0$ seien definiert durch:

$$null(x) = 0,$$
$$succ(x) = x + 1.$$

Zu einer Abbildung $f : N_0 \to N_0$ seien $f^0 : N_0 \to N_0$ und $f^1 : N_0 \to N_0$ definiert durch:

$$f^0(0) = 0, \qquad f^0(x+1) = f(f^0(x)),$$
$$f^1(0) = 0, \qquad f^1(x+1) = f(f^1(x)).$$

Die Menge F von Abbildungen $N_0 \to N_0$ sei induktiv definiert wie folgt:

i) *null* und *succ* sind in F enthalten.

ii) Ist $f \in F$, so ist auch f^0 und f^1 in F enthalten.

iii) Sind $f, g \in F$, so ist auch h mit $h(x) = f(g(x))$ in F enthalten.

1. Bestimmen Sie $null^0, null^1, succ^0$ und $succ^1$ (in rekursionsfreier Darstellung).
2. Beweisen Sie, dass die Abbildungen

$$eins : N_0 \to N_0 \qquad eins(x) = 1$$
$$add2 : N_0 \to N_0 \qquad add2(x) = 2 + x$$
$$mult2 : N_0 \to N_0 \qquad mult2(x) = 2 * x$$
$$pot2 : N_0 \to N_0 \qquad pot2(x) = 2^x$$

   in F enthalten sind.

3. Beweisen Sie: Ist $f \in F$, so ist auch die Abbildung $null : N_0 \to N_0$ mit

$$r(x) = \begin{cases} 1, & \text{falls } 2^{f(x)} + 2 > 3 \\ 0 & \text{sonst} \end{cases}$$

   in F enthalten.

Zu einer Abbildung $h : N_0 \to N_0$ sei nun die Abbildung $ack_h : N_0 \to N_0$ („verallgemeinerte Ackermann-Funktion") definiert durch:

$$ack_h(0, y) = h(y)$$
$$ack_h(x+1, 0) = ack_h(x, 1)$$
$$ack_h(x+1, y+1) = ack_h(x, ack_h(x+1, y))$$

Die unendliche Folge $h_0, h_1, h_2, \ldots$ von Abbildungen $N_0 \to N_0$ sei gegeben durch:
$h_i(y) = ack_h(i, y)$ für $i = 0,1,2,\ldots$

4. Beweisen Sie: Falls $h(0) \neq 0$ und $h(y+1) > h(y)$ für alle $y \in N_0$, so gilt für alle $i \in N_0$ und $y \in N_0$:
   4.1 $h_i(y) > y$
   4.2 $h_i(y+1) > h_i(y)$
   4.3 $h_{i+1}(y) > h_i(y+1)$
   4.3 $h_{i+1}(y) > h_i(y)$

5. Beweisen Sie: Falls $h \in F$, so gilt $h \in F$ für alle $i \in N_0$.

6. Sei $h \in F$. Sind die Abbildungen h_i ($i \in N_0$) dann primitiv-rekursiv? Begründen Sie Ihre Antwort.

1.2 Frühjahr 1990

Aufgabe 1

Gegeben sei ein endliches Alphabet A und eine ungeordnete, endliche, nichtzyklische Liste von K Paaren (n, t) für ein vorgegebenes K ∈ IN, worin die n nichtleeren endlichen Zeichenreihen aus A*\{ε} und die t natürlichen Zahlen aus lN seien. In A* steht die

lexikographische Ordnung zur Verfügung, die zur Unterscheidung von der Ordnung < in IN mit $\subset$ bezeichnet werde. Außerdem gelte für alle n in den Paaren der Liste: $|n| \leq L$ für ein vorgegebenes $L \in IN$, wobei mit |x| die Länge einer Zeichenreihe x E A* bezeichnet wird.
Die Paare der Liste können als einfache Karteikarten in einer Telefondatei aufgefasst werden mit der Bedeutung:

n	:	Name
t	:	Telefonnummer.

Es wird vorausgesetzt, dass für verschiedene Paare (n_i, t_i) und (n_j, t_j) in der Liste gilt:
$n_i \neq n_j$ und $t_i \neq t_j$.

1. Geben Sie Datenstrukturen durch Typ- und Identitätsvereinbarungen an, mit denen die folgenden Teilaufgaben bearbeitet werden können.
2. Formulieren Sie einen Algorithmus, mit dessen Hilfe eine Zugriffsstruktur auf die Liste aufgebaut wird. Die Zugriffsstruktur soll es ermöglichen, zu einem Namen n mit der Komplexität O(log K)

2.1 zu entscheiden, ob die Liste einen Eintrag zu n enthält, und
2.2 gegebenenfalls die zugehörige Telefonnummer anzugeben

3. Schreiben Sie eine Prozedur *zugriff* in PASCAL, die den unter 2. formulierten Algorithmus realisiert.
4. Schreiben Sie eine Prozedur *suche*, die mit Hilfe der unter 3. aufgebauten Zugriffsstruktur zu einem $n \in A^*$ feststellt, ob die Liste einen Eintrag zu n enthält, und gegebenenfalls die zugehörige Telefonnummer t ausgibt. Die Komplexität der Prozedur *suche* soll O(log K) sein.

Aufgabe 2

Gegeben sei das Alphabet A = {a, b}. Mit x_a bzw. x_b werde für ein $x \in A^*$ die Zeichenreihe aus {a}* bzw. {b}* bezeichnet, die durch Streichen aller b bzw. a aus x entsteht. Seien also z.B. x = aabab und y = bbbb, dann ist x_a = aaa, x_b = bb, $y_a = \varepsilon$ und y_b = bbbb. Gegeben sei nun die wie folgt definierte Teilmenge M von A*:

$$x \in M \Leftrightarrow_{df} |x_a| \leq |x_b|$$

wobei für ein $x \in A^*$ mit |x| die Länge von x bezeichnet wird.

1. Zeigen Sie, dass die Menge $M \subset A^*$ nicht regulär ist.
2. M ist als Sprachschatz einer kontextfreien Sprache über dem terminalen Alphabet A darstellbar. Beweisen Sie diese Aussage dadurch, dass Sie einen Kellerautomaten angeben, von dem Sie zeigen, dass er genau die Menge M akzeptiert.
3. Konstruieren Sie eine kontextfreie Grammatik über dem terminalen Alphabet A, die in A* genau die Menge M erzeugt, und begründen Sie die einzelnen Schritte Ihres konstruktiven Vorgehens.

Aufgabe 3

Gegeben seien zwei ganze Zahlen p und q mit 0 < q < p und eine wie folgt definierte rekursive Rechenvorschrift f für ganze Zahlen $z \in Z$:

$$f(z) = \begin{cases} f(f(z-p)) & \text{für } z \geq 100 \\ z+q & \text{für } z < 100 \end{cases}$$

Beweisen Sie, dass die Rechenvorschrift f für alle $z \in Z$ terminiert und somit eine Funktion

$$f: Z \rightarrow Z$$

definiert.

Hinweis:
Betrachten Sie für $z \geq 100$ die durch f(z) veranlassten rekursiven Aufrufe $f(z_i)$ von f und zeigen Sie, dass für alle i gilt: $z_i < z$.

1.3 Herbst 1990

Aufgabe 1

Gegeben sei das Alphabet A = {a, b, c}. Die Mengen M_a, M_b, M_c und M von Zeichenreihen über A seien definiert durch

$M_x = \{w \in A^* \mid w = uxxv \text{ mit } u,v \in A^*\}$ für x = a, b bzw. c,
$M = M_a \cup M_b \cup M_c$.

1. Beschreiben Sie M durch einen regulären Ausdruck!
2. Geben Sie einen deterministischen endlichen Automaten an, der genau die Zeichenreihen von M akzeptiert!
3. Geben Sie eine reguläre Grammatik an, die M als Sprachschatz hat!

Aufgabe 2

Gegeben sei die Grammatik Γ mit {**a,b**} als Menge der Terminalzeichen, den Nichtterminalzeichen **Z,A,B**, dem Axiom **Z** und den Produktionsregeln

Z ® AB
A ® ZA **B ® ZB**
A ® a **B ® b**

1. Zeigen Sie, dass die Zeichenreihe **aabbabab** zum Sprachschatz von Γ gehört!
2. Überführen Sie Γ in die Greibach-Normalform!
3. Geben Sie einen (gegebenenfalls nicht-deterministischen) Kellerautomaten an, der genau den Sprachschatz von Γ akzeptiert!

Aufgabe 3

Durch die Funktionsvereinbarung

```
function h(m,n:nat)nat:
     if m=0 then n else 2*h(m-1,n) endif
```

ist eine Funktion h: $N_0^2 \rightarrow N_0$, definiert.

Beweisen Sie

1. durch Berechnungsinduktion
2. durch Parameterinduktion (nach m)

dass für alle $m,n \in N_0$ gilt:

```
h(m,2*n) = 2*h(m,n)
```

Aufgabe 4

Durch die Funktionsvereinbarung

```
function f(n:nat)nat:
     if n ≤ 2 then n else 2*f(n-1)+f(n-3) endif
```

ist eine Funktion f: N_0 ® N_0 definiert.

1. Beweisen Sie, dass f für alle $n \in N_0$ terminiert!
2. Die Funktion g: N_0^4 ® N_0 sei gegeben durch

   g(n,x,y,z) = x*f(n+2) + y*f(n+1) + z*f(n).

   Beweisen Sie

$$g(n,x,y,z) = \begin{cases} 2*x+y & \text{falls } n = 0 \\ g(n-1,2*x+y,z,x) & \text{falls } n > 0 \end{cases}$$

3. Entwickeln Sie mit Hilfe von 2. zunächst einen repetetiv rekursiven Algorithmus zur Berechnung von g(n,x,y,z) für gegebene $n,x,y,z \in N_0$ und daraus einen iterativen Algorithmus zur Berechnung von f(n) für gegebenes $n \in N_0$!

   Hinweis:
   Formulieren Sie die Algorithmen in einer Programmiersprache wie PASCAL, MODULA o.ä. oder in einem "Pseudocode", wie er in obiger Funktionsvereinbarung verwendet ist!

Aufgabe 5

Für $r \in R$ bezeichne [r] die (eindeutig bestimmte) ganze Zahl z mit $z \le r < z+1$. Die Funktion f:$N_0 \rightarrow N_0$ sei gegeben durch

f(x) = [3 * sqrt(x)].

Beweisen Sie, dass f primitiv-rekursiv ist!

Hinweis:

Die üblichen arithmetischen und Booleschen Operationen wie $+,*,\leq,<,\wedge,\vee$ u.ä. dürfen als primitiv-rekursiv vorausgesetzt werden.

Aufgabe 6

A und B seien zwei rekursiv aufzählbare Teilmengen von N_0 mit $A \cup B = N_0$.

Beweisen Sie: Falls $A \cap B$ rekursiv ist, so sind A und B rekursiv!

Hinweis: Für $M,N \subseteq N_0$ gilt:

1. M ist genau dann rekursiv, wenn M und $N_0 \backslash M$ rekursiv aufzählbar ist.
2. Falls M und N rekursiv aufzählbar sind, so ist $M \cap N$ rekursiv aufzählbar.

1.4 Herbst 1991

Aufgabe 1

Gegeben seien das Alphabet A = {a,b} sowie folgende Mengen M_1 und M_2:

M_1 = Menge aller Zeichenreihen über A, die mindestens ein Paar aufeinanderfolgender Zeichen a enthalten,

M_2 = Menge aller Zeichenreihen über A, die höchstens ein Paar aufeinanderfolgender gleicher Zeichen enthalten.

1. Geben Sie ien reguläre Grammatik an, die M_1 als Sprachschatz hat.
2. Geben Sie einen deterministischen endlichen Automaten an, der genau die Zeichenreihen von M_2 akzeptiert!
3. Geben Sie einen regulären Ausdruck an, der eine Menge L von Zeichenreihen über A beschreibt, für die gilt:
   $$M_1 = L\ A^*.$$
   Beweisen Sie Ihre Behauptung!
4. Beweisen oder widerlegen Sie: Es gibt eine Menge N von Zeichenreihen über A mit
   $$M_2 = N\ A^*.$$

Aufgabe 2

M sei die Menge aller Zeichenreihen w über dem Alphabet {0,1} mit der Eigenschaft, dass w doppelt so viele Zeichen 1 wie 0 enthält.

Geben Sie eine Turingmaschine an, die genau die Menge M akzeptiert!

Aufgabe 3

Hinweis:
Verwenden Sie zur Beschreibung der in dieser Aufgabe zu entwickelnden Algorithmen eine Syntax, wie sie in höheren Programmiersprachen wie PASCAL, MODULA o.ä. üblich ist.

Für die Menge bbchar aller Binärbäume über einer Grundmenge char von Zeichen seien als Grundoperationen verfügbar:

empty: → bbchar	empty = leerer Binärbaum
isempty: bbchar → boolean	isempty(b) = true ⇔ b ist leer
root: bbchar → char	root(b) = Wurzel von b, falls b ≠ empty
left: bbchar → bbchar	left(b) = linker Unterbaum von b, falls b ≠ empty
right: bbchar → bbchar	right(b) = rechter Unterbaum von b, falls b ≠ empty

(Für b=empty ist root(b), left(b), right(b) jeweils undefiniert.)

1. Definieren Sie mit Hilfe dieser Grundoperationen rekursiv die folgenden weiteren Operationen (wobei diese Definitionen gegebenenfalls auf weitere geeignet definierte Operationen abgestützt werden können):

1.1 bbgleich: bbchar x bbchar → boolean	bbgleich(b_1, b_2) = true ⇔ b_1 und b_2 sind gleich,
1.2 istord: bbchar → boolean	istord(b) = true ⇔ b ist geordnet (sortiert)
1.3 istvoll: bbchar → boolean	istvoll(b) = true ⇔ b ist vollständig

2. Die Operation enthalten: bbchar x char → boolean mit
   enthalten (b,x) = true ⇔ x ist als Knoten in b enthalten
   kann rekursiv wie folgt definiert werden:

```
enthalten(b,x) =   if isempty(b) then false
                   else x = root(b) ∨
                            enthalten(left(b),x) ∨
                            enthalten(right(b),x)
                   endif
```

   Geben Sie – unter Verwendung einer geegnet gewählten Datenstruktur keller (für Kellerspeicher) – einen iterativen Algorithmus an, der enthalten(b,x) für gegebene b und x berechnet!

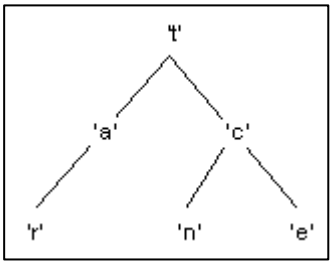
3. Unter der Voraussetzung, dass b geordnet (sortiert) ist, lässt sich enthalten linear rekursiv definieren.
   Geben Sie diese Definition und einen entsprechenden iterativen Algorithmus (ohne Verwendung eines Kellers) zur Berechnung an!

4. Binärbäume seien nun in üblicher Weise durch Geflechte realisiert, in PASCAL-Notation etwa:

```
TYPE bbchar = ↑bbelem;
     bbelem = RECORD
                wurzel : char;
                lub : bbchar;  (* linker Unterbaum *)
                rub : bbchar   (* rechter Unterbaum *)
              END;
```

   Geben Sie Algorithmen zur Realisierung der Operationen isempty, root und left gemäß dieser Darstellung an!

5. Geben Sie einen Algorithmus an, der den Binärbaum



in der Darstellung von Teilaufgabe 4 erzeugt!

Geben Sie dazu zunächst einen Algorithmus für die Operation

compose: char x bbchar x bbchar → bbchar,
compose (x, b_1, b_2) = Binärbaum mit Wurzel x, linkem Unterbaum b_1 und rechtem Unterbaum b_2

an und verwenden Sie diesen zum Aufbau des Binärbaums!

Aufgabe 4

Durch die Funktionsvereinbarung

```
function f(x,y,z:nat)nat:
     if x=y then z else f(x,y+1,(y+1)*z) endif
```

ist eine Funktion f : $N_0^3 \rightarrow N_0$ gegeben.

1. Bestimmen Sie den Wert von f (4,0,2) !
2. Beweisen Sie: f (x,y,z) terminiert für alle x,y,z $\in N_0$ mit x ≥ y !

1.5 Frühjahr 1993

Aufgabe 1 Formale Sprachen

1. Sei G = (T, N, P, S) eine Chomsky-Grammatik. Was bedeuten die angegebenen Komponenten? Wie sind die Produktionsmenge P und der Ableitbarkeitsbegriff definiert?
2. Welche formalen Einschränkungen auf P führen zu den monotonen, kontextsensitiven und kontextfreien Grammatiken? Wie liegen die Klassen der erzeugten Wortmengen zueinander? Soweit sie nicht gleich sind, geben Sie (ohne Beweis) je ein typisches Beispiel aus der Differenzmenge an!
3. Für kontextfreie Grammatiken gilt das Zerlegungslemma. Formulieren und beweisen Sie dieses Lemma!
4. Beweisen Sie, dass sich die Wortmenge
   $\{a^i b^j \mid i,j \geq 0 \wedge i \neq j\}$
   von einer kontextfreien Grammatik erzeugen lässt!

Aufgabe 2 Berechenbarkeit

1. Der Berechenbarkeitsbegriff kann nach Kleene durch rekursive Funktionen definiert werden. Geben Sie die Definition der primitiv-rekursiven und der μ–rekrsiven Funktionen an!
2. Zeigen Sie, dass der beschränkte μ-Operator nicht aus dem Bereich der primitiv-rekursiven Funktionen hinausführt!
3. Beweisen Sie, dass die ganzzahlige Quadratwurzelfunktion $\lfloor \sqrt{n} \rfloor$ primitiv-rekursiv ist!
4. Die μ-rekursiven Funktionen lassen sich unmittelbar in while-Programme übersetzen, die nur aus
   - Wertzuweisungen,
   - dem Aufruf der Nachfolger- und Vorgängerfunktion,
   - while-Schleifen und
   - Prozeduraufrufen

   bestehen. Zeigen Sie, dass es kein while-Programm mit genau einer Variablen gibt, das die Funktion f(x) = 2x berechnet!

Aufgabe 3 Algorithmische Sprachen

1. Eine Objektart ist durch eine Trägermenge und einen Satz Operationen definiert. Wie entsteht die Trägermenge bei einer Verbundart und welche Operationen sind darauf definiert? Verwenden Sie als Beispiel

```
mode datum = ( int[1..31] tag,
               int[1..12] monat,
               int[1900..1999] jahr)
```

2. Definieren Sie die Objektart der binären Bäume als rekursive Verbundart! Mit welchem Konzept realisiert man rekursive Verbundarten in den gängigen Programmiersprachen (PASCAL, C)?
3. Beschreiben Sie eine Technik zur Umwandlung beliebiger Bäume in binäre! Zeigen Sie durch eine Plausibilitätsbetrachtung, dass das Durchlaufen des Baumes in Prä-Ordnung von dieser Umwandlung nicht berührt wird!
4. Welche Konstruktionsvorschrift liegt den Artvarianten zugrunde? Erläutern Sie Ihre Antwort an einem geeigneten Beispiel! Zeigen Sie, dass die unmittelbare Verwendung des zugrundeliegenden mathematischen Konzeptes eine statische Typprüfung nicht zulässt!

Aufgabe 4 Programmiermethodik

Betrachten Sie folgende Spezifikation eines abstrakten Datentyps:

```
init:                    → type
add:      type x nat     → type
remove:   type           → type
is_empty: type           → boolean
first:    type           → nat
is_empty(init)           = true
is_empty(add(t,n))       = false
first(add(init,n))       = n
remove(add(init,n))      = init
first(add(add(t,n),m))   = first(add(t,n))
remove(add(add(t,n),m)) = add(remove(add(t,n)),m)
```

1. Erläutern Sie unter Verwendung dieses Beispiels, was die wesentliche Idee der algebraischen Spezifikation ist!
2. Die angegebene Spezifikation definiert einen in der Informatik häufig anzutreffenden Datentyp. Welcher Datentyp ist das? Begründung!
3. Das angegebene Gleichungssystem induziert auf der Termalgebra eine Kongruenzrelation. Geben Sie deren formale Definition an!
4. Zeigen Sie, dass es in jeder der durch 3. definierten Kongruenzklassen genau einen Term gibt, der entweder die Operation `remove` überhaupt nicht oder nur in der Form `remove(init)` enthält!

Aufgabe 5 Übersetzerbau

1. Was versteht man uner einem Compilergenerator? Erläutern Sie Aufgabe, Aufbau und Datenfluss!
2. Als Eingabe für einen Compilergenerator reicht eine kontextfreie Grammatik nicht aus. Wie werden die nichtkontextfreien Aspekte der Syntax und die Semantik beim Compilergenerator *Yacc* berücksichtigt?
3. Betrachten Sie folgende Grammatik für Schleifen:

```
WHILE_stat  ::= WHILE b LOOP seq_of_stat END;
seq_of_stat ::= statement | seq_of_stat statement
statement   ::= WHILE_stat | other_stat;
```

   Verändern und/oder ergänzen Sie die erste dieser Produktionen, so dass *Yacc* die Sprunganweisungen, die zur Realisierung einer Schleife nötig sind, an der richtigen Stelle erzeugt. Begründen Sie Ihre Änderung!
4. Bei Erreichen eines Sytaxfehlers kommt es darauf an, die Syntaxanalyse so fortzusetzen, dass möglichst wenige Folgefehler auftreten. Beschreiben Sie das auf Graham und Rhodes zurückgehende Verfahren zum Wiederaufsetzen! Ist das Verfahren im *Yacc* einsetzbar? (Hinweis: Bei der Beschreibung spielen die folgenden Begriffe eine Rolle: Kondensierung, Rückwärts-, Vorwärtsschritte.)

1.6 Herbst 1993

Hinweis:
Verwenden Sie zur Formulierung von Algorithmen eine Programmiersprache wie PASCAL, MODULA o.ä. oder einen "Pseudocode", wie er in einschlägigen Vorlesungen und Büchern üblicherweise benutzt wird!

Aufgabe 1

Gegeben sei die Grammatik Γ mit Σ = {a,b} als Menge der Terminalzeichen, den Nichtterminalzeichen Z, A und B, dem Axiom Z und den Produktionsregeln:

$Z \rightarrow a$	$A \rightarrow ab$	$B \rightarrow ba$
$Z \rightarrow aB$	$A \rightarrow aBb$	
$Z \rightarrow Aa$	$A \rightarrow abA$	

1. Beweisen Sie: Γ ist mehrdeutig.
2. Beweisen Sie: Für den Sprachschatz $L(\Gamma)$ von Γ gilt:
   $L(\Gamma) = \{a(ba)^n : n \in N_0\}$.
3. Geben Sie eine reguläre Grammatik an, die den gleichen Sprachschatz hat wie Γ.
4. Geben Sie einen deterministischen endlichen Automaten an, der genau die Zeichenreihen von $L(\Gamma)$ akzeptiert!

Aufgabe 2

Gegeben sei das Alphabet $\Sigma = \{a,b\}$. Für eine Zeichenreihe $w \in \Sigma^*$ bezeichne A(w) die Anzahl der Zeichen a in w und B(w) die Anzahl der Zeichen b in w.
Die Menge $M \subseteq \Sigma^*$ sei definiert durch

$M = \{w \in \Sigma^* : A(w)$ ist gerade, und $B(w)$ ist ungerade$\}$.

Beweisen Sie:
M ist entscheidbar: Geben Sie, dazu eine Turing-Maschine an, die für alle $x \in \Sigma^*$ anhält und genau alle $x \in M$ akzeptiert!

Aufgabe 3

Gegeben sei eine Funktion G: $N_0 \to N_0$ mit $G(x) \le x$ für alle $x \in N_0$.
Durch die Funktionsvereinbarung

```
function F(x:nat)nat:
     if x mod 2 = 0 then G(x) else F(F(x-1)) endif
```

ist eine Funktion F:$N_0 \to N_0$ definiert.

1. Beweisen Sie:

1.1 F(x) terminiert für alle $x \in N_0$.
1.2 Falls $G(x) = x$ für alle $x \in N_0$, so gilt $F(F(x)) = F(x)$ für alle $x \in N_0$.

2. Die Funktion F*: $N_0 \times N_0 \to N_0$ sei wie folgt definiert:

$$F^*(x,y) = \begin{cases} x & falls\ \ y = 0 \\ F^*(F(x), y-1) & sonst \end{cases}$$

2.1 Beweisen Sie: Für alle $x \in N_0$ gilt $F(x) = F^*(x,1)$.
2.2 Geben Sie eine repetetiv rekursive Funktionsvereinbarung für F* an (die sich nicht auf F abstützt), und entwickeln Sie daraus durch Entrekursivierung und Spezialisierung (gemäß Teilaufgabe 2.1) einen iterativen Algorithmus zur Berechnung von F.

Aufgabe 4

Gegeben seien folgende Produktionsregeln (in Backus-Naur-Form) für die Syntaxdefinition von Gleitpunktzahlen (über dem Alphabet {+,-,.,E,0,1,2,3,4,5,6,7,8,9}):

<Gleitpunktzahl> → <Vorzeichen><nichtnegative Zahl>|<nichtnegative Zahl>
<nichtnegative Zahl> → <Mantisse><Exponent>
<Mantisse> → <Ziffernfolge>.<Ziffer><Ziffernfolge>

<Exponent> → E<ganze Zahl>| ε
<ganze Zahl> → <Vorzeichen><Ziffer><Ziffernfolge>|<Ziffer><Ziffernfolge>
<Ziffernfolge> → <Ziffer><Ziffernfolge>| ε
<Vorzeichen> → + | -
<Ziffer> → 0|1|2|3|4|5|6|7|8|9

1. Geben Sie für die Gleitpunktzahl `4.538E-2`
   den Syntaxbaum gemäß dieser Definition an!

2. Geben Sie einen Syntaxanalyse-Algorithmus nach der Methode des rekursiven Abstiegs („recursive descent“) an, der feststellt, ob eine vorgelegte Zeichenreihe eine Gleitpunktzahl gemäß dieser Definition ist!

1.7 Frühjahr 1994

Aufgabe1 Formale Sprachen / Automatentheorie

1. Geben Sie eine kontextfreie Grammatik zu der Wortmenge

   $$L = \{a^i b^j \mid i \geq j \geq 0\}$$

   an.

2. Eine kontextfreie Grammatik G heißt reduziert, wenn jedes nichtterminale Symbol von G in einer aus dem Startsymbol ableitbaren Kette vorkommt und aus jedem nichtterminalen Symbol eine terminale Kette ableitbar ist. Geben Sie ein Verfahren an, das zu jeder kontextfreien Grammatik eine äquivalente reduzierte Grammatik liefert.

3. Beweisen Sie, dass die nichtdeterministischen, erkennenden endlichen Automaten nicht mehr können als die deterministischen.

4. Konstruieren Sie einen deterministischen, erkennenden endlichen Automaten, der die Wortmenge über dem Alphabet {a, b} akzeptiert, die aus allen Wörtern besteht, die jede der folgenden Bedingungen erfüllen:

   i. Die Länge des Wortes ist durch 3 teilbar.
   ii. Das Wort beginnt mit a und endet mit b.
   iii. aaa ist kein Teilwort des Wortes.

Aufgabe 2 Berechenbarkeit / Algorithmische Sprachen

1. While-Programme bestehen aus Anweisungen der Form;

```
X := 0;
X := succ(Y);
X := pred(Y);
while X /= Y do Folge-von-Anweisungen end;
```

wobei X und Y beliebige Variablen, *succ* die Nachfolgerfunktion und *pred* die Vorgängerfunktion bezeichnet. Schreiben Sie ein While-Programm, das die Addition von n und m realisiert.

2. Was versteht man unter einer primitiv-rekursiven Funktion? Zeigen Sie, dass man jede primitiv-rekursive Funktion durch ein While-Programm realisieren kann, das stets hält.

3. Wir nennen eine reelle Zahl *a näherungsweise berechenbar,* wenn es eine berechenbare Funktion f (n) gibt, die für jeden Parameter *n* die ersten *n* Stellen der Dezimalentwicklung von α liefert. Zeigen Sie, dass die Quadratwurzel jeder natürlichen Zahl näherungsweise berechenbar ist. (Die Berechenbarkeit der arithmetischen Grundoperationen sei bekannt.)

4. Zeigen Sie, dass es reelle Zahlen gibt, die *nicht* näherungsweise berechenbar sind.

Aufgabe 3 Programmiermethodik

1. Erläutern Sie die auf Floyd und Hoare zurückgehende Verifikationsmethode. (In Ihrer Antwort müssen mindestens die Begriffe Zusicherung, schwächste Vorbedingung und Prädikattransformation vorkommen.)

2. Betrachten Sie folgendes Programmfragment mit der Nachbedingung $a = n^3$:

```
a:=0; b:=1; c:=0;
WHILE c<6*n DO
     a := a + b;
     c := c + 6;
     b := b + c;
END;
```

Geben Sie die Schleifeninvariante an, und beweisen Sie diese durch Induktion. Beweisen Sie als zweites die Nachbedingung. Was fehlt dann noch für eine vollständige Verifikation?

Aufgabe 4 Übersetzerbau

1. Welches sind die drei wichtigsten Teilaufgaben, die ein Codegenerator in *jedem Fall zu* lösen hat? (Hinweis: Die Optimierung gehört nicht dazu.) Beschreiben Sie diese Aufgaben.

2. Generieren Sie Maschinencode zu den beiden folgenden C-Anweisungen.

```
x = a/(b+c) - d*(e+f);
a[i][j] = b[i][k] * c[k][j];
```

Gehen Sie dabei von einer Zielmaschine mit drei universell verwendbaren Registern aus.

3. Erläutern Sie die folgenden Parameterübergabemechanismen:
Call-by-value, Callby-reference, Call-by-name und Copy-Restore.

4. Beim Aufruf einer Prozedur muss eine Umgebungsbeschreibung angelegt werden (activation record). Welche Information muss diese Beschreibung bei pascalähnlichen Sprachen enthalten? Inwiefern vereinfacht sie sich bei Sprachen, die keine geschachtelten Prozedurdeklarationen zulassen?

1.8 Herbst 1994

Aufgabe 1

M sei die Menge aller Zeichenreihen über dem Alphabet {0,1}, die mindestens so viele Zeichen 1 wie Zeichen 0 enthalten.

Geben Sie eine (deterministische) Turingmaschine T an, die außer einem Leerzeichen nur die Zeichen aus {0,1} verwendet und *M* in folgendem Sinne akzeptiert:
Ein Wort $w \in \{0,1\}^*$ steht auf dem ansonsten leeren Band! Angesetzt auf das erste Zeichen von w (bzw. auf ein Leerzeichen, falls w das leere Wort ist), erreicht T genau dann nach endlich vielen Schritten einen Endzustand, wenn $w \in M$ ist.

Aufgabe 2

Gegeben sei die Grammatik Γ_1 mit {a,b} als Menge der Terminalzeichen, den Nichtterminalzeichen Z,A,B, dem Axiom Z und den Produktionsregeln

$Z \rightarrow aA$	$B \rightarrow a$
$A \rightarrow b$	$B \rightarrow b$
$A \rightarrow bB$	$B \rightarrow bB$

Die Grammatik Γ_2 entstehe aus Γ_1 dadurch, dass zu diesen Produktionsregeln noch die weitere Regel

$$B \rightarrow BA$$

hinzugenommen wird.

Der jeweilige Sprachschatz von Γ_1 und Γ_2 sei mit $L(\Gamma_1)$ bzw. $L(\Gamma_1)$ bezeichnet.

1. Beweisen Sie: $L(\Gamma_1) = \{ab^n \mid n \in N\} \cup \{ab^na \mid n \in N\}$.
2. Geben Sie einen deterministischen endlichen Automaten an, der genau die Zeichenreihen von $L(\Gamma_1)$ akzeptiert.
3. Beweisen Sie: Γ_2 ist mehrdeutig.
4. Beweisen Sie: $L(\Gamma_1) \neq L(\Gamma_2)$.
5. Überführen Sie Γ_2 in Greibach-Normalform.

Aufgabe 3

Sei int die Menge der ganzen Zahlen und **sequ** int die Menge aller endlichen Folgen ganzer Zahlen. Für **sequ** int seien als Grundoperationen verfügbar:

empty: $\rightarrow$ **sequ** int	empty = leere Folge
isempty: **sequ** int $\rightarrow$ boolean	isempty(s) = true $\Leftrightarrow$ s ist leer

first: **sequ** int → int	first: $(s_1, \ldots, s_n) \rightarrow s_1$
rest: **sequ** int → **sequ** int	rest: $(s_1, s_2, ..., s_n) \rightarrow (s_2, \ldots, s_n)$
prefix: int x **sequ** int → **sequ** int	prefix: $(x, (s_1, ..., s_n)) \rightarrow (x, s_1, \ldots, s_n)$

(Für s = *empty* sind *first(s)* und *rest(s)* nicht definiert.)

S_3 sei die Menge aller Folgen aus **sequ** int mit einer durch 3 teilbaren Anzahl von Komponenten.

1. Geben Sie (unter ausschließlicher Verwendung der genannten Grundoperationen) rekursive Funktionsvereinbarungen an für Funktionen *last, lead, postfix, conc* mit folgender Bedeutung *(last(s)* und *lead(s)* sind nur für s ≠ e*mpty* definiert):

last: **sequ** int → int	last: $(s_1, \ldots, s_n) \rightarrow s_n$
lead: **sequ** int → **sequ** int	lead: $(s_1, \ldots, s_{n-1}, s_n) \rightarrow (s_1, \ldots, s_{n-1})$
postfix: **sequ** int x int → **sequ** int	postfix: $((s_1, ..., s_n), x) \rightarrow (s_1, \ldots, s_n, x)$
conc: **sequ** int x **sequ** int → **sequ** int	
	conc: $((s_1, ..., s_n)(t_1, ..., t_m)) \rightarrow (s_1, ..., s_n, t_1, ..., t_m)$

2. Gegeben sei die Funktion *vorn* durch die Funktionsvereinbarung

```
function vorn(s:sequ int)sequ int:
     (* definiert nur für s ∈ S3 *)
     if isempty(s) then s
     else prefix(first(s),vorn(rest(lead(s)))) endif
```

   Beweisen Sie: Für $s \in S_3$ ist *vorn(s)* das "vordere Drittel von s", d.h. für $s = (s_1,...,s_{3k})$, $k \in N_0$, ist *vorn(s)* = $(s_1,...,s_k)$.

3. Geben Sie analog zur Funktion vorn aus Teilaufgabe 2. eine rekursive Funktionsvereinbarung für eine Funktion *hinten* an, die nur die genannten Grundoperationen und Funktionen aus Teilaufgabe 1. verwendet und die für $s \in S_3$ das hintere Drittel von s berechnet (d.h. *hinten(s)* = $(s_{2k+1},...,s_{3k})$ für $s = (s_1,...,s_{3k})$, $k \in N_0$).

4. Geben Sie eine rekursive Funktionsvereinbarung für eine Funktion *mitte* an, die außer den genannten Grundoperationen und den Funktionen aus Teilaufgabe 1. ausschließlich die Funktion *vorn* aus Teilaufgabe 2. verwenden darf und die für $s \in S_3$ das mittlere Drittel von s berechnet (d.h. *mitte(s)* = $(s_{k+1},...,s_{2k})$ für $s = (s_1,...,s_{3k})$, $k \in N_0$).

5. Die rekursive Funktionsvereinbarung von *vorn* in Teilaufgabe 2. soll in systematischer Weise in einen iterativen Algorithmus (mit gleicher Wirkung) überführt werden (der ebenfalls nur die angegebenen Grundoperationen und Funktionen, aus Teilaufgabe 1. verwendet). Betten Sie dazu *vorn* in einen geeigneten allgemeineren repetitiv rekursiven Algorithmus ein, und entrekursivieren und spezialisieren Sie diesen zu dem gesuchten iterativen Algorithmus.

6. Objekte aus **sequ** int können in Programmiersprachen wie PASCAL, MODULA o.ä. als lineare Listen realisiert werden. Geben Sie (in einer derartigen

Programmiersprache) entsprechende Typvereinbarungen und Algorithmen zur Realisierung der angegebenen Grundoperationen an.

7. Geben Sie (in PASCAL, MODULA o.ä.) einen iterativen Algorithmus *laenge* an, der unter Verwendung der Realisierungen der Grundoperationen gemäß Teilaufgabe 6. die Anzahl der Komponenten einer als lineare Liste realisierten Folge aus **sequ** int berechnet.

Aufgabe 4

Durch die Funktionsvereinbarung

```
function f(x,y,z:nat)nat:
    if x=y+1 then z+y
    else f(x+y+1,2*(y+1),z+y+l) endif
```

ist eine Funktion f : $N_0^3 \rightarrow N_0$ definiert.

1. Bestimmen Sie f (6,0,1).
2. Beweisen Sie: f (x,y,z) terminiert für alle x,y,z $\in N_0$ mit x > y.
3. Beweisen Sie, dass für alle x,y,z $\in N_0$ mit x > y gilt:
   f (x,y,z) ist genau dann eine gerade Zahl, wenn x + z ungerade ist.

1.9 Frühjahr 1995

Aufgabe 1

Gegeben sei die Grammatik Γ_1 mit {a,b} als Menge der Terminalzeichen, den Nichtterminalzeichen Z,A,B, dem Axiom Z und den Produktionsregeln

$Z \rightarrow aB$	$A \rightarrow a$	$B \rightarrow b$
$Z \rightarrow bA$	$A \rightarrow aZ$	$B \rightarrow bZ$

Die Grammatik Γ_2 entstehe aus Γ_1 dadurch, dass zu diesen Produktionsregeln noch die weiteren Regeln

$A \rightarrow bAA$
$B \rightarrow aBB$

hinzugenommen werden.

Der jeweilige Sprachschatz von Γ_1 und Γ_2 sei mit $L(\Gamma_1)$ bzw. $L(\Gamma_2)$ bezeichnet. ε bezeichne das leere Wort.

1. Beweisen Sie: $L(\Gamma_1) = \{ab,ba\}^* \setminus \{\varepsilon\}$.

2. Konstruieren Sie direkt aus Γ_1 einen nicht-deterministischen endlichen Automaten, der genau die Zeichenreihen von $L(\Gamma_1)$ akzeptiert. Konstruieren Sie dann aus diesem

Automaten einen deterministischen endlichen Automaten, der genau die Zeichenreihen von $L(\Gamma_1)$ akzeptiert.

3. Geben Sie eine (deterministische) Turingmaschine T an, die außer einem Leerzeichen nur die Zeichen aus {a,b} verwendet und $L(\Gamma_1)$ in folgendem Sinne akzeptiert:
   Ein Wort w ∈ (a,b)* steht auf dem ansonsten leeren Band.
   Angesetzt auf das erste Zeichen von w (bzw. auf ein Leerzeichen, falls w das leere Wort ist), erreicht T genau dann nach endlich vielen Schritten einen Endzustand, wenn $w \in L(\Gamma_1)$ ist.

4. Beweisen Sie: aaabbabbba $\in L(\Gamma_2)$

5. Für ein w ∈ {a,b}* entstehe $\overline{w}$ aus w, indem man jedes a in w durch b und jedes b in w durch a ersetzt.
   Beweisen Sie: Ist $w \in L(\Gamma_2)$, so ist auch $\overline{w} \in L(\Gamma_2)$.

6. Überführen Sie Γ_2 in Chomsky-Normalform.

Aufgabe 2

NAT bezeichne den abstrakten Datentyp mit der Sorte nat der natürlichen Zahlen (einschließlich 0) und den üblichen Operationen auf nat. Der abstrakte Datentyp VEKTOR sei wie folgt definiert:

```
abstract type VEKTOR
uses NAT                (* Alles, was NAT enthält, darf verwendet werden *)
sorts vektor, Index     (* Die Sorten von VEKTOR *)
functions     null: → vektor,
              proj: vektor x index → nat,
              succ: vektor x index → vektor
axioms ∀ x ∈ vektor ∀ i,j ∈ index:
              proj(null,i) = 0,
              proj(succ(x,i),i) = proj(x,i) + 1,
              proj(succ(x,i),j) = proj(x,j) für i≠j
endoftype
```

1. Die Funktion f : nat x Index → vektor sei gegeben durch die Funktionsvereinbarung

```
function f(k:nat,i:index) vektor:
if k=0 then null else succ(f(k-1,i),i) endif
```

   Beweisen Sie unter Verwendung der Axiome von VEKTOR, dass für alle $k \in$ nat und $i, j \in$ index gilt:

1.1 *proj (f(k,i) ,i) = k*
1.2 *proj (f(k,i), j) = 0* für $i \neq j$

2. Geben Sie in Analogie zur Funktion f in Teilaufgabe 1. eine rekursive Funktionsvereinbarung für eine Funktion g: vektor x nat x index → vektor an, für die für alle $k \in$ nat und $i, j \in$ index gilt:

2.1 *proj (g(x,k,i), i) = proj (x,i) + k*
2.2 *proj (g(x,k,i), j) = proj (x, j) für i≠* j

Beweisen Sie 2.1 und 2.2 für Ihre Lösung.

3. Für ein fest vorgegebenes $n \in N$ sei nun index = {i | i ∈ N und 1 ≤ i ≤ *n*} und vektor = $\{(x_1,...,x_n) \mid x_i \in N_0$ für 1 ≤ i ≤ n}. Die Funktionen *null, proj* und *succ* seien gegeben durch

   null = (0,0, ...,0) ("Nullvektor"),
   $proj((x_1,...,x_n),i) = x_i$,
   $succ((x_1,...,x_n),i) = (x_1,...,x_{i-1},x_i+1, x_{i+1},...,x_n)$

1.1 Zeigen Sie, dass die so definierten Funktionen die Axiome von VEKTOR erfüllen.
1.2 Objekte der Sorte vektor können in höheren Programmiersprachen als Reihungen der Länge *n* realisiert werden (Typbezeichnung etwa **array**[1..n]**of** nat). Geben Sie (in der Notation einer derartigen Programmiersprache) Algorithmen zur Realisierung der Funktionen *null, proj* und *succ* an.
1.3 Geben Sie (in einer Notation wie in Teilaufgabe 3.2 einen Algorithmus an, der für $(x_1,...,x_n),(y_1,...,y_n) \in$ vektor den "Summenvektor" $(x_1+y_1,...,x_n+y_n)$ berechnet und dabei nur die Funktionen *null, proj* und *succ* verwendet. Erläutern Sie die wesentlichen Schritte des Algorithmus durch geeignete Kommentare.

Aufgabe 3

Durch die Funktionsvereinbarung

```
function f(x,y,z:nat)nat:
     if z = 0 then x+y
     else if y = 0 then 1
     else f(f(x,y-1,z),x,z-1) endif
endif
```

ist eine Funktion f : $N_0^3 \rightarrow N_0$ definiert. Ferner sei die Funktion g: $N_0 \times N_0 \rightarrow N_0$ mit

$$g(x, y) = \sum_{k=0}^{y} x^k$$

gegeben.

1. Bestimmen Sie f (3,4,1).
2. Beweisen Sie: f (x,y,z) terminiert für alle x,y,z ∈ N_0.
3. Geben Sie (in der Notation einer höheren Programmiersprache) unter Verwendung eines Kellers als zusätzlicher "Hilfs"-Datenstruktur einen iterativen Algorithmus an, der f (x,y,z) fair beliebige x,y,z ∈ N_0 berechnet. Erläutern Sie Idee und wesentliche Schritte Ihrer Lösung.
4. Beweisen Sie, dass für alle x,y∈ N_0 gilt: f (x,y,2) = g (x,y).
5. Beweisen Sie, dass die Funktion g sich unter ausschließlicher Verwendung von Addition und Multiplikation (auf N_0) sowie primitiver Rekursion definieren lässt.

1.10 Frühjahr 1996

Aufgabe 1

Zeigen Sie, dass es zu jedem (nichtdeterministischen) endlichen erkennenden Automaten Asp mit spotanen Übergängen einen äquivalenten (nichtdeterministischen) endlichen erkennenden Automaten Ansp ohne spontane Übergänge gibt!

(Hinweis: Bei einem spontanen Übergang (z, ε, z') wird kein Symbol eingelesen.)

Aufgabe 2

Konstruieren Sie einen linear beschränkten Automaten, der die Sprache

L = {u§v | u ist Prefix von v}

akzeptiert!

Aufgabe 3

Zeigen Sie, dass es zu jeder nichtleeren, rekursiv aufzählbaren Menge A natürlicher Zahlen eine primitiv rekursive Funktion f gibt, deren Bildmenge A ist!

Aufgabe 4

Für welche Sprachen-Klassen der Chomsky-Hierarchie ist das Wortproblem entscheidbar (Begründung!)?

Aufgabe 5

Erklären und vergleichen Sie die Aufrufprinzipien ‚call by value' und ‚call by reference'!

Aufgabe 6

Geben Sie eine Syntax-Graphen-Darstellung zur EBNF

```
A = "x" | "(" B ")"
B = AC
C = {"+"A}
```

1.11 Herbst 1996

Aufgabe 1

Gegeben Sei folgendes Pascal-Programm:

```
program Parametertest;

var n: integer;
var a: array[1.2] of integer:
```

```
procedure update( x,y integer);
var n: integer;
begin
  n:= 1;
  z:= x+n;
  Y:= x*y;
end;

begin
  n:=1;
  a[1]:=2;
  a[2]:=7;
  update(n,a[n]);
end.
```

Geben Sie für die folgenden Parameterübergabetechniken jeweils an, welche Werte die Variablen n, a[1] und a[2] am Ende der Programmausführung haben.

1. call-by-value
2. call-by-reference
3. call-by-name

Aufgabe 2

Gegeben sei eine Funktion d : $N_\perp$ x $N_\perp \rightarrow N_\perp$, die wie folgt definiert ist:

$$d(x,y) = \begin{cases} \perp & , falls \quad x = \perp \; oder \; y = \perp \\ 1 & , falls \quad ein \; z \; existiert \; mit \; z \cdot y = x \\ 0 & , sonst \end{cases}$$

1. Beschreiben Sie informell (Stichworte), welche Funktion durch *d* dargestellt wird.
2. Schreiben Sie eine rekursive Funktion, welche die Funktion *d* berechnet! Die Funktionen DIV (Division) und MOD (modulo) dürfen nicht verwendet werden.
3. Stellen Sie das zu Ihrer Funktion gehörende Funktional Θ auf. Geben Sie dabei auch die Funktionalität von Θ an.
4. Zeigen Sie, dass *d* ein Fixpunkt des Funktionals Θ ist, d.h. dass Ihr Programm eine Implementierung von *d* ist.

Aufgabe 3

Programmieren Sie in Modula oder Pascal die folgenden Datentypen, Funktionen und Prozeduren:

1. Definieren Sie rekursiv den Typ *Tree* der binären Bäume, deren Knoten ganze Zahlen enthalten.
2. Programmieren Sie eine Funktion *is_in* die einen binären Baum *t* aus Aufgabe 1. und eine ganze Zahl *n* als Eingabeparameter nimmt und einen boole'schen Wert als Ergebnis liefert, so dass *is_in(t,n)* den Wert TAUE liefert genau dann, wenn *n* in *t* vorkommt.
3. Schreiben Sie eine rekursive Funktion *schachtelung*:
   `PROCEDURE schachtelung (x:REAL;ug,og:REAL; eps:REAL):REAL,`

die näherungsweise die Wurzel einer reellen Zahl nach der Methode der Intervallschachtelung berechnet, d.h. x ist die Zahl, deren Wurzel berechnet werden soll, ug und og sind die Unter- bzw. Obergrenze des gerade betrachteten Intervalls, eps ist die Genauigkeit, mit der die Wurzel berechnet werden soll. Die Rekursion soll abgebrochen werden, falls $|m^2 - x| < eps$. Dabei sei m der Mittelwert des Intervalls [ug . . . og]. Die Funktion wird aufgerufen mittels `schachtelung(x,0.0,x,eps)`. Der Absolutbetrag kann mit Hilfe einer Funktion `ABS(x)` berechnet werden.

Aufgabe 4

Gegeben sei der deterministische endliche Automat $M = (Q, \Sigma, \delta, q_0, F)$ mit $\Sigma = \{a, b\}$, $Q = \{q_0, q_1, q_2, q_3, q_4\}$, $F = \{q_4\}$ und

$\delta(q_0,a) = q_1$	$\delta(q_2,a) = q_4$
$\delta(q_0,b) = q_2$	$\delta(q_2,b) = q_3$
$\delta(q_1,a) = q_4$	$\delta(q_3,a) = q_4$
$\delta(q_1,b) = q_3$	$\delta(q_3,b) = q_4$
$\delta(q_4,a) = q_3$	$\delta(q_4,b) = q_3$

1. Zeichnen Sie den Automaten als Übergangsdiagramm.
2. Berechnen Sie einen äquivalenten Automaten mit minimaler Anzahl von Zuständen.
3. Geben Sie den Sprachschatz des Automaten an (ohne Beweis).
4. Ist diese Sprache Typ-3 (regulär) ? (mit Begründung !)
5. Zeigen Sie, dass folgende Sprache $L \subseteq \{a\}^*$ nicht Typ-3 (regulär) ist:

$$L = \{a^p \mid p = \sum_{i=1}^{n} i \text{ für ein } n \in N, n > 0\}$$

Aufgabe 5

Gegeben sei die Typ-2 Grammatik $G = (V, \Sigma, P, S)$ mit

$V = \{S, A\}$, $\Sigma = \{(,)\}$, $P = \{ \; S \to (A , \quad S \to (AS, \quad S \to (SA, \quad S \to (SAS, \quad A \to) \; \}$

2. Welche Sprache $L(G)$ erzeugt G ?
3. Geben Sie einen PDA (d.h. einen nichtdeterministischen Kellerautomaten) *K* an mit $N(K) = L(G)$ (ohne Korrektheitsbeweis). Zur Erinnerung: $N(K)$ ist die Sprache, die von *K* durch leeren Keller erkannt wird.
4. Inwiefern ist der PDA *K* geeignet zum Parsen der Sprache $L(G)$?
   Welche Parsingtechniken kennen Sie ?

1.12 Frühjahr 1997

Aufgabe 1 Automatentheorie und Formale Sprachen

Zu jedem *nichtdeterministischen* endlichen Automaten (NFA) *A* kann man einen *deterministischen* endlichen Automaten (DFA) *A'* konstruieren, der die gleiche Sprache akzeptiert, d.h. $L(A) = L(A')$.

1. Geben Sie ein Verfahren für die Konstruktion von *A'* aus *A* detailliert an!

2. Für das Alphabet $\Sigma = \{a, b\}$ und natürliche Zahlen n seien Sprachen L_n und R_n definiert durch

   $$L_n := \Sigma^n \cdot b \cdot \Sigma^* , R_n := \Sigma^* \cdot b \cdot \Sigma^n$$

   Geben Sie NFAs L_n bzw. R_n an, welche L_n bzw. R_n akzeptieren!

3. Konstruieren Sie in den Fällen n = 0,1,2 die zugehörigen DFAs L'_n bzw. R'_n.
4. Zeigen Sie, dass der minimale DFA, der L_n akzeptiert, n + 3 Zustände hat.
5. Zeigen Sie, dass der minimale DFA, der R_n akzeptiert, 2^n+1 Zustände hat.

Aufgabe 2 Berechenbarkeit

Zu den klassischen Problemen in der Theorie der Berechenbarkeit zählt das sog. "Korrespondenzproblem von POST" (PCP).

1. Formulieren Sie die Problemstellung des PCP.
2. Das PCP ist algorithmisch unentscheidbar. Begründen Sie dies!
3. Die Unentscheidbarkeit des PCP kann dazu verwendet werden, mittels Reduktion die Unentscheidbarkeit von Problemen im Bereich der kontextfreien Grammatiken und Sprachen nachzuweisen. Geben Sie ein konkretes Beispiel hierfür an (Problemstellung und Reduktion)!

Aufgabe 3 Algorithmische Sprachen

Zwei bekannte Analysealgorithmen für allgemeine kontextfreie Sprachen sind mit den Namen COOKE/YOUNGER/KASAMI (CYK) und EARLEY verbunden.

1. Erläutern Sie die Vorgehensweise bei beiden Algorithmen in ihren Grundzügen.
2. Welche Aussagen über die Laufzeitkomplexität (evtl. auch für Teilfamilien der Familie aller kontextfreien Sprachen) sind Ihnen bekannt?
3. Stellen Sie einen der beiden Algorithmen detailliert dar (z.B. in Pseudocode), begründen Sie dessen Korrektheit und Laufzeitkomplexität.

Aufgabe 4 Programmiermethodik, Effiziente Algorithmen

Eine der grundlegenden Aufgabenstellungen der Algorithmik ist die Identifizierung und Klassifikation von Problemen, für die es effiziente Entscheidungsalgorithmen bzw. effiziente Verifikationsalgorithmen gibt. Die entsprechenden Problemklassen werden mit *P* bzw. *NP* bezeichnet.

1. Geben Sie Definitionen für die Klassen *P* und *NP* an und erläutern Sie diese an Beispielen.
2. Erläutern Sie, in welchem Sinne die o.g. Klassen "robust" gegenüber definitorischen Variationen sind.
3. Was versteht man unter der *P*- vs.-*NP* –Problematik?
4. Beschreiben Sie das Reduktionskonzept innerhalb der Klasse *NP* an einem Beispiel.
5. Was sind und welche Bedeutung haben *NP* -vollständige Probleme? Erläutern Sie diese Begriffsbildung und geben Sie Beispiele für solche Probleme an.

6. Betrachten Sie das Problem, bei dem man danach fragt, ob eine vorgelegte aussagenlogische Formel eine Tautologie ist, d.h. ob sie unter allen Belegungen der in ihr vorkommenden Aussagenvariablen mit Wahrheitswerten "wahr" bzw. "falsch" den Wert "wahr" annimmt.
   Diskutieren Sie den Status dieses Problems in Bezug auf die Problemklassen *P* und *NP*.

1.13 Herbst 1997

Aufgabe 1 Rechenstrukturen und Termersetzungssysteme

Gegeben sei die Rechenstruktur `NAT` mit der Signatur:

$\Sigma = (S_{NAT}, F_{NAT})$ mit S_{NAT} = {bool, nat} und F_{NAT} = {zero, succ, pred}

Den beiden Sorten seien die folgenden Trägermengen zugeordnet:

$bool^{NAT} = B^{\perp} := B \cup \{\perp\}$, $nat^{NAT} = N^{\perp} := N \cup \{\perp\}$

Dabei ist N = {0, 1, 2, ...} die Menge der natürlichen Zahlen und B = {TRUE, FALSE} die Menge der Boolschen Werte. Das Symbol $\perp$ steht für „undefiniert“.

Die Spezifikationen der den Funktionssymbolen zero, succ und pred zugeordneten strikten Funktionen lauten:

$zero^{NAT}$:		$\rightarrow N^{\perp}$;	$zero^{NAT}$	= 0;
$succ^{NAT}$:	$N^{\perp}$	$\rightarrow N^{\perp}$;	$succ^{NAT}(x)$	= x+1;
$pred^{NAT}$:	$N^{\perp}$	$\rightarrow N^{\perp}$;	$pred^{NAT}(x)$	= x-1, falls $x \geq 1$ und
			$pred^{NAT}(0)$	= $\perp$;

Im Folgenden nehmen wir an, dass jede natürliche Zahl durch einen Grundterm dargestellt ist, der nur aus den Konstruktoren zero und succ besteht.

1. Geben Sie ein Termersetzungssystem an, das die Funktion pred auf die Konstruktoren zero und succ zurückführt!

2. Nun sollen die strikten Funktionen add und sub mit den Spezifikationen

add^{NAT}:	$N^{\perp} \times N^{\perp} \rightarrow N^{\perp}$;	$add^{NAT}(x,y) = x+y$
sub^{NAT}:	$N^{\perp} \times N^{\perp} \rightarrow N^{\perp}$;	$sub^{NAT}(x,y) = x-y$ falls $x \geq y$
		$sub^{NAT}(x,y) = \perp$ falls $x < y$

   zu F^{NAT} hinzugefügt werden ($x,y \neq \perp$). Geben Sie Termersetzungssysteme an, die add bzw. sub durch zero, succ und pred darstellen.

3. Stellen Sie analog zur Teilaufgabe 2. je ein Termersetzungssystem für die folgenden, ebenfalls strikten Funktionen auf (mit $x,y \neq \perp$):

$mult^{NAT}$:	$N^{\perp} \times N^{\perp} \rightarrow N^{\perp}$;	$mult^{NAT}(x,y) = x * y$
div^{NAT}:	$N^{\perp} \times N^{\perp} \rightarrow N^{\perp}$;	$div^{NAT}(x,y) = x / y$ falls $y > 0$
		$div^{NAT}(x,0) = \perp$

   Sie dürfen dabei auch die Funktionen add, sub aus 2. verwenden.

Wir betrachten nun den seit dem Altertum bekannten Euklidschen Algorithmus in der Notation einer imperativen Programmiersprache (a,b seien von der Sorte nat)

```
(*)   while a ≠ b do
          while a<b do b:=b-c od
          (a,b):=(b,a)
          od
```

4. Stellen Sie einen exemplarischen Ablauf des Algorithmus für die Variablenbelegung *a = 32, b = 18* als Folge von Zuständen des Variablenraumes dar.

5. Der Algorithmus soll nun mit Hilfe der Zusicherungsmethode nach Floyd und Hoare verifiziert werden. Geben Sie für beide `while`-Schleifen geeignete Invarianten an, und beweisen Sie damit die Korrektheit des Algorithmus.
   Hinweis:
   Gehen Sie von der Zusicherung {P ∧ a = mg ∧ b = ng ∧ n,m teilerfremd} mit P = a > 0 ∧ b > 0 vor Beginn der ersten `while`-Anweisung aus.

6. Formulieren Sie den Euklidischen Algorithmus rekursiv in einer beliebigen Notation.

7. Geben Sie ein Termersetzungssystem auf der Rechenstruktur `NAT` für den Euklidischen Algorithmus an. Sie dürfen dabei alle in den Teilaufgaben 1. mit 3. auf `NAT` eingeführten Funktionen und Hilfsfunktionen verwenden.

8. Beweisen Sie die Terminierung der beiden `while`-Schleifen in unserer ursprünglichen Formulierung (*) des Euklidischen Algorithmus.

Aufgabe 2 Rekursive Rechenstrukturen

Gegeben sei die rekursive Rechenstruktur

Liste = Leer | (Float, Liste)

der Listen über reellen Gleitpunktzahlen mit den Funktionen

- *head:* *Liste ® Float*
- *tail:* *Liste® Liste*
- *mklist:* *Float x Liste® Liste*

Für diese Funktionen gelten folgende Spezifikationen:

- *head(x)* und *tail(x)* sind partiell nur für nicht-leere Listen definiert, und
  head(x) ist das erste Element der Liste x,
  tail(x) ist die um das erste Element verkürzte Liste x.

- *mklist(r, x)* ist total und liefert die Liste, die durch Voransetzen des Elements *r vor* die Liste x entsteht.

Die Rechenstruktur *Liste* soll nun um die folgenden Funktionen erweitert werden:

- *length : Liste ® Int*

mit der Spezifikation: *length(x)* ist total und liefert die Anzahl der Elemente in der Liste x.

- *proj: Int x Liste ® Float*
  mit der Spezifikation: *proj(n, x)* ist partiell nur für nicht-leere Listen x sowie für n mit $1 \leq n \leq$ length(x) definiert und liefert das n-te Element der Liste x.

- *part: Int x Int x Liste ® Liste*
  mit der Spezifikation: *part(m, n, x)* ist partiell nur für nicht-leere Listen x sowie für m und n mit $1 \leq m \leq n \leq$ *length(x)* definiert und liefert die Teilliste von x vom m-ten bis zum n-ten Element einschließlich.

1. Programmieren Sie die drei Funktionen *length, proj* und *part* rekursiv unter Abstützung auf die primitiven Rechenstrukturen *Int* und *Bool* sowie auf die oben angegebene Rechenstruktur *Liste.*

2. Beweisen Sie für das von Ihnen für die Funktion *part* angegebene Programm,
2.1 dass es für alle zulässigen Parameter terminiert und
2.2 dass es die Spezifikation für die Funktion *part* erfüllt.

Aufgabe 3 Endliche Automaten und reguläre Mengen

Gegeben sei das Alphabet A = (A, *B, C).* In A^* zeichnen wir die Teilmenge T der Wörter aus, die weder ACC noch BCC als Teilzeichenreihe enthalten. Dabei ist $x \in A^*$ genau dann eine Teilzeichenreihe von $y \in A^*$, wenn es ein $y' \in A^*$ und ein $y'' \in A^*$ gibt, so dass y = y'xy'' ist.

1. Konstruieren Sie den (bis auf die Bezeichnungen der Zustände eindeutigen) minimalen deterministischen endlichen Automaten A = (S, I, δ, s_0, F) mit der Zustandsmenge S, dem Eingabealphabet I = A, der Zustandsübergangsfunktion $\delta : S \times I \rightarrow S$, dem Anfangszustand s_0 und der Endzustandsmenge F, der genau T akzeptiert! Stellen Sie hierzu den Automaten A durch seinen Zustandsübergangsgraphen dar.

2. Beweisen Sie, dass der von Ihnen in der Antwort zu 1. angegebene Automat A
2.1 genau T akzeptiert und
2.2 minimal ist.

3. Stellen Sie T als eine reguläre Menge über A dar.

1.14 Herbst 1998 I

Aufgabe 1 Grammatiken und Automaten

Wir wollen Boolesche Ausdrücke BA als Sprache über L* mit L = {0,1,x,y,+,-,(,)} darstellen. Dabei stehen 0,1 für die Booleschen Konstanten, x,y für Boolesche Variable, + für das Disjunktionszeichen und ein vorausgestelltes – für die Negation. Das Konjunktionszeichen wird weggelassen. Die Klammern „(„ und „)“ können weggelassen werden, falls sie wegen der Regel, dass die Konjunktion stärker bindet als die Disjunktion, oder wegen der Assoziativgesetze überflüssig sind.

Unter Verwendung der genannten Symbole sollen die Boolschen Ausdrücke nach den folgenden Regeln aufgebaut werden:

a) Literale sind Boolsche Konstanten und Variablen. Ein negiertes Literal ist auch ein Literal. Nichts sonst ist ein Literal. Beispiel: `---0`
b) Schreibt man zwei BA hintereinander, so erhält man eine Konjunktion. Beispiel: `x-1, xy0, 0y-1x`
c) Jeder BA ist eine (triviale) Disjunktion. Die Summe zweier Disjunktionen ist wieder eine Disjunktion. Beispiel: `x-1, x+1, y+-x+0`
d) Jedes Literal und jede Konjunktion ist ein BA. Jede eingeklammerte Disjunktion ist ein BA.

1. Geben Sie eine kontextfreie Grammatik (Chomsky-Typ 2) für die oben definierten BA an und leiten Sie die beiden Ausdrücke `0x-1y` und `((0+x)y-x)` mit Hilfe Ihrer Grammatik ab.
2. Beweisen Sie die folgende Aussage:
   Ersetzt man in der Sprache (bzw. der in Aufgabe 1 konstruierten Grammatik) alle Terminalzeichen bis auf die Klammern „`(`“ und „`)`“ durch das leere Wort ε, dann ist die so erhaltene Sprache immer noch nicht regulär.

Nun definieren wir mit den obigen Gesetzen eine andere Sprache BA*, indem wir a), b), d) beibehalten und c) folgendermaßen verändern:

c)* Jedes Literal ist eine (triviale) Disjunktion. Die Summe zweier Disjunktionen ist wieder eine Disjunktion.

3. Stellen Sie eine Grammatik auf, die BA* erzeugt und leiten Sie damit die Ausdrücke `(0)(x)` und `(1+x+y)` ab.
4. Zeigen Sie, dass BA* regulär (vom Chomsky-Typ 3) ist, indem Sie einen endlichen Automaten angeben, der BA* erkennt.
5. Nun vereinfachen wir die Sprache BA aus Teilaufgabe 1 zu BA', indem wir als einziges Literal das Zeichen *l* zulassen. Geben Sie einen Kellerautomaten an, der die Sprache BA' erkennt.

Aufgabe 2 Turingmaschinen und Berechenbarkeit

Wir betrachten eine Turingmaschine $T = (I, B, Q, \delta, q_0)$, wobei $I = \{0,1\}$ das Eingabealphabet, $B = I \cup \{\#\}$ das Bandalphabet mit dem Leerzeichen #, $Q = \{q_0, \ldots, q_6\}$ eine Menge von Zuständen, $\delta = Q \times B \rightarrow Q \times B \times \{\leftarrow,\downarrow,\rightarrow\}$ die Zustandsübergangsfunktion und q_0 der Anfangszustand ist.

Die Zeichen ← bzw. → symbolisieren eine Bewegung des Schreib-/Lesekopfes (nach der aktuellen Operation) nach links bzw. rechts. Die Maschine hält nach der aktuellen Operation, wenn die Zustandsübergangsfunktion nicht definiert ist oder auf ein ↓ führt.

Als Eingabewörter lassen wir nur Zeichenfolgen der Form 0^n10^m mit $n,m \in N_0$ zu, wobei 0^n für eine Zeichenkette aus n Nullen stehe. Am Anfang stehe der Schreib-/Lesekopf auf dem ersten (linken) Zeichen des Eingabewortes. Links und rechts vom Eingabewort stehen ausschließlich Leerzeichen auf dem Band.

Die Werte der Zustandsübergangsfunktion $\delta(q,b)$ seien durch folgende Tabelle definiert:

Bandzeichen $b \in B$ Zustand $q \in Q$	0	1	#
q_0	$(q_1, \#, \rightarrow)$	$(q_5, \#, \rightarrow)$	$(q_6, \#, \downarrow)$
q_1	$(q_1, 0, \rightarrow)$	$(q_2, 1, \rightarrow)$	nicht definiert
q_2	$(q_3, 1, \leftarrow)$	$(q_2, 1, \rightarrow)$	$(q_4, \#, \leftarrow)$
q_3	$(q_3, 0, \leftarrow)$	$(q_3, 1, \leftarrow)$	$(q_0, \#, \rightarrow)$
q_4	$(q_4, 0, \leftarrow)$	$(q_4, \#, \leftarrow)$	$(q_6, 0, \downarrow)$
q_5	$(q_5, \#, \rightarrow)$	$(q_5, \#, \rightarrow)$	$(q_6, \#, \downarrow)$

1. Beschreiben Sie die vollständige Berechnung von T für das Eingabewort `0010`, indem Sie für jeden Berechnngsschritt den jeweiligen Maschinenzustand und die jeweilige Bandbelegung angeben.
2. Stellen Sie die Arbeitsweise der Maschine T mit Hilfe eines geeigneten Zustandsübergangsdiagramms dar.
3. Die zulässigen Eingabewörter 0n10m sollen nun jeweils ein Paar natürlicher Zahlen (m,n) repräsentieren. Welche Funktion f: N_0 x $N_0 \rightarrow N_0$ wird von T berechnet? Begründen Sie Ihre Aussage.
4. Untersuchen Sie, ob Ihre in Teilaufgabe 3 angegebene Funktion primitiv rekursiv ist und beweisen Sie Ihre Ausage.
5. Geben Sie eine möglichst kleine obere Schranke für die Anzahl der Berechnungsschritte von T für ein zulässiges Eingabewort der Länge `|w|=k` an.

Aufgabe 3 Spezifikation und Implementierung

Geben sei die folgende Spezifikation der natürlichen Zahlen

```
NAT =     based_on  BOOL
          sorts     nat
          ops       0: nat
                    s: nat → nat
                    p: nat → nat
                    +: nat x nat → nat
                    zero: nat → bool

axioms    nat generated_by 0, s
          p(0) = 0
          p(s(x)) = x
          0+y = y
          s(x)+y = s(x+y)
          zero(0) = True
          zero(s(x)) = False
```

Dabei bedeutet **based_on** BOOL, dass NAT auf einer geeigneten Spezifikation von BOOL aufbaut. Die Aussage nat **generated_by** 0,s drückt aus, dass alle Elemente aus nat als Terme der Form `s(s(..(0)..))` dargestellt werden können.

1. Nun soll die Spezifikation NAT in einer funktionalen Programmiersprache Ihrer Wahl implementiert werden, wobei natürliche Zahlen als Bitlisten, das heißt als Sequenzen

Boolscher Werte vom Typ [bool] dargestellt werden sollen. Programmieren Sie die dazu nötigen Funktionen.

2. Geben Sie die Abstraktionsfunktion phi: [bool] $\rightarrow$ Nat an, die jede Bitliste, die sich mit Hilfe Ihrer Implementierung erzeugen lässt, auf die dazugehörige natürliche Zahl abbildet. Mit Nat ist dabei die Sorte gemeint, die in der von Ihnen gewählten Programmiersprache die natürlichen Zahlen bzw. eine geeignete Obermenge davon implementiert.

1.15 Herbst 1998 II

Aufgabe 1

1. Gegeben sei eine Grammatik G mit {1,2,a,b} als Menge der Terminalzeichen, der Variablen S, der Startvariablen S und den Produktionen

   $S \rightarrow 1Sab, \quad S \rightarrow 1ab, \quad S \rightarrow 2Sbbb, \quad S \rightarrow 2bbb$

   Sei L = L(G) die von G erzeugte Sprache, w_1 = ab und w_2 = bbb.

1.1 Beweisen Sie: L(G) = {i1 ... ik | i1 ... ik $\in$ {1,2}}.
1.2 Konstruieren Sie einen deterministischen Kellerautomaten, der L akzeptiert.
1.3 Ist L regulär? (Begründung)

2. Beweisen oder widerlegen Sie für beliebige Sprachen L_1, L_2:
2.1 $(L_1 \cup L_2)^* = L_1^* \cup L_2^*$
2.2 $(L_1^*)^* = L_1^*$

Aufgabe 2

Beweisen Sie: Ist f: N $\rightarrow$ N LOOP-berechenbar, so ist auch g: N $\rightarrow$ N mit

$$g(n) = \sum_{i=1}^{n} f(i)$$

LOOP-berechenbar.

Aufgabe 3

22 Vertreter aus sieben Staaten besuchen eine internationale Konferenz. Welche der folgenden Aussagen muss wahr sein? (Begründung)

3.1 Sechs Staaten haben jeweils drei Vertreter und der siebte Staat hat vier Vertreter.
3.2 Kein Staat hat mehr als vier Vertreter.
3.3 Mindestens ein Staat hat vier oder mehr Vertreter.
3.4 Kein Staat hat weniger als zwei Vertreter.

Aufgabe 4

1. Aussagenlogik
1.1 Beweisen Sie:
   „$(A \rightarrow B) \leftrightarrow (\neg A \vee B)$“ ist eine Tautologie.
1.2 Begründen Sie die Aussage:
   „Aus {F,G} folgt H.“ ist äquvalent zu „{F,G,$\neg$H} ist unerfüllbar“

1.3 Beweisen Sie mit dem Resolutionskalkül:
„{¬A∨B , ¬B∨C, A, ¬C } ist unerfüllbar“
1.4 Beweisen Sie (unter Verwendung der Teilaufgaben 1.1, 1.2 und 1.3):
„Aus {A→B, B→C} folgt A→C“

2. Prädikatenlogik
Gegeben ist die folgende Formel: F: $((\forall x \exists y P(x,y)) \rightarrow (\exists x \forall y P(x,y)))$
2.1 Ist F allgemeingültig?
2.2 Ist F erfüllbar?
2.3 Falls F ein Modell hat, geben Sie ein solches an.

Aufgabe 5

1. Stellen Sie sich vor, Sie starten für jede der folgenden Teilaufgaben den SCHEME-Interpreter erneut und geben die folgenden Ausdrücke ein. Geben Sie für jede Teilaufgabe das Ergebnis der Auswertung des letzten Ausdrucks an.

1.1
```
(define zwei 2)
(define drei 3)
(define eins 8)
(define plus +)
(define kleiner <)
(kleiner eins (plus zwei drei))
```
1.2
```
(cons ´a (cons ´b (cons (cdr (cons (cons ´d 1)(cons ´c 2))) ´())))
```
1.3
```
(define (rek a)
        (cond  ((= a 0) 5)
               ((= a 5) (* 2 (rek (- a 5))))
               (else (lamda (x) (rek (- x 5))))))
((rek (rek 5)) (rek 5))
```

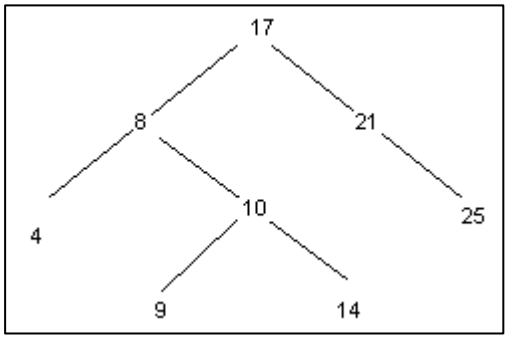
2. Definieren Sie in einer funktionalen Sprache Ihrer Wahl eine Funktion, die das Skalarprodukt zweier Vektoren berechnet, die in rechtwinkligen Koordinaten gegeben sind. Die Vektoren sollen als Listen repräsentiert sein. Sie können davon ausgehen, dass Ihre Funktion nur mit Vektoren aufgerufen wird, die gleich lang sind.
Beispiel in SCHEME: (skalarprodukt ´(1 2 3) ´(2 2 2)) liefert 12.

Aufgabe 6

Ein geordneter binärer Baum ist entweder ein leerer Baum oder besteht aus einer Wurzel und zwei binären Bäumen als als linkem und rechtem Unterbaum. Dabei sind die Knoten des Baums so geordnet, dass für alle Knoten v’ im linken Unterbaum und für alle Knoten v’’ im rechten Unterbaum von Knoten v gilt:

$$v' \leq v \leq v''$$

In SCHEME werden Bäume durch Listen kodiert. Der folgende Baum



hat die Listendarstellung:

```
´(17 (8 (4 () ())
        (10 (9 () ())
            (14 () ())))
     (21 ()
         (25 () ())))
```

1. Definieren Sie in einer funktionalen Sprache Ihrer Wahl die folgenden Funktionen:

1.1 `root` liefert den Wert an der Wurzel des Baumes.
1.2 `left-branch` liefert den linken Teilbaum der Wurzel.
1.3 `right-branch` liefert den rechten Teilbaum der Wurzel.
1.4 `make-tree` bildet aus einer Wurzel und zwei Bäumen einen neuen Baum.
1.5 Der leere Baum sei als leere Liste definiert. Definieren Sie die Funktion `empty-tree?`, die abfragt, ob ein Baum leer ist.

2. Definieren Sie die Funktion `contains?`, die als Argumente einen binären Baum und ein Element nimmt und überprüft, ob das Element im Baum enthalten ist.

3. Definieren Sie die Funktion `insert`, die als Argumente einen binären Baum und ein Element nimmt und das Element an der richtigen Stelle im Baum einsortiert.

Aufgabe 7

Für Ihr örtliches Rathaus sollen Sie eine Datenbank entwerfen, die personenbezogene Daten zu jedem Einwohner Ihrer Stadt speichert. Zu jedem Einwohner und jeder Einwohnerin werden die personenbezogenen Daten gespeichert, die auf seinem bzw. ihrem Personalausweis vermerkt sind. Dies sind eine 9-stellige Personalausweisnummer, Vor- und Nachname der Person, Geburtsdatum, Geburtsort, Staatsangehörigkeit, Ablaufdatum der Gültigkeit, Wohnort, Augenfarbe und Größe.

1. Definieren Sie mit den Mitteln einer beliebigen imperativen Programmiersprache (z.B. C) zunächst einen geeigneten Datentyp `pers_Daten`, der die oben genannten Informationen aufnehmen soll. Sie können dabei davon ausgehen, dass kein Orts- oder Personenname länger als 40 Zeichen ist.
2. Sie wollen die Personen nun in Stadtteilkarteien zusammenstellen. Ein Stadtteil besteht aus einer konstanten maximalen Anzahl von Einwohnern. Definieren Sie mit den Mitteln der gewählten Programmiersprache zunächst eine Konstante `MAXANZAHL` mit dem Wert 12000. Legen Sie anschließend einen Datentyp `stadtteil` fest, der eine feste, vorher bekannte Anzahl von Einwohnern, nämlich `MAXANZAHL`, vom Typ `pers_Daten` aufnehmen soll.

1.16 Frühjahr 1999 I

Aufgabe 1

In dieser und den folgenden Aufgaben geht es um die Klasse der sog. *regulären* Sprachen, also der Sprachen vom Typ 3 in der Klassifizierung von Chomsky.

1. Geben Sie eine genaue Definition der Klasse der regulären Sprachen an und erläutern Sie die darin vorkommenden Begriffe.
2. Es gibt mehrere äquivalente Beschreibungen für diese Klassen der regulären Sprachen (die man also ebensogut als Definitionen verwenden könnte):
   Zählen Sie solche weiteren Beschreibungen auf, die Sie kennen. Erläutern Sie ggf. die verwendeten Begriffe.
3. Unter welchen Operationen auf Sprachen ist die Klasse der regulären Sprachen abgeschlossen? Zählen Sie die Ihnen bekannten auf.
4. Welche Hilfsmittel kennen Sie, um die *Nicht-Regularität* einer formalen Sprache nachzuweisen. Erläutern Sie dies an einem Beispiel Ihrer Wahl.
5. In welchen Bereichen der Informatik haben reguläre Sprachen eine praktische Bedeutung? Erläutern Sie solche Anwendungen, wenn möglich unter Angabe relevanter Algorithmen (in Pseudocode).

Aufgabe 2

In dieser Aufgabe werden Sprachen über dem Alphabet $\Sigma = \{0,1\}$ betrachtet. Für ein Wort $w = w_0w_1...w_m \in \Sigma^*$ bezeichne $(w)_2$ die Interpretation von w als Binärdarstellung einer natürlichen Zahl, wobei das führende Bit links steht:

$$(w)_2 = w_0 2^m + w_1 2^{m-1} + ... + w_{m-1} 2^1 + w_m 2^0$$

(N.B. es wird nicht vorausgesetzt, dass $w_0 = 1$ ist, d.h. es sind beliebig viele führende Nullen erlaubt). Für $n,k \in N$ mit $0 \leq k < n$ sei

$$\text{mod}(n,k) := \{w \in \Sigma^*; (w)_2 \equiv k \pmod n\}$$

die Sprache der Binärdarstellungen derjenigen Zahlen, die congruent zu k modulo n sind.

1. Konstruieren Sie einen endlichen Automaten, der die Sprache `mod(5,0)` erkennt. Wie erhält man daraus die endlichen Automaten, die die Sprachen `mod(5,k)` mit $1 \leq k \leq 4$ erkennen?
2. Formulieren Sie eine allgemeine Aussage über die Erkennbarkeit der Sprachen `mod(n,k)` durch endliche Automaten. Skizzieren Sie einen Beweis.

Aufgabe 3

In dieser Aufgabe werden Sprachen über dem Alphabet $\Sigma = \{a,b\}$ betrachtet. Für ein Wort $w \in \Sigma^*$ bezeichne $|w|_a$ (bzw. $|w|_b$) die Anzahl der Vorkommen von a (bzw. b) in dem Wort w. Betrachten Sie die Sprachen:

$C_k := \{w \in \Sigma^*; |w|_a = |w|_b \wedge w = u \cdot v \Rightarrow ||u|_a - |u|_b \leq k\}$ $(k \geq 1)$
$D := \{ w \in \Sigma^*; |w|_a = |w|_b \}$

Offenbar gilt:

$C1 \subset C2 \subset C3 \subset ... \subset D$ und $\lim\uparrow_{k\to\infty} C_k = D$

1. Zeigen Sie, dass die Sprachen C_k $(k \geq 1)$ regulär sind.
2. Konstruieren Sie endliche Automaten, die die Sprachen C_1 bzw. C_2 akzeptieren.
3. Zeigen Sie, dass die Sprache D nicht regulär ist.
4. Zu welcher Sprachklasse gehört die Sprache D? Mit welchem Typ von „Automaten" kann man die Sprache D erkennen? (Auf beide Fragen ist natürlich eine möglichst stark eingegrenzte Antwort erwünscht).

1.17 Frühjahr 1999 II

Aufgabe 1

Geben Sie primitiv rekursive Terme für die Funktionen in 1. bis 5. an:

1. Die Fakultätsfunktion f(x) = x! .
2. Die Funktion *even* mit $even(x) = \begin{cases} 1 & \text{wenn x gerade} \\ 0 & sonst \end{cases}$
3. Die Signumfunktion *sig* mit $sig(x) = \begin{cases} 1 & \text{wenn x} > 0 \\ 0 & sonst \end{cases}$
4. Die Funktion *case* mit $case(x, y, z) = \begin{cases} y & \text{wenn x} = 0 \\ z & sonst \end{cases}$
5. Die ganzzahlige Division *div* mit *div(x,y)*=k genau dann, wenn $k \cdot y + m = x$, wobei $m < y$.
6. Gegeben sei eine primitiv rekursive Funktion F: $N_0 \to N_0$. Wir definieren nun eine Funktion f: $N_0 \to N_0$ mit

$$f(n) = \begin{cases} 0 & \text{falls es ein } k < n \text{ gibt mit } F(k) = 0 \\ 1 & sonst \end{cases}$$

Ist f primitiv rekursiv? Beweisen Sie Ihre Antwort. Sie dürfen voraussetzen, dass die vier Grundrechenarten (+, -, ·, /) auf N primitiv rekursiv sind, alle weiteren von Ihnen eingeführten Hilfsfunktionen müssen explizit als primitiv rekursiv bewiesen werden.

Aufgabe 2

Gegeben sei das Alphabet $A = \{a,b\}$ und dei Zeichenreihenmenge $K \subseteq A^*$ mit $K = \{x \in A^*; |x| \geq 2$ und das vorletzte Zeichen von x ist ein a$\}$.

1. Beschreiben Sie K durch eine reguläre Menge.
2. Geben Sie eine Typ-3-Grammatik G mit dem Sprachsachatz K an und zeigen Sie, dass $L(G) = K$ gilt.

3. Geben Sie einen (evtl. nichtdeterministischen) endlichen Automaten M mit $T(M) = K$ an.
4. Geben Sie einen deterministischen Kellerautomaten M an mit $N(M) = \{a^{2n}b^n; n \in N\}$.

Aufgabe 3

Programmieren Sie in einer imperativen Prtogrammiersprache Ihrer Wahl nach folgenden Angaben:

Schreiben Sie eine Funktion *Tag*, die bestimmt, der wievielte Tag nach dem letzten Februartag eines Jahres zwischen 1900 und 2099 Ostersonntag ist. Dies lässt sich nach dem folgenden Verfahren ermitteln:

- q sei das Ergebnis der ganzzahligen Division der Jahreszahl j (1900 < j < 2099) durch 4.
- a sei der Rest der ganzzahligen Division von j durch 19.
- b sei der Rest der ganzzahligen Division von (204 - 11· a) durch 30. Falls b = 28 oder b = 29 ist, so ist mit b = 27 bzw. b = 28 weiterzurechnen.
- c sei der Rest der ganzzahligen Division von (j + q + b - 13) durch 7.

Das Ergebnis ist dann 28 + b - c.
Schreiben Sie ein Programm, das eine natürliche Zahl j mit 1900 < j < 2099 einliest, nach Überprüfung auf korrekte Eingabe mit Hilfe der Funktion *Tag* und einer Prozedur *Datum* das Datum des Ostersonntags des Jahres bestimmt und das Ergebnis ausgibt. Das Datum sei dabei in der Prozedur dargestellt durch zwei Ausgabeparameter t und m (z.B. t=7 und m=3 für den 7. März).

Aufgabe 4

Ein Firmengebäude hat 140 Räume. Es gibt ein Verzeichnis aller Räume, in dem für jeden Raum folgende Angaben gemacht werden:

- Raumkennung, bestehend aus einem Zeichen (z.B. ‚E' für Erdgeschoss, ‚K' für Keller) und einer natürlichen Zahl,
- Größe in Quadratmetern (als REAL-Zahl),
- Angabe, ob der Raum einen Telefonanschluss hat,
- Maximale Anzahl der Personen, die in diesem Raum arbeiten können (Null, wenn es sich um einen Lagerraum handelt),
- Aktuelle Anzahl der Personen, die in diesem Raum arbeiten.

Programmieren Sie in einer imperativen Programmiersprache Ihrer Wahl nach folgenden Angaben:

1. Geben Sie geeignete Datenstrukturen Raum und Raumverzeichnis an.
2. Geben Sie eine Prozedur an, die das Raumverzeichnis verändert, wenn einem neuen Mitarbeiter ein Raum zugeteilt wird, wobei man in einem Parameter angeben kann, ob man einen Raum mit oder ohne Telefonanschluss haben will.
   Beachten Sie dabei, dass die maximale Personenzahl pro Raum nicht überschritten wird. Der erste passende Raum kann gewählt werden. Sie können annehmen, dass ein solcher Raum vorhanden ist.
3. Geben Sie eine Funktion an, die die Gesamtgröße aller Lagerräume in Quadratmetern ermittelt.

1.18 Herbst 1999 I

Aufgabe 1

Gegeben seien die Sprache $L = \{a^{2n}b^{3n} \mid n > 0\}$ über $\Sigma = \{a,b\}$, die Grammatiken G_i (i=1,2) mit Startvariablen S_i und Produktionenmengen
$P_1 = P_0 \cup \{A \to aa, B \to bbb\}$
$P_2 = P_0 \cup \{S_2 \to LS_1R, LA \to aa, aA \to aaa, aB \to abbb, bB \to bbbb, bR \to b\}$,
wobei $P_0 = \{S_1 \to ABS_1, S_1 \to AB, BA \to AB\}$

1. Beweisen Sie:
1.1 $L \subseteq L(G_1)$,
1.2 $L \subseteq L(G_2)$.
2. Geben Sie ein Wort in $L(G_1)$ an, das nicht Element von $L(G_2)$ ist.
3. Ist L regulär? (Begründung!)
4. Ist L kontextfrei? (Begründung!)
5. Ist L kontextsensitiv? (Begründung!)

Aufgabe 2

Zeigen Sie:
Ist die Funktion f: $N^2 \to N$ (N Menge der natürlichen Zahlen einschließlich 0) LOOP-berechenbar (primitiv rekursiv), so ist auch die Funktion g: $N^2 \to N$ mit

$$g(x,y) = \prod_{i=0}^{y} f(x,i)$$

LOOP-berechenbar (primitiv rekursiv).

Aufgabe 3

Welche der folgenden Eigenschaften sind für (deterministische) Turingmaschinen M entscheidbar? (Begründung!)

1. M terminiert bei Eingabe 1999.
2. M berechnet eine LOOP-berechenbare Funktion.
3. Zu der von M berechneten Funktion f: $N^2 \to N$ gibt es ein f berechnendes WHILE-Programm mit höchstens zwei (ineinander geschachtelten) WHILE-Schleifen.

Aufgabe 4

Erklären und vergleichen Sie zwei Ihnen bekannte Parameter-Übergabe-Mechanismen bei Prozeduraufrufen.

1.19 Herbst 1999 II

Aufgabe 1

1. Geben Sie einen Datentyp für einfach verkettete Listen von ganzen Zahlen an. Als Programmiersprache können Sie dafür Pascal, Modula oder C wählen. Verwenden Sie zur Definition des Datentyps das Pointerkonzept der von Ihnen gewählten Sprache.
2. Schreiben Sie eine Prozedur oder eine Funktion `insert`, die eine ganze Zahl i in eine aufsteigende geordnete Liste l an die korrekte Stelle einordnet;
   z.B. liefert `insert` von 3 in <1,2,2,7,8> die Liste <1,2,2,3,7,8>.
3. Schreiben Sie eine Prozedur oder eine Funktion `löschen`, die in einer Liste l eine ganze Zahl i an der Stelle löscht, an der i in l zuerst vorkommt.

Aufgabe 2

1. Erklären Sie die Begriffe von Zeit- und Speicherplatz-Komplexität.

2. Ein bekanntes Spiel trägt den Namen „Türme von Hanoi“. Das Spiel wird mit n Scheiben verschiedener Größe gespielt, die auf Stäbe A, B, C gesteckt werden können. Zu Beginn stecken alle Scheiben auf Stab A und zwar derart, dass jeweils eine kleinere Scheibe auf einer größeren Scheibe liegt. Die Aufgabe besteht darin, die Scheiben von Stab A nach Stab B umzustecken, wobei folgende zwei Regeln zu beachten sind:
   (1) In jedem Schritt darf nur genau eine Scheibe bewegt werden,
   (2) eine größere Scheibe darf nie auf einer kleineren Scheibe liegen.

   Der Algorithmus, der die Zugfolge beschreibt, ist wie folgt:

```
hanoi(int n,var Stab quelle,var Stab ziel,var Stab via)
     if n=1 then
          /* bewege Scheibe von Quelle nach Ziel */
     else begin
          hanoi(n-1,quelle,via,ziel);
          /* bewege Scheibe von Quelle nach Ziel */
          hanoi(n-1,via,ziel,quelle);
     end;
end hanoi
```

   Beweisen Sie, dass die Zeitkomplexität von `hanoi` exponentiell ist. Sie können annehmen, dass das Bewegen von Scheiben immer 1 Zeiteinheit braucht.
   Hinweis: Geben Sie eine untere und eine obere Schranke für die Zeitkomplexität von `hanoi` an und verwenden Sie vollständige Induktion zum Beweis der Schranken.

Aufgabe 3

1. Sei $\{I_n\}_{n\in N}$ eine Familie von Sprachen, definiert durch
   $I_n = \{w \in \{0,1\}^* \mid |w| \geq n \geq 1$ und das n-letzte Zeichen von w ist 1$\}$
   Dabei bezeichnt |w| die Länge des Wortes w.

1.1 Geben Sie die allgemeine Form eines nichtdeterministischen endlichen Automaten für I_n an.

1.2 Geben Sie eine deterministischen endlichen Automaten für I_3 an.

1.3 Geben Sie den regulären Ausdruck r_3 für I_3 an ()d.h. $L(r_3) = I_3$.

2. Gegeben sei die Sprache $L = \{ w \in \{a,b\}^* \mid |w|_b = 2\cdot|w|_a\}$, d.h. w ist ein Wort von L, wenn die Anzahl der b’s in w gleich zweimal die Anzahl der a’s in w ist.

2.1 Geben Sie eine kontextfreie Grammatik Γ an, die die Sprache L erzeugt, d.h. L = L(Γ). (Hinweis: Die Regel $S \rightarrow \varepsilon$, wobei S das Startsymbol bezeichnet, ist zugelassen.) Begründen Sie, warum Ihre Grammatik die Sprache L erzeugt.
2.2 Wandeln Sie diese Grammatik in eine Backus-Naur-Form um. (Dabei ist sowohl die BNF- wie EBNF-Schreibweise zulässig.)
2.3 Geben Sie die Ableitung des Wortes w = babbab an.

Aufgabe 4

1. Erläutern Sie die auf Floyd und Hoare zurückgehende Verifikationsmethode. (In Ihrer Antwort müssen Sie mindestens die Begriffe von Zusicherung, schwächste Vorbedingung und Prädikattransformation erklären.)

2. Betrachten Sie folgendes Programmfragment P mit der Vorbedingung
   $V \equiv (x = a \,\&\, y = b)$ und der Nachbedingung $N \equiv (z = a \cdot b)$,
   wobei a,b ganze Zahlen sind.

```
z:=0;
while x<>0 do
begin z:=z+y; x:=x-1 end;
```

2.1 Zeigen Sie, dass die Formel I eine Schleifeninvariante ist.
2.2 Beweisen Sie die partielle Korrektheit von P bezüglich der Vorbedingung V und der Nachbedingung N.
2.3 Was fehlt noch für eine vollständige Verifikation? Terminiert dieses Programm immer? Begründen Sie Ihre Antwort.

1.20 Frühjahr 2000 I

Verwenden Sie zur Beschreibung von Algorithmen bzw. Datentypen eine imperative Programmiersprache wie PASCAL oder MODULA oder einen entsprechenden „Pseudocode“. Alle anzugebenden Algorithmen, Programme, Grammatiken oder Automaten müssen durch Kommentare erläutert werden. Die Qualität der rläuterungen trägt wesentlich zur Bewertung bei.

Einführung:

Die folgenden Aufgaben sind unabhängig voneinander und sollen alle gelöst werden. Sie beziehen sich auf einen Dokumenttyp namens Bericht, der wie folgt definiert ist:

- Ein *Bericht* besteht aus folgenden Komponenten in der gegebenen Reihenfolge:
  - *einem* Titel
  - einer *Autorenliste* (die aber optional sein ist, also auch entfallen kann)
  - einer *Zusammenfassung*
  - einer nichtleeren und endlichen Folge von *Kapiteln.*
- Eine *Autorenliste* ist eine nichtleeren und endlichen Folge von Namen.
- Eine *Zusammenfassung* besteht aus genau einem Absatz.
- Ein *Kapitel* besteht aus folgenden Komponenten in der gegebenen Reihenfolge:
  - *einem Titel*
  - einer nichtleeren und endlichen Folge von *Absätzen*.

Das foldende Dokument ist vom Typ *Bericht*. Die Aufgaben 2 und 5 beziehen sich auf dieses Beispiel.

Dokumentstruktur		Dokument		
Bericht		Titel	>	The Computer in two Chapters
	Autorenliste	Name	>	Anton Ackermann
		Name	>	Berta Bloch
	Zusammenfassung	Absatz	>	This report explains the essentials of computers.
	Kapitel	Titel	>	Hardware
		Absatz	>	You migh t get it into your pocket.
	Kapitel	Titel	>	Software
		Absatz	>	You might get it into your head.
		Absatz	>	Can be pretty hard, too.

Auf welche Weise ein Teiltext eines Dokuments vom übrigen Text abgegrenzt und eindeutig als eine der Dokumentkomponenten *Titel*, *Name* oder *Absatz* klassifiziert werden kann, und in welchem inneren Aufbau er dazu haben muss, spie lt im Folgenden keine Rolle. Die Aufgaben behandeln nicht das Dokument, sondern die Dokumentstruktur, die auf den Dokumentkomponenten *Titel*, *Name* und *Absatz* aufbaut. Ab sofort werden diese drei Dokumentkomponenten mit *t*, *n*, *a* abgekürzt.
Auf dieser Beschreibungsebene wird das obige Dokument durch die Komponentenfolge *Titel Name Name Absatz Titel Absatz Titel Absatz Absatz* repräsentiert, abgekürzt *tnnatataa*.
Also ist *tnnatataa* eine zulässige Komponentenfolge für Dokumente vom Typ *Bericht*.
Dagegen ist die Komponentenfolge *nnatataa* für Dokumente vom Typ *Bericht* nicht zulässig, da ein Bericht nach der obigen Definition mit einem Titel beginnen muss.

Aufgabe 1 Grammatik (EBNF)

Geben Sie eine Grammatik G_1 in Erweiterter Backus-Naur-Form (EBNF) an, die genau die zulässigen Komponentenfolgen für Dokumente vom Typ Bericht erzeugt. Die Grammatik soll so weit wie möglich der verbalen Definition des Dokumenttyps Bericht in der Einführung entsprechen.
Verwenden Sie dazu die Terminalsymbole t, n, a und die folgenden Nichtterminalsymbole: Bericht, Autorenliste, Zusammenfassung, Kapitel
Geen Sie sämtliche Bestandteile der Grammatik G_1 vollständig an.

Aufgabe 2 reguläre Grammatik

1. Geben Sie eine Grammatik G_2 vom Typ 3 (regulär) an, die genau die zulässigen Komponentenfolgen für Dokumente vom Typ *Bericht* erzeugt. Verwenden Sie dazu die Terminalsymbole *t*, *n*, *a*.
   Geben Sie sämtliche Bestandteile der Grammatik G_2 vollständig an.
2. Zeigen Sie, dass die Komponentenfolge *tnnatataa* des Beispieldokuments in der Einführung von der Grammatik G_2 erzeugt wird.

Aufgabe 3 Automat

Geben Sie einen nichtdeterministischen endlichen Automaten *A* an, der genau die zulässigen Komponentenfolgen für Dokumente vom Typ *Bericht* akzeptiert. Verwenden Sie dazu die Terminalsymbolen *t*, *n*, *a*.

1. Definieren Sie den endlichen Automaten in Form eines Diagramms.
2. Definieren Sie den endlichen Automaten in der Form $A = (Z, S, d, z_0, E)$.
   Geben Si ejeden der Bestandteile Z, Σ δ, z_0 und *E* des Automaten *A* vollständig an und nennen Sie jeweils seine Bedeutung.

Aufgabe 4 kontextfreie Grammatik

Der Dokumenttyp *erweiterter_Bericht* sei definiert wie *Bericht*, aber zusätzlich kommt noch unmittelbar nach der *Zusammenfassung* ein *Inhaltsverzeichnis*. Dieses besteht aus der Folge der *Titel* der *Kapitel* des Dokuments. Das folgende Dokument ist vom Typ *erweiterter_Bericht*

Dokumentstruktur		**Dokument**		
Bericht		Titel	>	The Computer in two Chapters
	Autorenliste	Name	>	Anton Ackermann
		Name	>	Berta Bloch
	Zusammenfassung	Absatz	>	This report explains the essentials of computers.
	Inhaltsverzeichnis	Titel	>	Hardware
		Titel	>	Software
	Kapitel	Titel	>	Hardware
		Absatz	>	You might get it into your pocket.
	Kapitel	Titel	>	Software
		Absatz	>	You might get it into your head.
		Absatz	>	Can be pretty hard, too.

Die Komponentenfolge *tnnatttataa* dieses Dokuments ist also zulässig für Dokumente vom Typ *erweiterter_Bericht*. Allerdings hätte ein Dokument, dess Inhaltsverzeichnis die Titel in einer anderen Reihenfolge enthält, die gleiche Komponentenfolge. Auf der Beschreibungsebene der Komponenten ist dieser Unterschied nicht darstellbar. Die Komponentenfolge *tnnattataa* ist dagegen nicht zulässig, da darin zwei *t* vorkommen, die zu *Kapiteln* gehören, aber nur ein *t*, das zum *Inhaltsverzeichnis* gehören kann. Die beiden Anzahlen müssen aber gleich sein.

1. Beweisen Sie, dass die Menge aller zulässigen Komponentenfolgen für Dokumente vom Typ *erweiterter_Bericht* nicht durch eine Grammatik vom Typ 3 (regulär) erzeugt werden kann. Formulieren Sie die Sätze aus, die Ihr Beweis verwendet.
2. Geben Sie ein Grmamatik vom Typ 2 (kontextfrei) an, die genau die Menge aller zulässigen Komponentenfolgen für Dokumente vom Typ *erweiterter_Bericht* erzeugt.

Aufgabe 5 Datenstrukturen

Es stehe ein Datentyp Komponente mit folgenden Operationen zur Verfügung:

istTitel(k : Komponente) : BOOLEAN	liefert TRUE gdw. k ein Titel ist
istName(k : Komponente) : BOOLEAN	liefert TRUE gdw. k ein Name ist
istAbsatz(k : Komponente) : BOOLEAN	liefert TRUE gdw. k ein Absatz ist

Weiter stehe ein Datentyp Komponentenfolge mit folgenden Operationen zur Verfügung:

leer(f : Komponentenfolge) : BOOLEAN	liefert TRUE gdw. f die leere Komponentenfolge ist.
komponente1(f : Komponentenfolge) : Komponente	liefert die erste Komponente in f; undefiniert, falls f leer ist.
rest(f : Komponentenfolge) : Komponentenfolge	liefert die Teilfolge von f ab der zweiten Komponente; undefiniert, falls f leer ist.

1. Definieren Sie eine Funktionsprozedur anzahlTitel_rek(f : Komponentenfolge) : INTEGER, die die Anzahl der Titel in f rekursiv berechnet.
2. Definieren Sie eine Funktionsprozedur anzahlTitel_iter(f : Komponentenfolge) : INTEGER, die die Anzahl der Titel in f iterativ berechnet.
3. Elemente des Datentyps Komponentenfolge sollen als einseitig verkettete Listen mit Verbunden und Verbundzeigern repräsentiert werden. Geben Sie eine Definition des Datentyps Komponentenfolge für diese Repräsentation an. Definieren Sie dazu

passende Funktionsprozeduren für die angegebenen drei Operationen auf dem Datentyp Komponentenfolge.

4. Unabhängig von den obigen Vorgaben seien Datentypen Titel, Name und Absatz gegeben, mit denen die entsprechenden Komponenten eines Dokuments repräsentiert werden können.
   Definieren Sie einen Datentyp Bericht und alle weiteren benötigten Datentypen, um die gesamte Dokumentstruktur von Dokumenten vom Typ Bericht repräsentieren zu können. Die Definitionen sollen soweit wie möglich der verbalen Definition des Dokumenttyps Bericht in der Einführung (siehe Frühjahr 2000 I im Teil zu Grammatiken und Automaten) entsprechen. Halten Sie sich dabei an die Konvention, dass Bezeichner von Typen mit Großbuchstaben anfangen und andere Bezeichner mit Kleinbuchstaben.
   Sei bericht eine Variable des von Ihnen definierten Datentyps Bericht, deren Wert die Struktur des Beispieldokuments in der Einführung repräsentiert. Geben Sie einen Ausdruck an, mit dem auf den ersten Absatz des zweiten Kapitels zugegriffen wird.

1.21 Frühjahr 2000 II

Aufgabe 1

1. Zeigen Sie, dass die folgende Sprache L nicht regulär ist:
   $L = \{a^n b a^m b a^{n+m} \mid n, m \geq 1\}$.
2. Definieren Sie den Begriff eines Kellerautomatens.
3. Beschreiben Sie einen Kellerautomaten M, der die Sprache
   $L = \{a^n b a^m b a^{n+m} \mid n, m \geq 1\}$
   akzeptiert.

Aufgabe 2

Sei L die Sprache $L = \{w \in \{0,1\}^* \mid$ der Teilstring 011 ist nicht in w enthalten$\}$.

1. Geben Sie das Zustandsdiagramm an für einen endlichen Automaten, der L akzeptiert.
2. Geben Sie eine reguläre Grammatik G an, die L erzeugt.
3. Geben Sei einen regulären Ausdruck für die Sprache L an.

Aufgabe 3

Sei $G = (V, \Sigma, R, S)$ eine kontextfreie Grammatik, wobei $V = \{A, S\}$, $\Sigma = \{a, b\}$ und R die Menge der Regeln oder Produktionen

$S \rightarrow aAS \mid a \qquad A \rightarrow SbA \mid SS \mid ba.$

Geben Sie einen Parsebaum und eine Linksableitung an für das Wort aabbaa.

Aufgabe 4

Betrachten Sie die durch folgende Produktionen definierte kontextfreie Grammatik G mit den Regeln:

$S \rightarrow aAB \qquad A \rightarrow aBBC \qquad A \rightarrow a$

$B \rightarrow bCC \qquad B \rightarrow b \qquad C \rightarrow c$

Geben Sie einen regulären Ausdruck für die Sprache L(G) = {w ∈ {a, b, c}* | $S \Rightarrow_G {}^*w$} an; d.h. L(G) ist die Menge aller Wörter, die nur aus Terminalsymbolen bestehen und die von der Grammatik abgeleitet werden können.

Aufgabe 5

1. Geben Sie eine genaue Formulierung des Pumping Lemmas für kontextfreie Sprachen an.
2. Zeigen Sie mit dem Pumping Lemma, dass die folgende Sprache nicht kontextfrei ist: $\{a^i b^j \mid i, j \in N, j = i^2\}$.

Aufgabe 6

Ist die folgende Funktion F: {0, 1, ..., 9}* → {0, 1} rekursiv? Die Eingabe x wird als Dezimalzahl interpretiert. Es ist f(x) = 1 genau dann, wenn es in der Dezimaldarstellung von x einen geschlossenen Block von mindestens x Siebenen gibt. Beweisen Sie Ihre Antwort.

Aufgabe 7

1. Definieren Sie den Begriff einer kontextsensitiven Grammatik.
2. Sei $L = \{a^n b^m c^n \mid n,m \geq 0, m \leq n\}$. Geben Sie eine kontextsensitive Grammatik für L an.
3. Geben Sei eine Ableitung des Wortes $a^3b^2c^3$ mittels Ihrer Grammatik an.

1.22 Herbst 2000 I

Aufgabe 1 Lexikalischer Vergleich

Listen von Listen ganzer Zahlen kann man lexikalisch ordnen. Eine Liste $[x_1, ..., x_m]$ heißt lexikalisch kleiner als eine Liste $[y_1, ..., y_n]$, falls es ein i mit $0 \leq i \leq \min(m,n)$ so gibt, dass gilt:

$x_k = y_k$ für alle k = 1, ..., i und
entweder $i = m < n$ oder $x_{i+1} < y_{i+1}$.

Wählen Sie für die folgenden Teilaufgabe 1. bis 4. eine beliebige (jedoch für alle Teilaufgaben dieselbe) Programmiersprache.

1. Geben Sie einen Datentyp für Listen von ganzen Zahlen und einen Datentyp für Listen von Listen von ganzen Zahlen an.
2. Programmieren Sie den lexikalischen Vergleich `lex` auf Listen von ganzen Zahlen als eine Funktion oder Prozedur.
3. Schreiben Sie eine Prozedur oder eine Funktion `lex_insert`, die eine Liste `l` von ganzen Zahlen in eine lexikalisch geordnete Liste `ll` von Listen von ganzen Zahlen an die korrekte Stelle einsortiert.
   z.B. liefert die Anwendung von `lex_insert` auf die Argumente [1,3] und [[1,2], [1,7], [4]] die Liste [[1,2], [1,3], [1,7], [4]] als Ergebnis.
4. Schreiben Sie eine Prozedur oder Funktion `insert_sort`, die eine Liste von Listen von Zahlen `ll` lexikalisch sortiert. Die Prozedur oder die Funktion soll `lex_insert` verwenden.

Aufgabe 2 Berechenbarkeit

Seien $A,B \subseteq N_0$ rekursiv aufzählbar. Beweisen Sie die folgenden Behauptungen:

1. $A \cup B$ ist rekursiv aufzählbar.
2. A x B ist rekursiv aufzählbar.

Aufgabe 3 Sprachen und Automaten

1. Zeigen Sie, dass die Sprache

   $L = \{a^m b^n$: (m ist durch 3 teilbar und n ist ungerade) oder (m ist gerade und n = 0)$\}$

   regulär ist. Geben Sie den regulären Ausdruck an.

2. Konstruieren Sie einen endlichen nichtdeterministischen Automaten, der L erkennt.

3. Konstruieren Sie einen endlichen deterministischen Automaten, der L erkennt.

4. Sei $L_1 = \{a^m b^n$: m,n > 0 und m/n = 2/3$\}$. Ist L_1 vom Typ Chomsky-2, ist L_1 zugleich vom Typ Chosky-3? Beweisen Sie Ihre Behauptung.

Aufgabe 4 Programmierung

1. Erläutern Sie die auf Floyd und Hoare zurückgehende Verifikationsmethode. (In Ihrer Antwort müssen Sie mindestens die Begriffe von Zusicherung, Vor- und stärkste Nachbedingung erklären.)

2. Betrachten Sie folgendes Programmfragment mit der Vorbedingung $V \equiv (s=0 \ \& \ i=0)$ und der Nachbedingung $N \equiv (s = (n+1)*n/2)$, wobei i, n und s ganze Zahlen sind.

```
while i<=n do
  begin
       s:=s+i;
       i:=i+1
  end.
```

   2.1 Zeigen Sie, dass $i \leq n+1 \ \& \ s = (i-1)*i/2$ eine Schleifeninvariante ist, und beweisen Sie, dass diese erhalten bleibt.
   2.2 Beweisen Sie die Nachbedingung.
   2.3 Terminiert das Programm immer? Beweisen Sie Ihre Antwort.

1.23 Herbst 2000 II

Aufgabe 1

Geben Sie die Zustandsdiagramme für deterministische endliche Automaten für die folgenden Sprachen an.

1. Die Menge der Wörter $w \in \{a,b\}^*$, deren Länge $|w|$ entweder durch 2 oder 3 ohne Rest teilbar ist.

2. Sei L die Sprache der Aufgabe 1.; d.h.
   $L = \{ w \in \{a,b\}^* : |w| \text{ ist entweder teilbar durch 2 oder 3}\}$
   Sei $\overline{L} = \{a,b\}^* \setminus L$ das Komplement von L. Geben Sie einen regulären Ausdruck für die Sprachen L und $\overline{L}$ an.

Aufgabe 2

Beweisen Sie, dass die Sprache $\{a^{2m}b^m : m \in N\}$ kontextfrei, aber nicht regulär ist.

Aufgabe 3

Skizzieren Sie in Pseudocode die Implementierung einer Operation, die aus einer verketteten Liste die Elemente entfernt, die mehrfach vorkommen. Geben Sie die notwendigen Datenstrukturen an, wobei die verkettete Liste mit Zeigern (Pointern) implementiert werden soll.

Aufgabe 4

In dieser Aufgabe werden in Pseudocode die Datenstruktur und Implementierung von bestimmten Operationen für binäre Bäume gegeben.

1. Definieren Sie in Pseudocode die Datenstruktur für die Implementierung von binären Bäumen (`BTree`), wobei Blätter dadurch dargestellt werden, dass `left` und `right` den Wert `null` haben.
2. Implementieren Sie eine Methode
   ```
   void printInorder()
   ```
   für `BTree`, die den Baum in Infixdarstellung ausgibt.
3. Geben Sie den Code an, der die Variable `t` mit Datentyp `BTree` mit einem Baum belegt, dessen Infixdarstellung `2+4·3` ist.

Aufgabe 5

Geben Sie eine kontextfreie Grammatik (oder BNF) an für die Menge der Palindrome gerader Länge über {a,b,c} wobei die Mitte durch. markiert ist, also z.B. ab.ba . Skizzieren Sie die Implementierung eines *recursive descent* Parsers in Pseudocode.

Aufgabe 6

Diese Aufgabe bezieht sich auf das Sortierungsverfahren `quicksort`.

1. Geben Sie Pseudocode für `quicksort` an.
2. Simulieren Sie die Schritte von `quicksort` auf der Eingabe `4,2,1,3,8,5,0,6,7`.
3. Was ist das *worst-case*-Verhalten von `quicksort`?
4. Für welche Eingaben zeigt `quicksort` das *worst-case*-Verhalten?
5. Was ist das *average-case*-Verhalten von `quicksort`?

1.24 Frühjahr 2001 I

Aufgabe 1

Gegeben sei das Alphabet $\Sigma = \{0,1\}$ und die Sprache

$$L = \{1\ 0^i\ 1_j\ 0 \mid i \text{ und } j \text{ sind gerade, } i,j \geq 0\}.$$

1. Geben Sie einen regulären Ausdruck mit Sprache L an.
2. Geben Sie eine rechtslineare Grammatik an, die L erzeugt.
3. Konstruieren Sie einen nichtdeterministischen Automaten ohne Leerübergänge, der die Sprache L akzeptiert.
4. Konstruieren Sie einen minimalen deterministischen Automaten, der L akzeptiert.

Aufgabe 2

Gegeben sei die folgende Grammatik G zur Erzeugung arithmetischer Ausdrücke:

$G = (\{E\}, \{a, b, +, *, (,)\}, P, E)$ wobei
$P = \{E \rightarrow E+E, E \rightarrow E*E, E \rightarrow (E), E \rightarrow a, E \rightarrow b\}$.

1. Geben Sie eine Ableitung für den Ausdruck (a+b)*(a*b) an.
2. Zeigen Sie, dass die Grammatik G nicht eindeutig (d.h. ambig) ist.
3. Gesucht ist eine eindeutige Grammatik G', die dieselbe Sprache wie G hat.
   Vervollständigen Sie dazu den folgenden Ansatz durch Hinzunahme weiterer Produktionsregeln:

   $G' = (\{E, F, G\}, \{a, b, +, *, (,)\}, P', E)$ wobei
   $P' = \{E \rightarrow E+F, E \rightarrow F, ...\}$.

Aufgabe 3

Beim Beweis der folgenden Aufgaben ist jeweils das Schema der primitiven Rekursion mit geeigneten primitiv rekursiven Funktionen $g: N^n \rightarrow N$ und $h: N^{n+2} \rightarrow N$ anzuwenden.
(N bezeichnet dabei die Menge der natürlichen Zahlen einschließlich 0.)

1. Zeigen Sie, dass die folgenden Funktionen primitiv rekursiv sind:

iszero: $N \rightarrow N$,	iszero(0) = 1,	iszero(x) = 0 falls $x \neq 0$.
even: $N \rightarrow N$,	even(x) = 1 falls x gerade,	even(x) = 0 falls x ungerade.
case: $N^3 \rightarrow N$,	case(x,y,z) = y falls x=0,	case(x,y,z) = z falls $x \neq 0$.

2. Zeigen Sie:
   Ist $f: N \rightarrow N$ primitiv rekursiv, dann ist auch $r: N^2 \rightarrow N$ mit r(x,i) = f(x) primitiv rekursiv. Dabei bezeichnet f^0 die Identität und für i>0 bezeichnet f^i die i-malige Hintereinanderausführung f(...f(x)...) von f.

Aufgabe 4

Beweisen Sie mit Hilfe der Zusicherungsmethode, dass das folgende Programm bezüglich der eingefügten Zusicherungen partiell korrekt ist:

```
{n ≥ 0}
x := n; z := 1;
{x ≥ 0 ∧ z = 2^(n-x)}
WHILE x > 0 DO
     z := 2*z; x := x-1;
END;
{z = 2^n}
```

Aufgabe 5

Die folgenden Aufgaben können in Pascal, Modula-2, C, C++ oder Java gelöst werden.

1. Definieren Sie einen Datentyp (oder eine Klasse) mit Namen „Bin Tree“ zur Darstellung binärer Bäume mit ganzen Zahlen als Knotenmarkierungen.
2. Ein binärer Baum t ist sortiert, wenn t leer ist, oder wenn seine Wurzel n größer als alle im linken Teilbaum l von t vorkommenden Knoten und kleiner gleich allen im rechten Teilbaum r von t vorkommenden Knoten ist und wenn l und r ebenfalls sortiert sind. Implementieren Sie eine Operation mit Namen „isSorted“, die für einen gegebenen binären Baum feststellt, ob er sortiert ist.
3. Implementieren Sie eine Operation mit Namen „insert“, die eine ganze Zahl x so in einen binären Baum einfügt, dass dieser nach Ausführung der Operation wieder sortiert ist.

1.25 Frühjahr 2001 II

Aufgabe 1

Für eine nicht-leere Menge von Personen sei gegeben, wer mir wem
- eng verwandt,
- weitläufig verwandt,
- bekannt (aber nicht verwandt),
- weder verwandt noch bekannt

ist. Davon ausgehend sollen folgende Aufgaben gelöst werden:

1. Für eine beliebige Person soll bestimmt werden, mit wie vielen anderen Personen sie eng verwandt ist.
2. Es sollen diejenige(n) Person(en) mit den meisten Bekannten und weitläufig Verwandten bestimmt werden.
3. Es soll festgestellt werden, ob es drei Personen gibt, die untereinander weder verwandt noch bekannt sind.

Geben Sie einen geeigneten Datentyp zur Darstellung der gegebenen Personenbeziehungen sowie Algorithmen zur Lösung der Aufgaben 1., 2. und 3. an.

Aufgabe 2

Gegeben sei die Grammatik Γ mit der Menge {a,b} von Terminalzeichen, dem Startsymbol S als einzigem Nicht-Terminalzeichen und den vier Produktionsregeln

$S \rightarrow a$ $\quad$ $S \rightarrow aSb$
$S \rightarrow aS$ $\quad$ $S \rightarrow bSa$

sowie der reguläre Ausdruck

$R = a\,(aa^*b + baaa^*)$.

L(Γ) sei die von Γ erzeugt Sprache, L(R) die durch R beschriebene Sprache. Schließlich sei L die Menge aller nicht-leeren Zeichenreihen über {a,b}, in denen a öfter als b vorkommt.

1. Geben Sie eine Ableitung der Zeichenreihe aababaaab in Γ an.
2. Beweisen Sie: $L(R) \underset{\neq}{\subset} L(\Gamma) \underset{\neq}{\subset} L$.
3. Geben Sie eine kontextfreie Grammatik (Typ 2) mit höchstens fünf Produktionsregeln an, die L(R) erzeugt.
4. Beweisen Sie, dass L keine reguläre Sprache ist.
5. Geben Sie einen nichtdeterministischen Kellerautomaten an, der die Sprache L(Γ) akzeptiert.
6. Geben Sie eine deterministische Turingmaschine an, die die Sprache L akzeptiert.

Aufgabe 3

Durch den rekursiven Algorithmus

```
F ≡ function f(n,m:nat) nat:
      if n<m then n else f(n-m,m+1) endif
```

ist eine Funktion f: $N_0 \times N_0 \rightarrow N_0$ definiert.

1. Bestimmen Sie den Wert von f (31, 3).

2. Beweisen Sie: Der Algorithmus F terminiert für alle $n,m \in N_0$.

3. Geben Sie einen iterativen Algorithmus an, der f (n, m) für beliebige $n,m \in N_0$ berechnet.

4. Beweisen Sie: Zu jedem Paar $(n,m) \in N_0 \times N_0$ gibt es ein $k \in N_0$ mit
   $k * m \leq n$ und
   $2 * f(n,m) + k^2 + k * (2 * m - 1) = 2 * n$.

5. Welche der folgenden Aussagen ist richtig?
5.1 Die Zeitkomplexität von F ist $O(n)$.
5.2 Die Zeitkomplexität von F ist $O(m)$.
5.3 Die Zeitkomplexität von F ist $O(n/m)$.
   Begründen Sie Ihre Antworten.

6. Ist die Funktion f primitiv-rekursiv? Begründen Sie Ihre Antwort.

1.26 Herbst 2001 I

Aufgabe 1

Gegeben sei die Sprache $L = \{a^n b^m c^n \mid n > 0, m > 0\}$ über $\Sigma = \{a, b, c\}$.

1. Ist L regulär? (Begründung!)
2. Ist L kontextfrei? (Begründung!)
3. Ist L kontextsensitiv? (Begründung!)
4. Ist L entscheidbar? (Begründung!)

Aufgabe 2

Konstruieren Sie einen vollständigen deterministischen erkennenden Automaten, der genau die Wörter der Form a^n mit „n ist weder durch 4 noch durch 6 teilbar“ akzeptiert.

Aufgabe 3

Diskutieren Sie die Frage, ob die Menge
$M = \{ x \in N \mid$ Die Ziffernfolge der Dezimaldarstellung von x kommt (als Teilwort) in der Dezimaldarstellung der Zahl π vor$\}$
entscheidbar ist. (Beispiel: $\pi = 3{,}141592...$ enthält 159, also $159 \in M$)

Aufgabe 4

Für die Simulation der Stapelalgebra mit WHILE-Programmen benötigt man eine Funktion code: $N^* \rightarrow N$, die endliche Folgen natürlicher Zahlen codiert. Sei durch stapel = $[x_1, ..., x_k]$ ein Stapel natürlicher Zahlen mit oberstem Element x_1 gegeben.

1. Definieren Sie eine geeignete Funktion code und erläutern Sie code(stapel) (Hinweis: Eine Paarungsfunktion ist nützlich!).
2. Berechnen Sie code([2,4]).
3. Geben Sie WHILE-Programme an, die die Stapel-Operationen *empty*, *push*, *pop*, *top*, *isempty* implementieren.

Aufgabe 5

Neben der Turingmaschine mit einem beidseitig unendlichen Band gibt es auch eine Variante, die nur einseitig (nach rechts) unendliches Band zur Verfügung hat.

1. Präzisieren Sie die Definition einer solchen „einseitigen“ Turingmaschine.
2. Zeigen Sie die Äquivalenz der beiden Varianten von Turingmaschinen.

Aufgabe 6

Sei F die Formel $(\neg (A \rightarrow B) \vee \neg (\neg A \vee C)) \wedge \neg A$

1. Ist F erfüllbar?
2. Ist F eine Tautologie?

1.27 Herbst 2001 II

Aufgabe 1

Für eine Klasse N von „Nimm“-Spielern gelten folgende gemeinsame Spielregeln: Jedes Spiel wird von zwei Spielern A und B mit Spielsteinen auf einem Spielbrett gespielt und beginnt in einer Anfangsstellung, in der $n > 0$ Spielsteine auf dem Brett sthen. A und B machen der Reihe nach abwechselnd jeweils einen Spielzug, wodurch sich eine Folge von Nachfolge-Stellungen als Spielablauf ergibt. A macht den ersten Spielzug. In jeder Stellung stehen jedem Spieler höchstens drei verschiedene Spielzüge zur Verfügung. Jeder Spielzug verringert die Anzahl der Spielsteine auf dem Brett. Jede Stellung ist charakterisiert durch die Angaben,

- welcher Spieler am Zug ist,
- wieviele Spielsteine auf dem Brett stehen,
- ob A oder B gewonnen hat oder ob noch kein Spieler gewonnen hat.

Ein Spielablauf endet, wenn ein Spieler gewonnen hat oder wenn keine Spielsteine mehr auf dem Brett stehen. Die Länge eines Spielablaufs ist die Anzahl der dabei durchgeführten Spielzüge.

Jedes Spiel der Klasse N definiert durch seine zusätzlichen Regeln (über Art der Spielzüge, Art der Gewinnstellungen u.a.) eine Menge S aller seiner möglichen Spielabläufe.

Geben Sie einen geeigneten Datentyp zur Darstellung derartiger Mengen S sowie Algorithmen an, die für eine derartige Darstellung eines Spiels folgende Aufgaben lösen:

1. Bestimmung der maximal auftretenden Länge eines Spielablaufs.

2. Bestimmung der Anzahl der Spielabläufe, in denen der Spieler A gewinnt.

3. Klärung der Frage, ob es einen Spielablauf gibt, in dem eine Stellung erreicht wird, in der alle möglichen Spielfortsetzungen irgendwann zum Gewinn des Spielers führen, der gerade nicht am Zug ist.

Aufgabe 2

Gegeben sei das Alphabet $\Sigma = \{0,1\}$ und

$$Z = \{w \in \Sigma^+ \mid w = 0 \text{ oder } w \text{ beginnt mit dem Zeichen } 1\}.$$

Jedes $w = w_m ... w_2 w_1 w_0 \in Z$ kann als Binärdarstellung der natürlichen Zahl

$$N(w) = w_m 2^m + ... + w_2 2^2 + w_1 2 + w_0$$

aufgefasst werden. Umgekehrt gibt es zu jedem $n \in \mathrm{N}_0$ eine derartige Binärdarstellung $W(n) \in Z$ (d.h. $N(W(n)) = n$).

Weiter sei die Grammatik $G = (V, \Sigma, P, S)$ mit $V = \{S, A\}$ und der Menge P der Produktionen

$$S \rightarrow 1 \mid 1A$$
$$A \rightarrow 1 \mid 1A \mid 0S$$

gegeben. $L(G)$ sei die von G erzeugte Sprache.

1. Geben Sie eine reguläre Grammatik (Typ 3) an, die Z erzeugt.

2. Geben Sie eine Ableitung der Zeichenreihe `11101011011` in G an.

3. Beweisen Sie: $L(G) \subseteq Z$.

4. Beweisen Sie: Für jedes $w \in L(G)$ ist $N(w)$ ungerade.

5. Geben Sie einen deterministischen endlichen Automaten an, der die Sprache $L(G)$ akzeptiert.

6. Ist die Menge $\{n \in \mathrm{N}_0 \mid W(n) \in L(G)\}$ entscheidbar? Begründen Sie Ihre Antwort.

Sei nun $N \subseteq \mathrm{N}_0$ die Menge aller natürlichen Zahlen

$n = 2^p - 2^q - 1$ mit $p \geq 3$ und $0 < q < p-1$

(Beispiele: $23 = 2^5 - 2^3 - 1$, $59 = 2^6 - 2^2 - 1$, also $23, 59 \in \mathrm{N}$)

und $Z_N = \{w \in Z \mid w = W(n), n \in N\}$.

7. Beweisen Sie: $W(23) \in L(G)$ und $W(59) \in L(G)$.

8. Beweisen Sie: $Z_N \subset\!\!\!\!\neq L(G)$.

9. Geben Sie einen regulären Ausdruck an, der Z_N beschreibt.

10. Beweisen Sie: Es gibt eine Teilmenge M von N derart, dass die Menge
    $Z_M = \{w \in Z \mid w = W(n), n \in M\}$
    keine reguläre, jedoch eine kontextfreie (Typ 2-) Sprache ist.

Aufgabe 3

Mit deterministische Turingmaschinen (DTM) lassen sich Funktionen $f: \mathrm{N}_0^k \rightarrow \mathrm{N}_0$ berechnen.

1. Präzisieren Sie formal, was es bedeutet, dass eine DTM eine derartige Funktion berechnet mit der Vorgabe, dass alle auftretenden natürlichen Zahlen m in Binärdarstellung $W(m)$ (wie in Herbst 2001 II Aufgabe 2) dargestellt sein sollen.

2. Geben Sie eine DTM an, die in der unter 1. angegebenen Weise die Funktion
   $s(n) = n+1$
   berechnet.

   Für $n \in \mathrm{N}_0$ sei n_b die Länge der Zeichenreihe $W(n)$. Welche der folgenden Aussagen über die Zeitkomplexität der Berechnung von $s(n)$ ist richtig?
   2.1 Die Zeitkomplexität der Berechnung ist $O(n_b)$.
   2.2 Die Zeitkomplexität der Berechnung ist $O(\log n_b)$.

2.3 Die Zeitkomplexität der Berechnung ist O (log n).
Begründen Sie Ihre Antworten.

3. Gibt es eine andere Zahldarstellung als die Binärdarstellung, mit der die Berechnung von $s(n)$ konstante Zeitkomplexität hat? Begründen Sie Ihre Antwort.

2 Datenbanksysteme, Betriebssysteme, Rechnernetze, Rechnerarchitektur

2.1 Herbst 1989

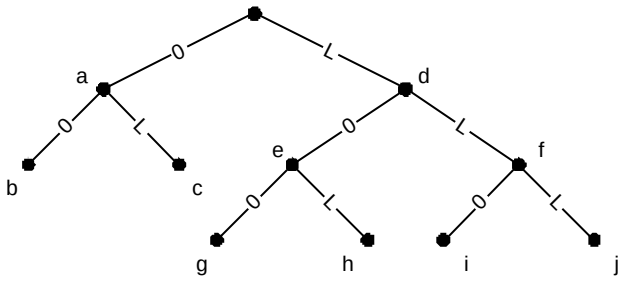
Aufgabe 1 Binärcodierung

Für einen endlichen nicht-leeren Zeichenvorrat Z sei eine Binärcodierung

$$C : Z \rightarrow \{O , L\}^*$$

gegeben.

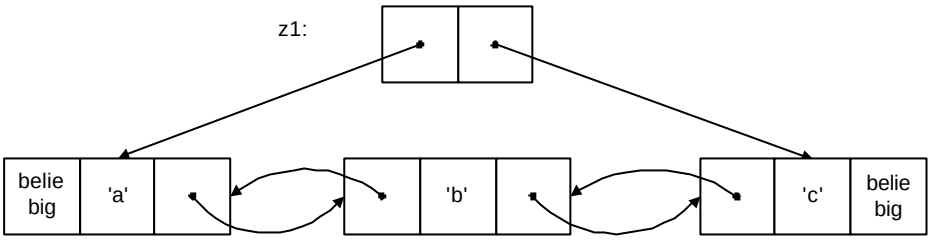
1. Geben Sie eine Datenstruktur (durch Typdefinitionen) an, mit deren Hilfe ein beliebiger Binärcode c im Hinblick auf die folgende Teilaufgabe geeignet dargestellt werden kann.
2. Schreiben Sie eine Funktionsprozedur *fano*, mit deren Hilfe festgestellt werden kann, ob ein Binärcode c der Fano-Bedingung genügt. Die Funktionsprozedur *fano* soll dabei folgender Spezifikation genügen:
   2.1 Eingabeparameter: c vom Typ CODE
   2.2 Der Funktionswert ist vom Typ BOOLEAN und hat den Wert TRUE genau dann, wenn c der Fano-Bedingung genügt.
   Hinweise:
   - Der Typ CODE muss in Ihrer Antwort zu 1 definiert werden.
   - Ein Code erfüllt die Fano-Bedingung genau dann, wenn in ihm kein Codewort Anfang eines von ihm verschiedenen Codewortes ist.
3. Der Zeichenvorrat Z = {a, b, c, d, e, f, g, h, i, j} lässt sich durch Wörter aus {O, L}* binär codieren, die höchstens die Länge drei haben. Ein Beispiel dafür ist durch den folgenden Code-Baum gegeben:



   Zeigen Sie, dass es keine 3-stellige Binärcodierung (d.h. mit Codewortlänge ≤ 3) für Z gibt, die die Fano-Bedingung erfüllt.

Aufgabe 2 Datenstrukturen

Zeichenreihen über dem endlichen Alphabet A = (‚a', ‚b', ..., ‚z') sollen durch vorwärts und rückwärts verkette Listen dargestellt werden, für deren Anfang und Ende Anker (Verweise) vorgesehen sind. Die Darstellung der Zeichenreihe z1=‚abc' lässt sich also beispielsweise folgendermaßen graphisch veranschaulichen:



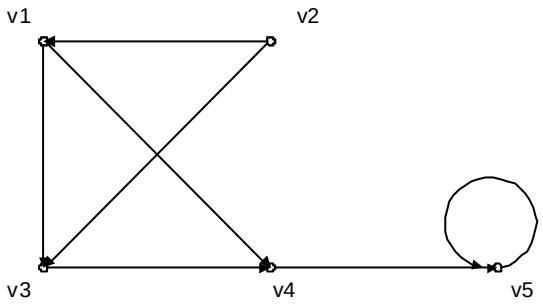
1. Geben Sie eine Datenstruktur mit Hilfe geeigneter Typdefinitionen an, durch die sich die o.a. Darstellung von Zeichenreihen realisieren lässt.
2. Wie kann die leere Zeichenreihe ε dargestellt werden?
3. Schreiben Sie eine Funktionsprozedur *istepsilon*, die feststellt, ob eine Zeichenreihe gleich ε ist, also folgender Spezifikation genügt:
   3.1 Eingabeparameter: x1 vom Typ ZEICHENREIHE
   Hinweis: ZEICHENREIHE muss als Typ in Ihrer Antwort zu 1. definiert sein.
   3.2 Der Funktionswert ist vom Typ BOOLEAN und ist genau dann gleich TRUE, wenn x1 = ε ist.
4. Schreiben Sie zwei Funktionsprozeduren *anfang* und *teil*, die feststellen, ob x Anfangs- bzw. Teilzeichenreihe einer Zeichenreihe y ist. Die entsprechenden Spezifikationen lauten:
   4.1 Eingabeparameter (für beide Funktionsprozeduren):
   x, y vom Typ ZEICHENREIHE
   4.2 Der Funktionswert ist jeweils vom Typ BOOLEAN. Er hat den Wert TRUE für die Funktion *anfang* bzw. *teil* genau dann, wenn x Anfangs- bzw. Teilzeichenreihe von y ist.

   Hinweise:
   - x ist genau dann Anfangszeichenreihe von y , wenn es ein y' so gibt, dass y =xy' ist.
   - x ist genau dann Teilzeichenreihe von y, wenn es ein y' und ein y'' so gibt, dass y = y'xy'' ist.
   - Es ist empfehlenswert, erst die Funktionsprozedur *anfang* zu schreiben und dann in der Funktionsprozedur *teil* die Funktion *anfang* zu verwenden. Dabei darf auch die in 3. spezifizierte Funktion *istepsilon* verwendet werden.

5. Ein Palindrom ist eine Zeichenreihe, die mit ihrer rückwärts gelesen identisch ist. Beispiele sind
   ‚*otto*', ‚*reliefpfeiler*', ε (leere Zeichenreihe).
   Schreiben Sie eine Funktionsprozedur *palin*, die feststellt, ob eine Zeichenreihe ein Palindrom ist, also folgender Spezifikation genügt:
   5.1 Eingabeparameter: x1 vom Typ ZEICHENREIHE
   5.2 Der Funktionswert ist vom Typ BOOLEAN und ist genau dann gleich TRUE, wenn x1 ein Palindrom ist.
   Hinweis: In der Funktionsprozedur *palin* darf die unter 3. spezifizierte Funktion istepsilon verwendet werden.

Aufgabe 3 Gerichtete Graphen

Ein endlicher gerichteter Graph G = (V, E) mit der Knotenmenge V = $\{v_1, v_2, ..., v_n\}$ und der Menge gerichteter Kanten E $\subseteq$ V x V kann u.a. durch seine direkte graphische Darstellung und durch seine Adjazenzmatrix dargestellt werden. Ein Beispiel:



Dieser direkt dargestellte Graph mit V = {v_1, v_2, v_3, v_4, v_5} und E = {(v_1, v_3), (v_1, v_4), (v_2, v_1), (v_3, v_2), (v_4, v_3), (v_5, v_4), (v_5, v_5)} lässt sich auch durch seine Adjazenzmatrix

$$A = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

darstellen. Allgemein gilt für die Komponenten a_{ij} der Adjazenzmatrix A:

$$a_{ij} = \begin{cases} 1 & \text{falls es eine Kante von } v_i \text{ nach } v_j \text{ gibt} \\ 0 & \text{sonst} \end{cases}$$

Hat ein gerichteter Graph die Eigenschaft, dass von wenigstens einem Knoten zwei gerichtete Kanten ausgehen, so sagt man, dass er den Verzweigungsgrad 2 hat. Der angegebene Beispiel-Graph hat also den Verzweigungsgrad 2. Solche Graphen können in direkter Darstellung durch einen Verweis auf ein Geflecht von Verbunden der Art *knoten* entsprechend folgender Typ-Vereinbarung aufgebaut werden:

```
TYPE  graph = ↑ knoten;
      knoten = RECORD inhalt: INTEGER;
                      nachf1, nachf2: graph
               END;
```

Mit VAR g : graph sei nun eine Variable gegeben, die auf einen endlichen gerichteten Graphen mit n Knoten und mit Verzweigungsgrad 2 verweist.

1. Geben Sie einen terminierenden Algorithmus an, der die Adjazenzmatrix von y berechnet.
   Hinweis:
   Es wird nicht verlangt, dass dafür ein Programm geschrieben wird; der Algorithmus soll aber klar und korrekt beschrieben werden.
2. Als Wegematrix W zu einem endlichen gerichteten Graphen G bezeichnet man die Matrix, die durch ihre Komponente w_{ij} angibt, ob es in G einen gerichteten Kantenzug von v_i nach v_j gibt. W = $(w_{ij})_{1 \le i,j \le n}$ ist also folgendermaßen definiert:

$$W_{ij} = \begin{cases} 1 & \text{falls ein gerichteter Kantenzug von } v_i \text{ nach } v_j \text{ existiert} \\ 0 & \text{sonst} \end{cases}$$

Zu dem o.a. Graphen ist beispielsweise

$$W = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Die zugehörige Wegematrix.

Schreiben Sie eine Prozedur *weg*, die zur Adjazenzmatrix A eines endlichen gerichteten Graphen mit n Knoten und mit einem beliebigen Verzweigungsgrad (≤ n) die Wegematrix W berechnet. Die Prozedur *weg* soll also folgender Spezifikation genügen:

2.1 Eingabeparameter: Adjazenzmatrix a vom Typ ARRAY [1..n,1..n] OF 0..1

2.2 Ausgabeparameter: Wegematrix w vom Typ ARRAY [1..n,1..n] OF 0..1
Dabei soll w die Wegematrix zu dem durch a dargestellten gerichteten endlichen Graphen sein.

2.2 Frühjahr 1990

Aufgabe 1

1. Welche zwei Arten von Unterbrechungen unterscheidet man auf Grund ihrer Auslösung? Erklären Sie diese und geben Sie jeweils drei Beispiele für jede Art an!
2. In welcher Situation im Befehlszyklus (Mikroprogramm des Rechenkerns) wird abgefragt, ob ein Unterbrechungswunsch vorliegt? In welchem Fall wird dieser zurückgestellt? Für welche Unterbrechungsart kann er nicht zurückgestellt werden?
3. Bei einer Unterbrechung wird außer dem Unterbrechungswunsch noch weitere Unterbrechungsinformation übertragen, die von der speziellen Unterbrechung abhängig ist. Auf welche Weise wird diese Information dem Betriebssystem übermittelt? Geben Sie die Unterbrechungsinformation für drei verschiedene Unterbrechungen Ihrer Wahl an!

Aufgabe 2

Geben Sie diejenigen der nachfolgend genannten Bedingungen (1) bis (17) an, die gleichzeitig erfüllt sein müssen, damit bei der Vergabe von Betriebsmitteln (BM) eine Verklemmung entstehen kann.

(1) BM sind entziehbar
(2) BM sind nicht entziehbar
(3) BM-Zugriff wird nicht synchronisiert
(4) BM-Zugriff erfolgt durch Synchronisationsprozess
(5) BM sind zugriffsbeschränkt
(6) BM sind gemeinsam verwendbar

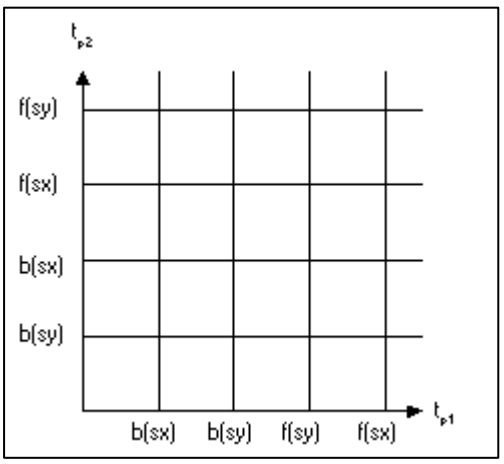
(7) Prozesse fordern BM in beliebiger Reihenfolge an
(8) BM dürfen nicht aus mehreren unabhängig vergebbaren Einheiten bestehen
(9) BM sind wiederverwendbar
(10) BM sind nicht wiederverwendbar
(11) Prozesse, die auf Zuteilung weiterer BM warten, geben bereits belegte BM nicht frei
(12) Prozesse fordern BM schrittweise an
(13) Prozesse fordern alle benötigten BM auf einmal an
(14) Die maximalen BM-Forderungen sind bekannt
(15) Die maximalen BM-Forderungen sind nicht bekannt
(16) Die Auslastung des Systems ist hoch
(17) Die Auslastung des Systems ist gering

Aufgabe 3

Gegeben seien zwei Betriebsmittel x und y mit zugehörigen Koordinationsvariablen (Semaphoren) sx und sy sowie zwei Prozesse p1 und p2 mit:

```
p1:   begin                          p2:   begin
      SVC belegen (sx);                    SVC belegen (sy);
      SVC belegen (sy);                    SVC belegen (sx);
      <benutze x und y>;                   <benutze x und y>;
      SVC freigeben (sy);                  SVC freigeben (sx);
      SVC freigeben (sx);                  SVC freigeben (sy);
      end                                  end
```

Zur Darstellung des Fortschrittes der beiden Prozesse diene das folgende Prozess-Zeit-Diagramm:

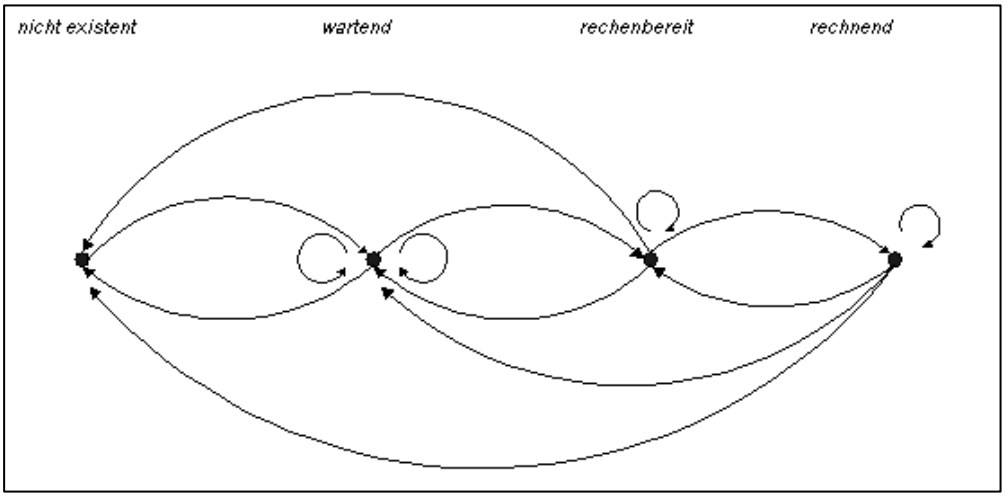


(Dabei gilt: b = belegen, f = freigeben)

1. Schraffieren Sie (///) die nicht betretbaren Gebiete.
2. Schraffieren Sie (\\\ ) die verklemmungsbedrohten Gebiete.
3. Zeichnen Sie (---) einen Verlauf ein, der zur Verklemmung führt, und markieren Sie den Punkt der Verklemmung mit V.
4. Zeichnen Sie (──) einen Verlauf ein, der die Verklemmungsmöglichkeit umgeht und beider Prozesse zu Ende führt.
5. Ändern Sie das angegebene Programm p2 so ab, dass Verklemmung ausgeschlossen ist.
6. Begründen Sie Ihre Lösung zu 3. mit Hilfe des zugehörigen Prozess-Zeit-Diagramms, in das Sie Einträge gemäß 1. und 2. machen.

Aufgabe 4

1. Das folgende Diagramm zeigt die Arbeitszustände eines Prozesses nebst Übergängen:



   Beschriften Sie die Übergänge anhand folgender Liste von Diensten (p = Prozess, e = Ereignis).
   (1) erzeugen(p)
   (2) löschen(p)
   (3) anhalten_bis_Ereignis(p,e)
   (4) melden_Ereignis(p,c)
   (5) Prozessorzuteilung(p)
   (6) Prozessorentzug(p)

2. Nennen Sie – über den Arbeitszustand hinaus – 5 weitere Bestandteile der Zustandsinformation (Status) eines Prozesses.

Aufgabe 5

Welche programmiersprachlichen Konstrukte zur Beschreibung von Synchronisationsproblemen kennen Sie für die Systemprogrammierung?
Beschreiben Sie diese, beginnend mit den hardwarenahen Sprachmitteln.

Aufgabe 6

1. Wodurch sind Transaktionen charakterisiert?
2. Mit welchen Techniken können Transaktionen realisiert werden?

Aufgabe 7

1. Skizzieren Sie die wesentlichen Charakteristiken der Rechnerarchitektur, die als *von-Neumann*-Rechner bekannt ist.
2. Was versteht man unter dem von Neumann'schen Flaschenhals?
3. Unter Speicherbesuchszahl versteht man die Anzahl der Arbeitsspeicherzugriffe pro Maschinenbefehl.
   Wovon ist die Speicherbesuchszahl abhängig?
   Was ist ein typischer mittlerer Wert?
   Gibt es eine obere Schranke?

4. Welche architekturellen Erweiterungen der von Neumann-Architektur kennen Sie, die den Flaschenhalseffekt abmildern?

Aufgabe 8

1. Für welches konzeptuelle Datenbank-Schema sind Normalformen von Bedeutung? Wozu dienen sie?
2. Welche Normalformen kennen Sie?
   Wie überführt man eine Normalform in eine andere?
3. Aus welchen wesentlichen Komponenten sind im Allgemeinen Expertensysteme aufgebaut?

Aufgabe 9

1. Nennen Sie drei wichtige Fehlertypen, die bei der Datenübertragung in Rechnernetzen vorkommen und geben Sie Fehlerursachen dafür an.
2. Nennen Sie für die Fehlertypen übliche Erkennungsmethoden und Auswege aus der Fehlersituation (recovery).
3. *Ethernet* und *Token Ring* sind wichtige Beispiele von lokalen Netzen. Kennzeichnen Sie kurz die zugehörigen Medien-Zugriffsverfahren. Wie beurteilen Sie die Verfahren unter Last?

2.3 Herbst 1990

Aufgabe 1

1. Beschreiben Sie die Verfahren der Segmentadressierung und der Seitenadressierung! Diskutieren Sie die durch diese Verfahren entstehenden Vorteile!
2. Skizzieren Sie die Organisation des Adressraums bei der kombinierten Segment-Seiten-Adressierung! Beschreiben Sie, wie der Zugriff auf eine Adresse erfolgt!
3. Beschreiben Sie das Working Set-Modell!
4. Beschreiben Sie 4 Ihnen bekannte Seiten-Verdrängungsstrategien und stellen Sie diese bewertend gegenüber!

Aufgabe 2

1.1 Datenorganisation ist auch mit algorithmischen Sprachen (Programmiersprachen) möglich. Nennen Sie Gründe, die für die Verwendung von Datenbanken sprechen!
1.2 Erklären Sie den Begriff der Integritätsbedingung an Hand eines selbstgewählten Beispieles!
1.3 Warum ist Datenredundanz ein Problem in Datenbanken?

2.1 Mit welchen drei grundsätzlichen Operationen (Relationsoperationen) arbeiten Abfragesprachen von relationalen Datenbanken, um Informationen aus der Datenbank zu erhalten? Welche Funktionalität haben diese jeweils?
2.2 Betrachten Sie die Komplexitäten dieser Relationsoperationen! Berücksichtigen Sie auch eventuelle Fallunterscheidungen!

3.1 Was ist der Sinn von Normalformen?

3.2 Welche Normalformen kennen Sie? Definieren Sie diese! Geben Sie jeweils ein Beispiel an, das die jeweilige Normalform erfüllt, höhere Normalformen aber nicht!

4. Gegeben sind folgende Relationen:

Teile	Tname x Tfarbe x Tgewicht x T#
Lieferung	L# x T# x Anzahl
Lieferant	L# x Lname x Status x Stadt

Gesucht ist eine SQL-Abfrage, die folgendes erfüllt:
„Lieferanten mit Status, die Welle mit Teilenummer 5 liefern“.

Aufgabe 3

1. Geben Sie die allgemeine Architektur eines Expertensystems an und beschreiben Sie die Aufgaben der einzelnen Komponenten!

2.1 Erläutern Sie die beiden grundlegenden Problemlösungsverfahren Breitensuche und Tiefensuche! Stellen Sie Vor- und Nachteile des jeweiligen Verfahrens gegenüber!

2.2 Wählen Sie ein geeignetes Beispiel und erklären Sie daran die beiden Verfahren!

3. Erklären Sie den Mechanismus der Unifikation in Prolog allgemein und anhand eines selbstgewählten Beispiels!

Aufgabe 4

1. Erläutern Sie kurz die Aufgaben der Schicht 3 des OSI-Referenzmodells!
2. Beschreiben Sie kurz die verschiedenen Schicht 3 Protokolle!
3. Nennen Sie die gängigsten Vermittlungsverfahren und erläutern Sie deren Wirkungsweise!
4. Geben Sie Beispiele für Netze an, die mit den genannten Vermittlungsverfahren arbeiten!
5. Diskutieren Sie Vor- und Nachteile der Verfahren und leiten Sie daraus sinnvolle Anwendungen für sie ab!
6. Erklären Sie die Begriffe „virtuelle Verbindung“ und „Datagramm“!

Aufgabe 5

1.1 Beschreiben Sie die wesentlichen Prinzipien der von-Neumann-Maschine (architekturell, operationell)!

1.2 Geben Sie einen einfachen Leitwerkzyklus für die von-Neumann-Maschine an!

2. Was versteht man unter „Mikroprogrammierung“?
   Erläutern Sie den Unterschied zwischen horizontaler und vertikaler Mikroprogrammierung! Welche Alternative zur Mikroprogrammierung gibt es?
3. Diskutieren Sie zwei Erweiterungen der von-Neumann-Prinzipien, die zu einer Leistungssteigerung beitragen!

2.4 Herbst 1991

Aufgabe 1 Virtueller Speicher

In einem ‚Demand-Paging’-System sei die folgende Seitenreferenzfolge (von links nach rechts gelesen) zu bearbeiten:

0 1 2 3 0 1 4 0 1 2 3 4

1. Ermitteln Sie, um wieviel Prozent die Zahl der Einlagerung bei Verwendung der Ersetzungsstrategie LRU (least recently used) höher liegt als bei einer optimalen Ersetzungsstrategie, und zwar bei Verwendung von
   a) drei Kacheln,
   b) vier Kacheln.
2. Führen Sie die gleiche Untersuchung für die Strategie ‚Second Chance' durch!

Aufgabe 2 Prozessorvergabestrategien

Gegeben sei ein System mit einer Bedienstation, die eine Reihe unabhängiger Aufträge zu bearbeiten hat, deren Bedienzeiten bekannt sind.

1. Zeigen Sie, dass die Abarbeitung nach steigender Bedienzeitanforderung (shortest job first) die mittlere Verweilzeit minimiert, wenn sämtliche zu bearbeitenden Aufträge bereits beim Start des Systems vorhanden sind!
2. Wie müsste die Zuordnungsstrategie geändert werden, damit (eventuell unter einschränkenden Annahmen) die mittlere Verweilzeit auch dann minimiert wird, wenn nicht alle Aufträge bereits beim Systemstart vorhanden sind, sondern teilweise erst danach in das System gelangen? Falls Sie einschränkende Annahmen machen, sind sie kurz zu diskutieren.

Aufgabe 3 Betriebsmittelverwaltung

Ein Betriebssystem bestehe aus n Prozessen (nummeriert von 0 bis n-1) und m wiederverwendbaren Betriebsmitteltypen (nummeriert von 0 bis m-1). Zur Darstellung der Betriebsmittelgraphen sollen folgende Datenstrukturen verwendet werden:

- ein Vektor v mit m Komponenten, der im jeweiligen Zustand angibt, wieviele Exemplare jedes Betriebsmitteltyps noch frei verfügbar sind,
- eine (n,m)-Matrix, die im jeweiligen Zustand angibt, wieviele Exemplare eines jeden Betriebsmittels den einzelnen Prozessen zugeordnet sind,
- eine (n,m)-Matrix, die im jeweiligen Zustand angibt, wieviele Exemplare eines jeden Betriebsmittels von den einzelnen Prozessen angefordert, aber noch nicht zugeteilt sind, und
- einen Vektor ps von n Semaphoren, dessen i-te Komponente als privater Semaphor dem i-ten Prozess (zu seiner Blockierung bzw. Deblockierung) zugeordnet ist.

Das Betriebssystem stelle folgende Prozeduren zur Verfügung:

- P- und V-Operationen für Semaphore und
- Die Funktion ‚this_process', die als Ergebnis die Nummer des aufrufenden Prozesses liefert.

Formulieren Sie unter diesen Voraussetzungen in einer geeigneten Programmiersprache folgende von den Prozessen gemeinsam benutzbaren Prozeduren (Beachten Sie, dass die obigen Datenstrukturen von den Prozessen gemeinsam benutzt werden!):

1. Eine Prozedur ‚anfordern', die von den Prozessen aufgerufen werden kann und als Parameter den Vektor der angeforderten Betriebsmittel besitzt. Falls die Anforderungen insgesamt erfüllbar sind, soll die entsprechende Zuteilung vorgenommen und zum aufrufenden Programm zurückgekehrt werden. Andernfalls sind die Anforderungen vorzumerken und der aufrufende Prozess zu blockieren.
2. Eine Prozedur ‚freigeben', die von den Prozessen aufgerufen werden kann und als Parameter den Vektor der freizugebenden Betriebsmittel besitzt. Ist für einen Betriebsmitteltyp die Angabe der freizugebenden Exemplare größer als die Zahl der an den aufrufenden Prozess zugeteilten, so werden lediglich die zugeteilten Exemplare

freigegeben. Sind nach der Freigabe blockierte Prozesse vorhanden, deren Anforderungen nun abgedeckt werden können, so sollen (nach einer beliebigen Strategie) so lange Prozesse deblockiert werden, bis weitere Deblockierungen nicht mehr möglich sind.
3. Eine Prozedur ‚prüfen', die feststellt, ob partielle Veklemmungen vorliegen (Berücksichtigen Sie, daß diese Prozedur die Daten, auf denen die Prozeduren ‚freigeben' und ‚anfordern' operieren, nicht modifizieren darf!).
4. Geben Sie die Anfangsbesetzung aller benutzten Semaphore an!

Aufgabe 4

1. Welche Eigenschaften muss eine Relation besitzen, damit sie die erste bzw. zweite bzw. dritte Normalform erfüllt?
2. Das Bundesamt für Kfz-Wesen habe in einem ersten Entwurf folgende unnormalisierte Relationen aufgestellt, um Verkehrssünder zu registrieren.
   EIGENTÜMER (PKZ, NAME, ADR, AUTO)
   AUTO (KFZNR, TYP, FARBE, EINTRAG)
   EINTRAG (LFDNR, DATUM, ART, STRAFPUNKTE, FAHRER)
   FAHRER (PKZ, NAME, ADR, FÜHRERSCHEIN, FÜHRERSCHEINENTZUG)
   FÜHRERSCHEIN (FNR, ADATUM, KLASSE, AUSSTELLUNGSORT)
   Die Primärschlüssel sind durch Unterstreichung kenntlich gemacht.
   Man normalisiere diese Relationen, und überführe sie in die dritte Normalform, wobei folgende Semantik zu unterstellen ist:
   - Jedes Auto kannnur einen Eigentümer, ein Eigentümer aber mehrere Autos haben.
   - Ein Auto und ein Fahrer sind jeweils durch einen Eintrag betroffen. Pro Auto und pro Fahrer können mehrere verschiedene Einträge existieren, d.h. ein Fahrer kann sich Einträge mit verschiedenen Autos einhandeln. Die LFDNR wird in Bezug auf Auto geführt.
   - Pro Fahrer sind verschiedene Führerscheine (auch Ersatzführerscheine) registriert.

2.5 Frühjahr 1993

Aufgabe 1 Virtuelle Speicherverwaltung

1. Als Verfahren zur Zuordnung von vituellen Adressräumen zum realen Speicher sind Ihnen Segmentierung und Paging bekannt. Was haben diese Verfahren gemeinsam, und wo liegen die Unterschiede?
2. Bei einem System mit Paging habe eine Seite die Länge k (Wörter). Die durchschnittliche Segmentlänge sei L. Für den Eintrag in die Seiten-Kachel-Tabelle sei jeweils ein Wort pro Seite erforderlich. Wie ist k zu wählen, um bei einer durchschnittlichen Segmentlänge L den durschnittlichen Speicherverschnitt pro Segment zu minimieren?
3. Wie kommt es beim Paging zu Seitenflattern, und was kann man dagegen tun?

Aufgabe 2 Synchronisationssysteme

Versetzen wir uns für diese Aufgabe in den Alltag einer Familie mit drei kleinen Kindern, die auch im Winter draußen spielen wollen. Damit sie sich nicht erkälten, braucht jedes Kind eine Mütze, ein Paar Handschuhe und einen Schal. Leider ist ihre Oma mit dem Stricken noch nicht so weit, so dass jedes Kind im Moment erst eines dieser drei Kleidungsstücke besitzt und zwar jedes ein anderes. Außerdem haben sie im Kleiderschrank noch eine alte Mütze, ein altes Paar Handschuhe und einen großen Schal gefunden.

Wenn nun ein Kind nach draußen will, braucht es noch zwei von den Sachen aus dem Schrank:

- entweder Schal und Mütze
- oder Mütze und Handschuhe
- oder Handschuhe und Schal

1. Geben Sie eine lösung dieses Problems an unter Verwendung von
   - P/V-Semaphoren
   - $P/V_{multiple}$-Semaphoren
   - Einem Petrinetz

2. Diskutieren Sie, ob sich das von Ihnen angegebene Prozesssystem verklemmen kann! (Welches allgemeine Verfahren verwenden Sie dabei zur Verhinderung von Verklemmungen?)

Aufgabe 3 Banker-Algorithmus

Gegeben sei ein System aus drei Prozessen P_1, P_2, P_3 und Betriebsmitteln B_1, B_2, B_3, B_4. Die Anzahl der vorhandenen Betriebsmittel ist festgelegt durch den Vektor:

$V = (6, 7, 6, 4)$

Die maximal benötigten Betriebsmittel der drei Prozesse sind gegeben durch die Matrix:

$$B = \begin{pmatrix} 1 & 5 & 3 & 2 \\ 2 & 3 & 3 & 1 \\ 4 & 5 & 2 & 4 \end{pmatrix}$$

Die folgenden Matrizen ergeben jeweils einen Status der zugeteilten Betriebsmittel:

$$Q_1 = \begin{pmatrix} 0 & 1 & 3 & 0 \\ 1 & 3 & 1 & 0 \\ 3 & 2 & 0 & 1 \end{pmatrix}$$

$$Q_2 = \begin{pmatrix} 1 & 4 & 1 & 0 \\ 1 & 3 & 3 & 1 \\ 4 & 0 & 2 & 4 \end{pmatrix}$$

$$Q_3 = \begin{pmatrix} 1 & 0 & 1 & 2 \\ 2 & 3 & 1 & 1 \\ 1 & 3 & 1 & 1 \end{pmatrix}$$

1. Sind die Zustände Q_1, Q_2 und Q_3 realisierbar?
2. Existieren für Q_1, Q_2 und Q_3 sichere Sequenzen?
3. Es sei nun $V = (6, 7, 6, 5)$, d.h. von Betriebsmittel B_4 steht ein Exemplar mehr zur Verfügung. Wie ändern sich dadurch die Antworten auf die Teilaufgaben 1. und 2.?

Aufgabe 4

1. Transaktionen

Welche vier grundlegenden Eigenschaften kennzeichnen eine Transaktion und was bedeuten diese Eigenschaften?

2. SQL

Gegeben seien die folgenden Relationen:

PERSONAL (PNR, NAME, GEHALT, BERUF, ANR)
ABTEILUNG (ANR, NAME, ORT)

Mit PNR Personalnummer und ANR Abteilungsnummer.

Folgende Operationen sind in SQL zu formulieren:

a) Gib mir die Namen der Angestellten, die in der Abteilung „Software“ arbeiten!
b) Bestimme das Durchschnittsgehalt der Angestellten aus Abteilung „4711“!
c) Gib mir das höchste Gehalt aus jeder Berufsgruppe!
d) Referentielle Integritätsbedingungen: Jeder Angestellte muss in einer Abteilung arbeiten! (keine genaue Syntax)

3. Codasyl

a) Geben Sie für obige beiden Relationen das Bachmann-Diagramm für den Codasyl-Set an!
b) Erstellen Sie ein Ausprägungsdiagramm für folgende Einträge: „In der Abteilung 4711 arbeiten die Angestellten Müller und Meier.“!

4. Normalformenlehre

Führen Sie nachfolgende Relation in die dritte Normalform über:
STUDENT (MATRNR NAME GEB ADR FBR FBNAME DEKAN)

9516570	HUBER	010148	XX	11	ING-WISS WALTER
9517058	MAIER	210849	YY	11	ING-WISS WALTER
9110457	BAUER	130548	XX	3	MEDIZIN HILBERG

2.6 Frühjahr 1994

Aufgabe 1

Erläutern Sie den Begriff des Deadlock anhand eines Beispiels!

Aufgabe 2

Gegeben sei ein Speicherbereich mit 1000 freien Speicherplätzen, welcher bei Adresse 10.000 beginnt. Es werden freie Speicherbereiche wie folgt angefordert und zurückgegeben:

a) Zeitpunkt 1 : Anforderung 500 Plätze
b) Zeitpunkt 2 : Anforderung 300 Plätze
c) Zeitpunkt 3 : Anforderung 100 Plätze
d) Zeitpunkt 4 : Rückgabe von a)
e) Zeitpunkt 5 : Anforderung 70 Plätze
f) Zeitpunkt 6 : Anforderung 430 Plätze
g) Zeitpunkt 7 : Anforderung 100 Plätze

Wie sieht die Speicherbelegung zu jedem Zeitpunkt aus, falls die Speichervergabe nach der

1. First-Fit-Methode
2. Best-Fit-Methode

erfolgt? Sind alle Speicheranforderungen jeweils zu erfüllen? Es ist davon auszugehen, dass freie Speicherbereiche in einer Liste verkettet werden und ein zurückgegebener Block jeweils am Anfang dieser Liste eingekettet wird.

Aufgabe 3

Erläutern Sie den Begriff des von-Neumann-Rechners!

Aufgabe 4

Gegeben seien die folgenden Attribute für Bücher, Entleiher und Autoren:

Signatur	Signatur eines Buches
Titel	Buchtitel
Gebiet	Fachgebiet des Buches
Art	Art des Buches, z.B. Einführungsbuch o.ä.
ErschOrt	Erscheinungsort
Matr	Matrikelnummer eines Entleihers
StudName	Name des Entleihers
StudGeb	Geburtsdatum des Entleihers
Sem	Semester des Entleihers
StudWohnort	Wohnort des Entleihers
StudFachr	Fachrichtung des Entleihers
AutNr	Nummer eines Autors
AutName	Name eines Autors
AutWohnort	Wohnort eines Autors
AutBuchHonorar	Honorar, welches ein Autor für ein Buch bekommt

Studenten können mehrere Bücher entliehen haben. Ein Autor kann mehrere Bücher geschrieben haben. Ein Buch kann mehrere Autoren haben, in diesem Fall bekommt jeder Autor ein eigenes autor- und buchspezifisches Honorar.
Welche funktionalen Abhängigkeiten gelten unter den obigen Attributen?

Aufgabe 5

Man betrachte das relationale Schema

R (Signatur, Titel, Gebiet, Art, ErschOrt, Matr, StudName, StudGeb, Sem, StudWohnort, StudFachr, AutNr, AutName, AutWohnort, AutBuchHonorar)

und die funktionalen Abhängigkeiten aus Aufgabe 4.
Ist die Dekomposition

R1 (Signatur, StudName, StudGeb, Sem, StudWohnort, StudFachr)
R2 (Signatur, Titel, Gebiet, Art, ErschOrt)
R3 (AutNr, AutName, AutWohnort)
R4 (Signatur, AutNr, AutBuchHonorar)

verlustfrei, d.h. hat sie die lossless-join-Eigenschaft? Man begründe die Antwort, z.B. durch Verwendung eines geeigneten Algorithmus'.

Aufgabe 6

Man betrachte das relationale Schema

R (Signatur, Titel, Gebiet, Art, ErschOrt, Matr, StudName, StudGeb, Sem, StudWohnort, StudFachr, AutNr, AutName, AutWohnort, AutBuchHonorar)

und die funktionalen Abhängigkeiten aus Aufgabe 4.
Ist die Dekomposition

R1 (Signatur, StudName, StudGeb, Sem, StudWohnort, StudFachr)
R2 (Signatur, Titel, Gebiet, Art, ErschOrt)
R3 (AutNr, AutName, AutWohnort)
R4 (Signatur, AutBuchHonorar)

abhängigkeitserhaltend? Man begründe die Antwort, z.B. durch Verwendung eines geeigneten Algorithmus'.

Aufgabe 7

Man betrachte das relationale Schema

R (Signatur, Titel, Gebiet, Art, ErschOrt, Matr, StudName, StudGeb, Sem, StudWohnort, StudFachr, AutNr, AutName, AutWohnort, AutBuchHonorar)

und die funktionalen Abhängigkeiten aus Aufgabe 4.
Man gebe eine abhängigkeitserhaltende und verlustfreie (lossless join) Dekomposition von R in 3. Normalform an. Man verwende einen geeigneten Algorithmus.

Aufgabe 8

Man betrachte das relationale Schema

R (Signatur, Titel, Gebiet, Art, ErschOrt, Matr, StudName, StudGeb, Sem, StudWohnort, StudFachr, AutNr, AutName, AutWohnort, AutBuchHonorar)

und die funktionalen Abhängigkeiten aus Aufgabe 4.
Man gebe eine Dekomposition von R in Boyce-Codd-Normalform an. Man verwende einen geeigneten Algorithmus. Ist die Dekomposition abhängigkeitserhaltend und verlustfrei (lossless join)?

Aufgabe 9

Für die Attribute aus Aufgabe 4 gebe man SQL CREATE Anweisungen für ein sinnvolles Schema an, z.B. für die Dekomposition aus Aufgabe 7 oder aus Aufgabe 8.

Aufgabe 10

Unter Verwendung des Schemas aus Aufgabe 9 gebe man SQL-Anweisungen und Ausdrücke in der relationalen Algebra für die folgenden Anfragen an:

1. Gesucht ist der Name des Studenten mit der Matrikelnummer 123456.
2. Gesucht sind die Signaturen der Bücher, die der Student mit der Matrikelnummer 654321 ausgeliehen hat.
3. Gesucht sind die Signaturen der Bücher, die der Student namens Albrecht ausgeliehen hat.
4. Gesucht sind die Autornamen der Bücher, die der Student namens Albrecht ausgeliehen hat.

2.7 Herbst 1994

Aufgabe 1 Verklemmungen

1. Wie lauten die vier notwendigen Bedingungen für Verklemmungen?

2. Wie können Verklemmungen behoben werden (Kurze Erläuterung)?
3. Gegeben sei ein Prozesssystem mit drei Prozessen.
   Prozeß 1 erzeugt Betriebsmittel vom Typ 2.
   Die Betriebsmittel vom Typ 1 und 3 sind wiederverwendbar.
   Anfangs sind an Betriebsmitteln vorhanden:
   BM 1 : 5 BM 2 : 3 BM 3 : 3
   Die drei Prozesse haben folgende Operationen bereits ausgeführt:
   Anforderung P2: {(BM 1, 1)}
   Anforderung P3: {(BM 3, 3)}
   Zuteilung(P2);
   Zuteilung(P3);
   Anforderung P1: {(BM 1, 2), (BM 3, 1)}
   Anforderung P2: {(BM 3, 1)}
   Anforderung P3: {(BM 2, 4)}
   Stellen Sie den augenblicklichen Zustand als Betriebsmittelgraph dar!
4. Liegt eine Verklemmung vor? Kurze Begründung!

Aufgabe 2 Arbeitsspeicherverwaltung

1. Beschreiben Sie den Seitenaustauschalgorithmus SC (second chance)!
2. Was enthält ein Segmentdeskriptor bei einem System, das nur Segmentierung und keine Seitenadressierung kennt und keinen Cache hat?
3. Was versteht man unter Seitenflattern (Thrashing) und was sind die Folgen?
4. Welche zusätzlichen Prozesszustände sind notwendig, um das Problem Thrashing zu lösen? Zeichnen Sie das erweiterte Prozess-Zustandsdiagramm!

Aufgabe 3 Periphere Geräte

Beschreiben Sie kurz die Funktionsweise sowie Vor- und Nachteile der folgenden Anschlussschemata für periphere Geräte!

1. Integrierte Ausführung
2. Abgesetzte Ausführung
3. Selbständige Ausführung

Aufgabe 4 Prozesssystem / Synchronisation

1. Implementieren Sie das Leser-Schreiber-Problem als Prozesssystem P, mit folgenden Eigenschaften:
   Es gibt eine beschränkte Zahl von k Lesern, die alle gleichzeitig lesen dürfen. Es gibt eine beliebige Zahl von Schreibern. Ein Schreiber braucht exklusiven Zugriff zu den Daten. Sie können frei wählen, welchers Synchronisationssystem Sie verwenden wollen, ebenso, ob die Schreiber Vorrang haben sollen oder nicht (bitte aber angeben). Ihre Lösung darf sich nicht verklemmen und darf kein aktives Warten (busy wait) aufweisen.
2. Beschreiben Sie das von Ihnen verwendete Synchronisationssystem!

2.8 Frühjahr 1995

Aufgabe 1

1. Drücken Sie durch eine Formel aus, wieviel Zeit benötigt wird, um ein Programm auf einem Rechner in Abhängigkeit der folgenden Größen abzuarbeiten:
   - l mittlere Zykluslänge
   - CPI Anzahl der Zyklen pro Maschineninstruktion (Cycles Per Instruction)
   - IC mittlere Befehlszahl (Instruction Count)
2. Welche Möglichkeiten gibt es für den Rechnerarchitekten einen Rechner zu entwickeln, der das Programm möglichst schnell abarbeitet? Welche dieser Möglichkeiten sind eher technologie-, welche eher architekturabhängig? Welche dieser Möglichkeiten lassen sich nur schlecht gleichzeitig verwirklichen?
3. Für einen Mix aus Benchmarks wurden folgende Anteile verschiedener Instruktionsarten ermittelt:

Instruktion	Anteil	Rechner 1	Rechner 2
Integer-Operation	51%	2	1
Load / Store	30%	4	1
Branch	17%	2	1
Jump	2%	2	1

Die beiden letzten Spalten geben die Anzahl der Zyklen pro Instruktion für Rechner 1 bzw. Rechner 2 an. Die Taktfrequenz von Rechner 1 beträgt 20MHz und die Taktfrequenz von Rechner 2 beträgt 8MHz. Welcher Rechner ist schneller?

Aufgabe 2

Betrachten Sie eine typische 6-stufige Befehlspipeline!

1. Welche Funktion haben die einzelnen Pipelinestufen?
2. Wie groß ist die Beschleunigung (Speed-up), die man erhält, wenn der Instruktionsstrom aus n Maschineninstruktionen besteht und keine Pipeline-Hemmnisse auftreten?
3. Welche Beschleunigung erhält man für einen Instruktionsstrom der Länge n im Idealfall, wenn die Prozessorarchitektur m derartige Pipelines enthält, die nebenläufig betrieben werden können (Superskalarprozessor)?
4. Nennen Sie (wenigstens) zwei Eigenschaften von Instruktionsströmen, die verhindern, dass dieser Idealfall eintreten kann! Erläutern Sie die Auswirkung dieser Eigenschaften anhand von Pipeline-Phasendiagrammen!
5. Skizzieren Sie eine 4-stufige (arithmetische) Pipeline für die Gleitpunkt-Addition, und erläutern Sie die Funktion der einzelnen Stufen!

Aufgabe 3

1. Erläutern Sie die Begriffe CISC und RISC! Stellen Sie dazu jeweils charakteristische Eigenschaften gegenüber!
2. Um die Befehlspipeline in einem Prozessor nicht zu bremsen, arbeiten manche RISC Prozessoren mit sogenannten *delayed branches*, d.h. der hinter einem Sprungbefehl stehende Befehl (im sogenannten *delay slot*) wird noch in jedem Fall ausgeführt.
   Geben Sie zwei Möglichkeiten zur Füllung des *delay slots* an und begründen Sie, wann sie verwendet werden können und wann dadurch eine Leistungssteigerung erreicht wird!
3. Nennen Sie neben Sprüngen eine weitere Ursache, die zu Verzögerungen in einer Befehlspipeline führen kann und geben Sie eine Möglichkeit zur Behebung an!
4. Welche Auswirkung haben RISC Prozessoren auf Compiler?

2.9 Frühjahr 1996

Aufgabe 1

Eine Prioritätswarteschlange kann als Heap implementiert werden.

1. Geben Sie die Definition eines Heaps an, und vereinbaren Sie eine entsprechende Datenstuktur!
2. Beschreiben Sie die Operationen
   - access (Zugriff auf das Maximum)
   - remove (Entfernen des Maximums)
   - insert (Einfügen eines Elements)
   - construct (Aufbau eines Heaps aus n Werten)
3. Geben Sie die Komplexität der angegebenen Operationen an, und begründen Sie Ihre Angaben!

Aufgabe 2

Eine Universitätsdatenbank soll folgende Daten verwalten.

- Studenten (Name, Matrikelnummer, Geburtsdatum, Adressen, Semesteranzahl, Studiengang, Fakultät, belegte Lehrveranstaltungen und deren Art)
- Lehrveranstaltungen (Anfangszeit, Semester, Art (Vorlesung, Seminar, Übung, Praktikum), Name, Anzahl der Hörer, Nummer im Vorlesungsverzeichnis, Anzahl Semesterwochenstunden, Dozent, Raum)

1. Entwerfen Sie ein ER-Schema für diese Applikation! Berücksichtigen Sie dabei, dass eine Vorlesung in 2 Unterrichtseinheiten aufgeteilt werden kann (z.B. Mo 10.00 und Do 14.00). Begründen Sie Ihren Entwurf!
2. Spezifizieren Sie für die Entity-Typen Attribute, und zeichnen Sie die Schlüsselattribute aus!
3. Geben Sie ein relationales Datenbankschema an!

Aufgabe 3

1. Beschreiben Sie den Aufbau des UNIX-Filesystems!
2. Welche Aufgabe kommt den Inodes zu?
3. Was ist ein Link?
4. Welche Tabellen werden durchlaufen, wenn durch einen Dateideskriptor auf ein File zugegriffen wird?

Aufgabe 4

Beschreiben Sie die Auswirkungen, die sich durch Anwenden des Pipeline-Prinzips

1. bei der Befehlsentschlüsselung
2. bei der Arithmetik von Vektorrechnern ergeben!

Aufgabe 5

Charakterisieren Sie

1. eine Batch Transformation
2. eine ereignisgesteuerte Benutzeroberfläche
3. eine Client-Server-Lösung!

Aufgabe 6

Folgende Tabelle gibt die IC-Fahrzeiten an:

Abfahrt	Ankunft	Zeit
KA	BAD	18 min
KA	MA	25 min
MA	MZ	46 min
MA	F	48 min
MZ	KO	49 min
MZ	F	51 min
F	MZ	51 min
F	KS	1:58 h
F	FD	1:05 h
KO	DO	1:36 h
FD	G	2:19 h
DO	H	1:32 h
DO	HB	1:47 h
H	HB	1:12 h
H	HH	1:14 h
HB	HH	53 min

Bestimmen Sie einen Weg von Karlsruhe nach Hamburg
1. mit Tiefensuche
2. mit Breitensuche
3. mit Hill-Climbing-Suche
4. mit Branch-and-Bound-Suche.

Vergleichen Sie die Fahrzeiten!

2.10 Herbst 1996

Aufgabe 1

1. Rechensysteme werden nach bestimmten Kriterien klassifiziert, um sie qualitativ und quantitativ analysieren und miteinander vergleichen zu können. Die Klassifikation kann dabei nach Leistung, Anwendungsanforderungen oder Anwendungsgebieten erfolgen. Nennen Sie zwei weit verbreitete Anwendungsgebiete und erläutern Sie deren Besonderheiten im Hinblick auf die Architektur von Rechensystemen.
2. Eine häufig verwendete Taxonomie ist die sogenannte *Flynn'sche Klassifikation*. Erklären Sie diese Taxonomie und nennen Sie die Merkmale der beiden Unterklassen der *Princeton-* und *Harvard-Architektur*. Welche Arten der Parallelität gibt es bei Parallelrechnern?
3. Der Datenaustausch in MIMD-Rechnern kann auf zwei Arten erfolgen. Beschreiben Sie die beiden Methoden!

Aufgabe 2

1. Ein Computer besitzt eine drei-stufige Speicherhierarchie aus Cache, Hauptspeicher und Plattenspeicher. Ein Cache-zugriff erfordert 1 Taktzyklus, ein Zugriff auf den Hauptspeicher 4 Taktzyklen und ein Zugriff auf den Plattenspeicher 1200 Taktzyklen: 0.5% der Zugriffe erfolgen auf den Plattenspeicher, 30% auf den Hauptspeicher und die restlichen 69.5% auf den Cache.
   Wieviele Zyklen werden im Mittel für einen Speicherzugriff benötigt? Wieviele Zugriffe benötigt derselbe Computer im Mittel, wenn er keinen Cache besitzt und stattdessen diese Zugriffe vom Hauptspeicher erledigt werden können?
   Wie wirkt sich die Einführung eines Second-Level-Caches aus, der eine Zugriffszeit von 2 Taktzyklen besitzt und auf den 60% der Speicherzugriffe erfolgen, die im ursprünglichen System vom Hauptspeicher erledigt wurden: die Wahrscheinlichkeiten für Zugriffe auf den Plattenspeicher und den First-Level-Cache bleiben unverändert.
2. Ein Computer besitzt eine Taktfrequenz von 120 MHz und benötigt für eine Integer-Instruktion im Mittel 6 Takte. Wieviele µs werden im Mittel für eine Instruktion benötigt und wie viele MIPS (million instructons per second) werden im Durchschnitt ausgeführt? Derselbe Rechner benötigt für einen Befehl mit Fließkommaverarbeitung durchschnittlich 18 Takte. Wieviele µs werden im Mittel für eine Fließkommaverarbeitung benötigt und wieviele FLOPS (floating point operations per second) werden im Mittel geleistet?

Aufgabe 3

Ein SISD-Rechner berechnet ein Problem, bestehend aus 9 gleichartigen Teilproblemen und benötigt für jeden Teil 3h. Der Anteil der parallelisierbaren Operationen beträgt 2/3 . Wie lange benötigt ein MIMD-Rechner mit 10 Prozessoren für das Gesamtproblem, falls die Kommunikation vernachläsigbar ist und jeder Prozessor die gleiche Rechenleistung wie der SISD-Rechner aufweist? Wie groß ist der Speed-up und die Effizienz? Wie groß sind die Durchsätze dieser Rechner?

Aufgabe 4

Gegeben sei die folgende, in dritter Normalform vorliegende relationale Datenbank zur Modellierung des Gebrauchtwagenparks eines Autohändlers.

Modelle	mnr	hnr	typ	neupreis	ps
	1	1	Corsa	18.000	60
	2	1	Vectra	30.000	90
	3	1	Omega	40.000	110
	4	1	Astra	28.000	90
	5	2	Golf	30.000	90

Hersteller	hnr	hersteller
	1	Opel
	2	VW

Fahrzeuge	mnr	fgnr	baujahr	Preis
	1	H674	1992	13.000
	1	C634	1990	12.000
	2	D459	1992	22.000
	3	C634	1989	22.000

	3	H789	1993	24.000

Die Relation *Modelle* beinhaltet alle Fahrzeugtypen, die der Händler im Gebrauchtwagenprogramm führt. Die Modelle sind über das Attribut ‚mnr' über alle Hersteller hinweg eindeutig numeriert. ‚mnr' ist daher Primärschlüssel in der Relation *Modelle.* Über das Attribut ‚hnr' wird von *Modelle* auf die Relation *Hersteller* verwiesen. In der Relation *Fahrzeuge* werden alle tatsächlich beim Händler am Lager befindlichen Fahrzeuge geführt. Über ‚mnr' wird von Fahrzeuge auf Modelle verwiesen. Bei gegebener Modellnummer ist die vergebene Fahrgestellnummer (‚fgnr') eindeutig. Darum bilden ‚mnr' und ‚fgnr' zusammen den Primärschlüssel der Relation *Fahrzeuge*.

Formulieren Sie folgende Anfragen in jeweils drei der vier Anfragesprachen: relationale Algebra, relationaler Tupelkalkül, SQL und Quel.

1. Bestimmen Sie alle Modelle mit mehr als 60 PS.
2. Bestimmen Sie die Typen aller Modelle des Herstellers VW.
3. Bestimmen Sie die Nummern aller Modelle des Herstellers Opel, von denen tatsächlich Fahrzeuge auf Lager sind.

Formulieren Sie folgende Anfragen nur in SQL.

4. Bestimmen Sie die Namen der Hersteller, für deren sämtliche Modelle mindestens ein Fahrzeug im aktuellen Bestand vorhanden ist.
5. Bestimmen Sie den Durchschnittspreis (Attribut ‚preis') der am Lager vorhandenen Opel-Fahrzeuge.
6. Bestimmen Sie die jeweils durchschnittlichen Neupreise (Attribut ‚neupreis') aller geführten Modelle der verschiedenen Hersteller.

2.11 Frühjahr 1997

Aufgabe 1 Maschinenprogrammierung

1. Nennen und erläutern Sie vier verschiedene Adressierungsmodi der maschinennahen Programmierung.
2. Erklären Sie anhand der folgenden Speicherbelegung die Adressierungsmodi ‚Register-direkt', ‚Register-indirekt' und ‚Register-Speicher-indirekt'.

Speicheradresse		...	18	...	117	...	269	...	293	...	456	...
Speicherinhalt			293		269		456		117		18	

   Es soll der Befehl

   ```
   LOAD to Register1, Register2;
   ```

   mit verschiedenen Adressierungsmodi ausgeführt werden. Der Inhalt von Register2 dient dazu, den Wert zu bestimmen, der in Register1 geladen wird. In Register2 befindet sich der Wert 293. Welcher Wert steht – abhängig vom jeweiligen Adressierungsmodus – nach Ausführung des Befehls in Register1?
3. Beschreiben Sie die Schritte bei der Ausführung eines Unterprogrammaufrufs und –rücksprungs. Es soll das Stapelprinzip verwendet werden, wobei der Stapel zu kleineren Adressen wächst, der Stackpointer auf das nächste freie Element zeigt und das Abspeichern eines Elements auf dem Stack den Stackpointer um 4 verändert. Nennen Sie wenigstens zwei wesentliche Vorteile dieser Unterprogrammverwaltung.

Aufgabe 2 Speichersystem

1. Was versteht man unter Speicherzugriffszeit und unter Speicherzykluszeit? Wann sind beide gleich?
2. Was versteht man unter der räumlichen und der zeitlichen Referenzlokalität? Erläutern Sie, wie diese zustande kommen! Geben Sie wenigstens drei Beispiele an, die zeigen, wie diese Eigenschaften von der Rechnerarchitektur ausgenützt werden können!
3. Was versteht man unter einer Speicherhierarchie? Geben Sie ein Beispiel an! Wie lautet die Formel für die mittlere Speicherzugriffszeit bei einer zweistufigen Hierarchie?
4. Welche Aufgaben hat die MMU (Memory Manage Unit)? Welches sind ihre wichtigsten Hardwarekomponenten?

Aufgabe 3 Ein- / Ausgabe

1. Was versteht man unter einer speicherabgebildeten und unter einer separaten Ein/Ausgabe? Geben Sie jeweils mindestens zwei charakterisierende Eigenschaften an! Nennen Sie für jede der beiden Organisationsformen wenigstens einen Vorteil!
2. Was versteht man unter ‚programmierter Ein/Ausgabe'? Welche Alternativen dazu gibt es? Erläutern Sie den Unterschied zwischen ‚Polling' und interruptgesteuerter Ein/Ausgabe!
3. Weshalb lassen sich Peripheriegeräte u.a. nicht direkt mit der CPU und dem Hauptspeicher verbinden? Nennen Sie mindestens vier Möglichkeiten, eine Verbindung herzustellen!

2.12 Herbst 1997

Aufgabe 1 Virtuelle Speicherverwaltung

1. Erklären Sie den Unterschied zwischen einer Maschinenadresse und einer virtuellen Adresse. Beschreiben Sie den Aufbau beider Adresstypen unter der Annahme von Segment-Seiten-Adressierung.
2. Erklären Sie die Schritte zur Abbildung der virtuellen Adresse auf die zugehörige Maschinenadresse; insbesondere beschreiben Sie dabei auch den Aufbau der dazu notwendigen Datenstrukturen.
3. Beschreiben Sie die Aktionen, die im Betriebssystem ablaufen, wenn der Prozess auf ein Datum zugreifen will, dessen Seite gerade aus dem Arbeitsspeicher ausgelagert ist (Hinweis: Seitefehlt-Alarm).
4. Betrachtung von Seitenersetzungsstrategien:

4.1 Beschreiben Sie die Seitenersetzungsstrategien FIFO, LRU und LFU.

4.2 Ein Programm umfasse 5 Seiten, während die Kapazität des Arbeitsspeichers 2 Seiten betrage. Der Arbeitsspeicher sei zu Anfang leer. Zeigen Sie anhand der folgenden Seitenzugriffsreihenfolge jeweils das Verhalten von FIFO, LRU und LFU.
Wieviele Seitefehlt-Alarme werden für jede der Ersetzungsstrategien ausgelöst?

4.3 Wieviele Seitefehlt-Alarme würde eine optimale Ersetzungsstrategie auslösen? Begründen Sie Ihre Antwort, indem Sie Ihre Strategie kurz beschreiben. Warum ist diese Strategie in der Praxis nicht anwendbar?

4.4 Das Working-Set-Modell kann als formale Basis für Seitenersetzungsstrategien dienen. Definieren Sie den Begriff Working-Set, und geben Sie für die obige Zugriffsreihenfolge (siehe 4.2) den Inhalt des Working-Sets jeweils nach dem 3., 6. und 9. Speicherzugriff an (unter der Annahme, daß $\Delta=3$ ist).

5. Welches Zugriffsverhalten haben die folgenden Programmkonstrukte, und wie wirkt sich dieses jeweils auf die Segment-Seitenadressierung aus?
   Goto, Iteration, Rekursion, Modularität

Aufgabe 2 Prozess-Synchronisation

1. Charakterisieren Sie den Begriff der Verklemmung an Hand eines Betriebsmittelzuteilungsgraphen!
   a) Welche Methoden kennen Sie zur Verklemmungsauflösung?
   b) Welche Methoden kennen Sie zur Verklemmungsvermeidung?
2. Implementieren Sie ein allgemeines, ganzzahliges Semaphor S mittels binärer Semaphore und einer integer-Variablen zusammen mit der P- und V-Operation für S (benutzen Sie dafür eine Pascal- oder C-ähnliche Programmiersprache).
3. Entwickeln Sie mittels Semaphore Lösungen für folgende Varianten des Leser-Schreiber-Problems! Welche der nachfolgenden Varianten können zu unendlichen Warten führen? Begründen Sie kurz Ihre Antwort!
   a) Jeweils nur ein Leser oder ein Schreiber ist zu einem Zeitpunkt erlaubt.
   b) Jeweils nur ein Schreiber oder viele Leser sind zu einem Zeitpunkt erlaubt.
   c) Jeweils nur ein Schreiber oder viele Leser sind zu einem Zeitpunkt erlaubt; Schreiber haben jeweils höhere Priorität.
   d) Jeweils nur ein Schreiber oder viele Leser sind zu einem Zeitpunkt erlaubt; Schreiber haben jeweils höhere Priorität, jedoch nach Beendigung eines Schreiberswerden die wartenden Leser berücksichtigt.
4. Erklären Sie, wie Semaphore und zugehörige Operationen im Betriebssystemkern realisiert werden können. Welche Voraussetzungen muss das Betriebssystem dabei erfüllen? Auf der Basis dieser Voraussetzungen skizzieren Sie einen Algorithmus zur Realisierung der P- und V-Operationen!

Aufgabe 3

Für ein Unternehmen soll eine Fertigungsdatenbank aufgebaut werden. Der Erhebungsprozess liefere folgenden Informationsbedarf:
Entity-Mengen:

- ABTEILUNG mit den Attributen ANR, ANAME, AORT, MNR
- PERSONAL mit den Attributen PNR, NAME, BERUF
- MASCHINEN mit den Attributen MANR, FABRIKAT, TYP, BEZ, LEISTUNG
- TEILE mit den Attributen TNR, BEZ, GEWICHT, FARBE, PREIS

Relationship-Mengen:

- ABT-PERS zwischen ABTEILUNG und PERSONAL
- SETZT-EIN zwischen ABTEILUNG und MASCHINEN
- KANN-BEDIENEN zwischen PERSONAL und MASCHINEN
- GEEIGNET-FÜR-DIE-HERSTELLUNG-VON zwischen MASCHINEN und TEILE
- PRODUKTION zwischen PERSONAL, TEILE und MASCHINEN mit den Attributen DATUM und MENGE

Dabei sollen folgende grundlegenden Integritätsbedingungen (in Form von erweiterten Kardinalitätsrestriktionen) gelten:

- Zu einer Abteilung gehört immer mindestens ein Beschäftigter.

- Eine Person ist immer nur genau einer Abteilung zugeordnet.
- Eine Maschine kann, wenn überhaupt, nur von einer Abteilung eingesetzt werden.
- Alle anderen (Teil-)Beziehungen sind nicht weiter eingeschränkt.

1. Zeichnen Sie zu dem obigen Szenario das zugehörige Entity-Relationship-Diagramm.
2. Legen Sie die Schlüsselkandidaten fest, und zeichnen Sie diese in das ER-Diagramm ein! Ergänzen Sie die o.g. Integritätsbedingungen um weitere erweiterte Kardinalitätsrestriktionen zur genauen Festlegung der Semantik der Miniwelt, und tragen Sie diese Informationen ebenfalls in das ER-Diagramm ein.
3. Führen Sie die gefundene Informationsstruktur über in Datenstrukturen nach dem Relationenmodell! Geben Sie im Relationenschema die Primär- und Fremdschlüssel an.

Aufgabe 4

Gegeben sei folgende Datenbank, die das Ausleihwesen einer Bibliothek unterstützt:
LESER: L (LSNR, NAME, VORNAME, WOHNORT, GEB.DAT)
BUCH: B (ISBN, TITEL, SEITENZAHL, VERLAG, ERSCHEINUNGSJAHR, ANZAHL, EXEMPLARE)
VERLAG: V (VERLAG, VERLAGSORT, ...)
EXEMPLAR: E (ISBN, EXPNR, INVENTARNR, STANDORT)
AUSLEIHE: A (LSNR, ISBN, EXPNR, DATUM)

Formulieren Sie folgende Anfragen mit SQL:

1. Welche Leser haben Bücher ausgeliehen, die an ihrem Wohnort verlegt wurden?
2. Von welchen Buchtiteln sind alle Exemplare ausgeliehen?
3. In die LESER-Relation werden drei neue Leser(innen) gespeichert, deren Daten wie folgt lauten:

LSNR	NAME	VORNAME	WOHNORT	GEB.DAT
4711	Müller	Hugo	KL	11121955
4712	Maier	Maria	PS	05091900
4713	Meyer	Heike	unbekannt	unbekannt

Aufgabe 5

Charakterisieren Sie das Verhalten des Tokenring gegenüber dem Ethernet bei hoher bzw. niedriger Last. Für welche Einsatzbereiche und Situationen würden Sie den Tokenring gegenüber dem Ethernet vorziehen? In welchen Fällen würden Sie das Ethernet vorziehen? Begründen Sie und diskutieren Sie jeweils Ihre Wahl!

2.13 Herbst 1998 I

Aufgabe 1

Gegeben sei folgendes Relationenschema
R1 = (MATNR, VORNAME, NACHNAME, PLZ, STADT)
R2 = (MATNR, VORLESUNG, ZEIT, NACHNAME)
R3 = (VORLESUNG, RAUM, ZEIT, PROFESSOR)
Und die folgende Menge F von funktionalen Abhängigkeiten
MATNR $\rightarrow$ VORNAME, NACHNAME
MATNR $\rightarrow$ PLZ

PLZ → STADT
VORLESUNG → PROFESSOR
VORLESUNG, ZEIT → RAUM
PROFESSOR, ZEIT → VORLESUNG

1. Nennen Sie jeweils alle möglichen Schlüssel von R1, R2 und R3.
2. Geben Sie für R1, R2, R3 die schärfste Normalform (1NF, 2NF, 3NF, BCNF) an, die für die jeweilige Relation gilt. Begründen Sie gegebenenfalls, warum die nächstschärfere nicht erfüllt ist.

Aufgabe 2

In einem Betrieb soll ein neues Informationssystem eingerichtet werden. Bei der Voruntersuchung zeigt sich, dass in der Datenbasis zumindest die zwei Bereiche PERS für Personal und ABT für Abteilung einzurichten sind, wobei die folgenden Informationen zu verwalten sind:

- Abteilung: ANR, ANAME, AORT
- Personal: PNR, NAME, GEHALT, BERUF, ANR, VNR, ORT

Dabei steht ANR für Abteilungsnummer, PNR für Personalnummer und VNR für die Personalnummer des jeweiligen Vorgesetzten. Alle anderen Bezeichnungen sind selbsterklärend.

Zusätzlich wird festgestellt, dass die folgenden grundlegenden Integritätsbedingungen zu erfüllen sind:

- Zu einer Abteilung gehört immer mindestens ein Angestellter.
- Ein Angestellter ist immer nur genau einer Abteilung zugeordnet.
- Zu einem Angestellten gibt es maximal einen vorgesetzten Manager.
- Ein Vorgesetzter kann mehrere Angestellte verantwortlich betreuen.

1. Modellieren Sie oben beschriebene Informationsstruktur mit Hilfe eines Entity-Relationship-Graphen. Tragen Sie darin auch die Integritätsbedingungen in geeigneter Weise ein.
2. Transformieren Sie Ihr ER-Modell in Datenstrukturen nach dem Relationenmodell. Geben Sie in Ihrem Relationenschema die Primär- und Fremdschlüssel an.
3. Formulieren Sie folgende Anfragen und Änderungsoperationen an Ihre Datenbank in der Sprache SQL:
    a. Welche Angestellten aus der Abteilung „Einkauf“ verdienen weniger als das durchschnittliche Gehalt aller Angestellten der Firma?
    b. Welche Abteilungen in Frankfurt beschäftigen mehr als 10 Angestellte mit dem Beruf „Programmierer“?
    c. In welchen Abteilungen (ANR, ANAME) verdienen die Angestellten im Durchschnitt weniger als 2500 DM?
    d. Welche Abteilungen haben keine Angestellten?
    e. Finde die Abteilungsnummern von Abteilungen in Darmstadt, in denen es Angestellte gibt, die weniger als 2000 DM verdienen.
    f. Finde die Namen der Angestellten, die den gleichen Beruf und das gleiche Gehalt wie der Angestellte „Müller“ haben.
    g. In der Abteilung „Marketing“ wird ein neuer Angestellter mit dem Namen „Meier“ eingestellt. Er soll die Personalnummer 353 erhalten.
    h. Lösche alle Abteilungen, die keine Angestellten haben.

Aufgabe 3 Maschinennahe Berechnung arithmetischer Ausdrücke

Ein Prozessor P1 mit den 32 Registern R0 bis R31 soll den arithmetischen Ausdruck

```
Z := (A*B+C)*(D-E*F)
```

berechnen. Die Inhalte der Variablen A bis F stehen in Speicherzellen mit den Adressen mA bis mF, das Ergebnis Z soll in der Speicherzelle mit der Adresse mZ abgelegt werden. P1 hat unter anderem die folgenden Befehle (r, r1, r2, r3 stehen für Register, m für eine Speicherzelle und <a> für den Inhalt des Speicherelements a):

Befehl	**Wirkung**
load r m	<r> := <m>
store r m	<m> := <r>
add r1 r2 r3	<r3> := <r1> + <r2>
sub r1 r2 r3	<r3> := <r1> - <r2>
mult r1 r2 r3	<r3> := <r1> * <r2>

In diesen Befehlen benötigt ein Operationscode 8 Bit, eine Registeradresse 4 Bit und eine Speicheradresse 32 Bit. Die Ausführungszeit der Befehle beträgt 10 Takte und zusätzlich 4 Takte je Speicherzugriff (Register- und Befehlszugriffe sind ausgenommen). So benötigen **load** und **store** zum Beispiel je 14 Takte.

1. Geben Sie eine Befehlsfolge an, welche den oben angegebenen Ausdruck Z berechnet.
2. Welchen Platzbedarf in Bits hat die gesamte Befehlsfolge?
3. Wieviel Takte benötigt die Ausführung der Befehlsfolge auf einem seriell arbeitenden Prozessor (d.h. die Ausführung einzelner Befehle überlappt nicht)?

Der Prozess P1 soll nun zusätzlich über die folgenden Befehle verfügen:

Befehl	**Wirkung**
madd r m	<r> := <r> + <m>
msub r m	<r> := <r> - <m>
mmult r m	<r> := <r> * <m>

4. Geben Sie eine Befehlsfolge an, die so weit wie möglich ohne load/store-Befehle auskommt und ebenfalls Z berechnet.
5. Welchen Platzbedarf in Bits hat diese Befehlsfolge?
6. Wieviele Takte benötigt die Ausführung der Befehlsfolge auf einem seriell arbeitenden Prozessor?

Alternativ zu der oben beschriebenen Maschine wird nun ein anderer Prozessor P2 betrachtet, der den angegebenen Ausdruck Z mit Hilfe eines Stacks berechnen soll. Folgende Befehle stehen ihm dabei zur Verfügung:

Befehl	**Wirkung**
push m	legt den in Zelle m stehenden Wert auf dem Stack ab
pop	bringt den obersten Wert vom Stack in die Zelle, deren Adresse in der zweitobersten Zelle des Stacks steht; beide Zellen des Stacks werden gelöscht.
pusha m	legt die Adresse m auf dem Stack ab

Die drei zusätzlichen aritmetischen Operationen **add**, **sub**, **mult** verknüpfen die beiden oberen Werte des Stacks, löschen sie und legen das Ergebnis wiederum auf den Stack. Bei der Subtraktion wird der oberste Wert des Stacks als Minuend behandelt. Für den Speicherbedarf der Befehle gelten die Konventionen von P1.

7. Zeichnen Sie einen Operatorbaum für den Ausdruck Z.
8. Schreiben Sie Z in Postfixform.
9. Geben Sie eine Befehlsfolge für P2 an, welche Z berechnet.
10. Welchen Platzbedarf in Bits hat Ihre Befehlsfolge aus der vorigen Teilaufgabe?
11. Welchen Zeitbedarf hat die Ausführung der Befehlsfolge, wenn **push** m und **pop** je 14 und die übrigen Befehle je 10 Takte brauchen, auf einem seriell arbeitenden Prozessor?

Aufgabe 4 Seitenersetzungsstrategien

Der Prozess p arbeite mit einem virtuellen Speicher von fünf Seiten. Der Zugriff auf diese Seiten finde in der durch die folgende Seitenreferenzkette w gegebenen Reihenfolge statt:

w = 0 1 2 3 0 1 4 0 1 2 3 4.

1. Für die Realisierug der Speicherfähigkeit des Prozesses p sollen zunächst drei Kacheln zur Verfügung stehen. Geben Sie für die beiden Ersetzungsstrategien LRU (least recently used) und FIFO (first in first out) jeweils die Entwicklung der Seiten-Kachel-Tabelle und die Anzahl der aufgetretenen Seitenfehler an. Markieren Sie jedes Auftreten eines Seitenfehlers!
2. Beschreiben Sie kurz die (theoretisch) beste Seitenersetzungsstrategie. Warum ist diese Strategie nicht realisierbar?
3. Erklären Sie den Begriff der Keller-(Stack-)Strategie und zeigen Sie, dass FIFO keine Kellerstrategie ist, indem Sie den Prozess p mit der angegebenen Seitenreferenzkette mit vier Kacheln realisieren.

Aufgabe 5 Prozesssysteme

Gegeben sei folgende Prozedur:

```
PROCEDURE berechnen (x, y, z : IN INTEGER; a : OUT REAL);
VAR b, c, d : INTEGER;
BEGIN
     B := x + y ; c := b * b ; d := b –z ; a := c / d ;
END;
```

Diese Prozedur soll nun mit Hilfe eines deterministischen Prozesssystems

$P = \{D, P, <^*, V, N\}$

modelliert werden.

1. Geben Sie die Menge der verwendeten Betriebsmittel (Speichervariablen) D und die Menge der benötigten Prozesse an. Die Prozesse aus P sollen dabei atomar sein (d.h. nur eine Rechenoperation ausführen).
2. Geben Sie für jeden Prozess $p \in P$ die jeweilige Rechenfunktion R_p sowie seinen Vorbereich V_P und seinen Nachbereich N_P an.
3. Aus Effizienzgründen sollen soviele Prozesse aus P wie möglich parallel ausgeführt werden können. Stellen Sie eine geeignete Vorrangrelation $<^*$ auf und zeichnen Sie den Vorranggraphen.
4. Beweisen Sie, dass Ihr System determiniert ist.

2.14 Herbst 1998 II

Aufgabe 1

1.1 Beschreiben Sie kurz die folgenden Prozessorvergabestrategien (Scheduling Algorithmen) sowie deren Vor- und Nachteile:
FCFS (first come first served), SJF (shortest job first) und RR (round robin)
1.2 Was versteht man unter Job-Scheduling (Longterm-Scheduling)?
1.3 Welche Aufgabe hat der Prozessumschalter (Dispatcher)?
2.1 Erklären Sie die Begriffe „Seite“ und „Kachel“.
2.2 Wie ist eine logische (virtuelle) Adresse aufgebaut (nur reines Paging)? Skizzieren Sie den Abbildungsmechanismus von logischen auf physikalische Adressen.
2.3 Erklären Sie den Seitenaustauschalgorithmus LRU (least recently used).
2.4 Was versteht man unter Seitenflattern (Thrashing) und was kann man dagegen tun?
3.1 Formulieren Sie umgangssprachlich in Stichpunkten die Schritte, die vom Kommandointerpreter eines Multi-User/Multi-Tasking Betriebssystems (wie etwa UNIX) zur Entgegennahme eines Kommandos durchgeführt werden.
3.2 Wie geht ein Kommandointerpreter vor, um ein Kommando „im Hintergrund“ auszuführen?

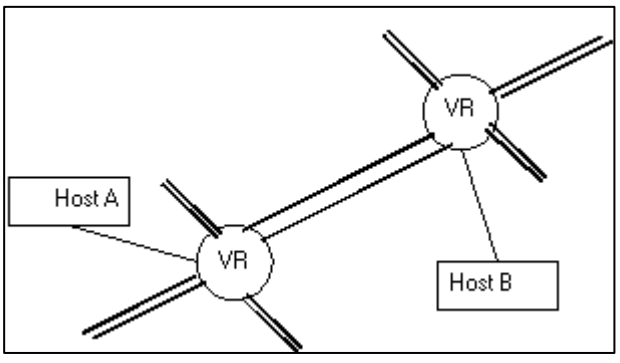
Aufgabe 2

1. Stellen Sie an einem selbstgewählten Beispiel den relationalen Normalisierungsvorgang bis hin zur dritten Normalform dar.
2. Definieren Sie die wichtigsten relationalen Operatoren.
3. Zeigen Sie an einem Beispiel, wie die Sprache SQL in eine Programmiersprache eingebettet wird.

Aufgabe 3

ISO/OSI-Kommunikationsarchitektur
Kommunikationsnetze werden immer leistungsfähiger, größer und komplexer; gefordert wird von einer Systemarchitektur, dass sie universell verwendbar, flexibel anpassbar und beherrschbar ist.

1. Nennen Sie die vier grundlegenden Strukturierungskonzepte, die diesen Forderungen beim gesamten Entwurf, bei der Implementierung und beim Betrieb Rechnung tragen.
2. Was verstehen Sie unter ISO/OSI-Schichtenmodell bzw. Dm ISO/DIN-Basis-Referenzmodell für offene Kommunikation? Welche der oben besprochenen Strukturierungsprinzipien werden dabei benutzt? Welche Dienstleistung ist der Schicht 4 zugeordnet?
3. Entwerfen Sie zu dem untern gezeigten Netzausschnitt inkl. Anwendungsrechner die entsprechende funktionelle Struktur entsprechend dem ISO/DIN-Basisreferenzmodell bzw. Dem ISO/OSI-Schichtenmodell. Zeigen Sie den Weg auf, den die zwischen zwei Anwenderprozessen zu übertragende Nutzinformation durch die einzelnen Schichten nimmt.



Netzmanagement
Unter dem Begriff Netzmanagement werden alle technischen und organisatorischen Vorkehrungen und Aktivitäten zum Management eines Kommunikationssystems zusammengefasst.

4. Nennen Sie die unterschiedlichen Managementbereiche des funktionalen Modells. Beschreiben Sie weiterhin stichwortartig deren Aufgaben.
5. Auf allen Komponenten des dargestellten Netzausschnitts laufen Management-prozesse ab! Welche Klassen von Managementprozessen unterscheiden wir beim sogenannten organisatorischen Modell?
6. Wie sieht jetzt die Gesamtarchitektur für Anwender- und Managementkommunikation innerhalb des Netzausschnitts aus?

2.15 Frühjahr 1999 I

Aufgabe 1

Virtuelle Adressen

1. Wann tritt ein Seitenfehler (page fault) auf?
2. Beschreiben Sie die Aktionen, die das Betriebssystem nach Auftreten eines Seitenfehlers ausführen muss!

Plattenspeicherzugriffe

3. Welche Kriterien sind bei der Auswahl einer Plattenspeicher-Zugriffsstrategie zu beachten?

Dateisystem

4. Was sind die wesentlichen Aufgaben eines Dateisystems?
5. Erläutern Sie die folgenden Begriffe und ihren Zusammenhang an Hand eines der Betriebssysteme: UNIX, Windows oder NT
   - Datei
   - Katalog (Directory)
   - Baumförmiger Namensraum
   - Pfad in diesem Namensraum
   - Partition
6. Geben Sie einen Überblick über die wesentlichen Dateiattribute (in dem von Ihnen gewählten Betriebssystem). Wie werden diese Dateiattribute verwaltet und was ist jeweils ihre Funktion?

Aufgabe 2

1.1 Relationales Datenmodell

Nennen und erläutern Sie kurz die drei grundlegenden relationalen Operatoren!
Erläutern Sie informal die Begriffe:
Schlüsselkandidat, Primärschlüssel, Fremdschlüssel und referentielle Integrität!

1.2 SQL-Anfragen

Für den Einkauf in einer Maschinenfabrik wird folgende Datenbank zur Verwaltung der Bestände verwendet:

LIEFERANT (Name, Firmensitz, Ansprechpartner)
TEIL (Teil-Id, Bezeichnung)
LIEFERUNG (Lieferant-Id, Teil-Id, Lieferdatum, Stückzahl)

Die Primärschlüssel der Relationen sind unterstrichen. Lieferant-Id in LIEFERUNG ist Fremdschlüssel zu Name in LIEFERANT. Teil-Id in LIEFERUNG ist Fremdschlüssel zu Teil-Id in TEIL.
Formulieren Sie die folgenden Datenbankoperationen in SQL.

Welche Lieferanten haben ihren Firmensitz in Erlangen, Nürnberg oder Fürth?
Wie sind die Bezeichnngen der Teile, bei denen eine Lieferung weniger als 10 Stück umfasste?

Was bedeutet umgangssprachlich folgende Anfrage:

```
SELECT    lf.name
FROM      Lieferant lf, Lieferung lg, Teil t
WHERE     lf.name = lg.Lieferant-Id AND t.Teil-Id = lg.Teil-Id AND
          t.Bezeichnung = ‚Bolzen' AND lg.Stückzahl>1000
```

2.1 Aus welchen Gründen wird eine Relation normalisiert?

2.2 Definieren Sie kurz Eigenschaften einer Relation in erster, zweiter und dritter Normalform?

2.3 Bringen Sie die folgende Relation
Verkäufe (Anr, Bezeichnung, Verkäufer (Verkäufername, Geburtstag, Geschäft, Geschäftsführer, Anzahl))
in die erste, zweite und dritte Normalform.

VERKÄUFE	ANR	BEZ	Verkäufer	VNAME	GEBTAG	GESCH	GESCHF	ANZ
	4711	Video x		F. Meier	12.12.50	Tele2000	Maier	22
	4711	Video x		H.Müller	01.01.55	Tele2000	Maier	11
	4712	Radio y		F. Huber	10.10.60	Tele2000	Maier	5
	4712	Radio y		H. Anton	02.02.65	Tele2000	Maier	17
	4713	Monitor z		H. Nie	08.08.70	Compu100	Schmidt	23
	4713	Monitor z		F. Immer	09.09.75	Compu100	Schmidt	37

3.1 Benennen und erläutern Sie die klassischen Eigenschaften, die von einer Datenbanktransaktion gefordert werden!

3.2 Im Datenbankbereich können 4 klassische Fehlerarten unterschieden werden.
Beschreiben Sie die Fehlerarten und die Recoverymaßnahmen, die notwendig werden, um die Transaktionseigenschaften sicherzustellen.

3.3 Zur Sicherung der Transaktionseigenschaften muss das Datenbanksystem Protokollinformationen sammeln, die zur Wiederherstellung der Datenbank im Fahlerfall dienen. Die dabei verwendeten Protokollierungsverfahren werden in physische und logische Protokollierungsverfahren eingeteilt.
Charakterisieren Sie beide Verfahren kurz und beschreiben Sie die jeweiligen Vor- und Nachteile.

Aufgabe 3

Lokale Netze

1. Die Schicht 2 des ISO/OSI-Referenzmodells wird in lokalen Netzen in zwei Unterschichten unterteilt. Wie heißen die zwei Schichten?
2. Warum ist es sinnvoll zwei Unterschichten zu haben?
3. Welches sind die Standard-Dienste der einen, welches die Standard-Dienste der anderen Schicht?

Kooperation durch Nachrichtenaustausch

4. Bei der Übermittlung von Nachrichten können eine Reihe von Fehlern passieren, die zu einer Verfälschung des Inhalts oder gar zum Verlust der Gesamtnachricht führen. Um welche Fehler handelt es sich hierbei?
5. Nachdem ein Übermittlungsfehler erkannt wird, wird die Korrektur normalerweise durch die Wiederholung der Nachricht erreicht. Welche Mechanismen kennen Sie, um die Wiederholung der Nachricht auszulösen? Beschreiben Sie kurz, wie die Mechanismen arbeiten.
6. Wenn die Kommunikationszeiten in einem Netzwerk schwanken, kann es passieren, dass Nachrichten sich gegenseitig überholen. Welche Mechanismen kann man benutzen, um dieses Problem zu lösen? Erläutern Sie knapp die grundsätzlichen Prinzipien der Verfahren.

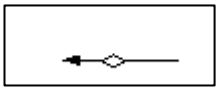
2.16 Frühjahr 1999 II

Aufgabe 1

Gegeben sei die folgende relationale Datenbank *Buch*, in der die Bücher und Autoren eines Verlages verwaltet werden. Jedes Buch hat eine eindeutige Nummer (*BNr*). Ein Buch kann von mehreren Autoren verfasst werden und wird einem bestimmten Gebiet zugeordnet. Jeder *Autor* ist durch seine Nummer (*Anr*) eindeutig bestimmt. Im Attribut *Fnr* werden die Bücher eines Autors fortlaufend durchnummeriert. Neben dem *Datum* werden die Anzahl der *Seiten* und der *Preis* eines Buches bei jeder *Auflage* neu festgelegt.

Bnr	Anr	Autor	Titel	Auf-lage	Gebiet	Fnr	Datum	Seiten	Preis
1	1	Lang	Datenbanksysteme	1	DBS	1	1996	448	68,-
1	2	Reiter	Datenbanksysteme	1	DBS	1	1996	448	68,-
2	1	Lang	OODB	1	DBS	2	1994	356	62,-
3	3	Dedos	Deductive Databases	1	DBS	1	1991	414	70,-
3	3	Dedos	Deductive Databases	2	DBS	1	1995	435	75,-
4	1	Lang	Programmierung in C	1	PRO	3	1998	267	58,-
5	3	Dedos	Logic Programming	1	PRO	2	1997	236	49,-
6	3	Dedos	Expertensysteme	1	KI	3	1999	567	87,-
2	1	Lang	OODB	2	DBS	2	1997	425	60,-
7	4	Meyer	Einführung in die Informatik	1	ALL	1	1993	347	36,-
8	5	Reiter	Informatik II	1	ALL	1	1993	198	15,-
8	5	Reiter	Informatik II	2	ALL	1	1996	210	21,-
8	5	Reiter	Informatik II	3	ALL	1	1997	233	23,-
9	5	Reiter	Lexikon Informatik	1	ALL	2	1996	896	136,-
9	4	Meyer	Lexikon Informatik	1	ALL	2	1996	896	136,-

1. Beschreiben Sie kurz, welche Redundanzen (i) bei mehreren Auflagen und (ii) bei mehreren Autoren eines Buches in der Datenbank vorhanden sind.
2. Welche Typen von Anomalien können bei diesem Relationenschema auftreten? Geben Sie jeweils ein Beispiel an.
3. Geben Sie für obige Datenbank alle vollen funktionalen Abhängigkeiten an.

4. Zeigen Sie an Hand eines Beispiels, dass die obige Datenbank die dritte Normalform verletzt.
5. Überführen Sie das Schema in die dritte Normalform. Skizzieren Sie die resultierende Datenbank, indem Sie die Schlüsselwerte der Tupel in die neuen Tabellen eintragen.
6. Geben Sie ein Entity-Relationship-Diagramm an, das die Struktur des neuen Schemas erkennen lässt. One-to-many Relationships werden als  dargestellt, bei many-to-many Relationships sind die Kanten ungerichtet.
7. Geben Sie eine SQL-Anweisung für die Datenbank in dritter Normalform an, die eine Sicht auf das ursprüngliche Schema definiert.
8. Erklären Sie kurz, welchen Nachteil Normalisierung allgemein für die Anfragebearbeitung hat.

Aufgabe 2

Das folgende relationale Schema beschreibt eine Getränkedatenbank, in der das Angebot verschiedener Lokale verwaltet wird. Zu jeder Sorte, die eindeutig durch die Nummer (*Snr*) bestimmt ist, wird der Name des Herstellers (*Hname*), der Sortenname (*Sname*) sowie die kcal pro Liter (*kcal_liter*) gespeichert. Ferner besitzt ein Hersteller keine Sorten mit demselben Namen. Jedes Lokal hat einen Namen (*Lname*) und ist mit Postleitzahl (*PLZ*) und *Ort* in der Datenbank enthalten. Die Relation Angebot beschreibt, zu welchem *Preis* ein Lokal ein bestimmtes Getränk verkauft. Der Preis bezieht sich jeweils auf einen viertel Liter.

Sorte (Snr, Hname, Sname, kcal_liter)
Angebot (Snr, Lnr, Preis)
Lokal (Lnr, PLZ, Ort, Lname)

1. Formulieren Sie die folgenden Anfragen in zwei der vier Anfragesprachen: *relationale Algebra, relationaler Tupelkalkül, SQL* und *Quel.* Bei der Verwendung von SQL oder Quel sind die Duplikate zu entfernen. Geben Sie DISTINCT bzw. UNIQUE für SQL bzw. Quel nur an, wenn Duplikate auftreten können.
   a) Bestimme die Namen aller Sorten, die in Münchner Lokalen angeboten werden.
   b) Gibt es Sortennamen, die von verschiedenen Herstellern verwendet werden? Geben Sie alle Daten der entsprechenden Sorten aus.
   c) Bestimme die Namen der Sorten des Herstellers XYZ, die in keinem Lokal teurer als 5,- DM sind.
2. Formulieren Sie die folgenden Anfragen in SQL. Verwenden Sie dabei die Aggregatfunktionen `avg, count, min` oder `max`.
   a) Bestimme den durchschnittlichen Preis, zu dem die Sorte mit Nr. 27 in München angeboten wird.
   b) Bestimme für jeden Hersteller den maximalen Gehalt an kcal pro Liter aller hergestellten Sorten.
   c) Herr Schmidt sucht in seiner Umgebung (PLZ 80428) ein Lokal, bei dem der Preis für die Sorte mit Nr. 4 unter dem Münchener Durchschnittspreis für dieses Getränk liegt. Geben Sie eine Liste der entsprechenden Lokale aus.

Aufgabe 3

Speicherhierarchie

1. Motivieren Sie die Einführung von Speicherhierarchien.
2. Nennen Sie die grundlegende Voraussetzung für deren sinnvolle Verwendung.

3. Geben Sie eine typische Hierarchie mit den zugehörigen Speichertypen und Größen- bzw. Zugriffszeitbereichen an.
4. Ordnen Sie Caches in die Speicherhierarchie ein, falls nicht bei Beantwortung der vorigen Teilaufgabe bereits erfolgt.
5. Beim Schreibenden Zugriff auf einen Cache kann man bzgl. der Behandlung des Hauptspeichers zwei Organisationsformen unterscheiden:
   write through: Bei jedem Schreibvorgang wird auch der Hauptspeicher aktualisiert (auch *als store through* bekannt)
   write back: Hauptspeicher nur aktualisiert, wenn nötig (auch als *copy back*, *write later* bezeichnet)
   Diskutieren Sie Vor- und Nachteile beider Verfahren. Berücksichtigen Sie dabei auch Mehrprozessor-Anlagen.
6. Für den Fall eines *write miss* (benötigter Block für den schreibenden Zugriff nicht im Cache) bestehen zwei prinzipielle Optionen:
   - *write allocate*: Der Block wird in den Cache geladen und dort geändert.
   - *no write allocate*: Der Block wird direkt im Hauptspeicher modifiziert und nicht in den Cache geladen.

Obwohl natürlich prinzipiell beide Optionen sowohl bei *write through* als auch bei *write back* verwendet werden können, wird in der Regel jeweils eine der Optionen bevorzugt. Welche? Warum?

Aufgabe 4

Adressierungsarten

1. Welche Befehlsadressformate kennen Sie? Beschreiben Sie diese und charakterisieren Sie die wesentlichen Unterschiede.
2. Nennen Sie wenigstens 5 Adressierungsformen und beschreiben Sie diese genauer.

Aufgabe 5

Parallele Rechensysteme

1. Nennen Sie die wesentlichen Unterschiede zwischen einem SIMD- und einem MIMD-Rechner!
2. Beschreiben Sie Vor- und Nachteile eines Mehrprozessorsystems mit gemeinsamen Hauptspeicher im Vergleich zu einem solchen mit lokalen Speichern für jeden Prozessor!
3. Erläutern Sie die Unterschiede zwischen einem eng-gekoppelten und einem lose-gekoppelten Mehrprozessorsystem!

2.17 Herbst 1999 I

Aufgabe 1

1. Übertragungssysteme
   Die Informationsübertragung in Kommunikationsnetzen basiert auf dem physikalischen Phänomen der Ausbreitung von elektromagnetischen Wellen.

1.1 Nennen Sie die vier wichtigsten Übertragungsmedien und kennzeichnen Sie kurz deren wichtigste Eigenschaften.

1.2 Alle oben diskutierten Übertragungsmedien haben eine Bandbreite, die in vielen Fällen weit über das hinausgeht, was für eine einzelne Kommunikationsverbindung notwendig ist. Zur wirtschaftlichen Nutzung verwendet man deshalb eine spezielle Technik.
Nennen Sie diese Technik, entwerfen Sie eine Prinzipskizze mit den wesentlichen Komponenten und erläutern Sie kurz die Grundidee.

1.1 Bei der Zeitmultiplextechnik gibt es grundsätzlich unterschiedliche Zuordnungsprinzipien.
Nennen Sie diese Prinzipien und erläutern Sie sie, ggf. mit einer Skizze.
Was verstehen Sie unter ATM? In welche der genannten Klassen fällt ATM?

2. Weitverkehrsnetze

2.1 Was verstehen Sie unter einem Netz, was unter einem Vermittlungssystem?

2.2 Netze werden oft nach der Art der End-zu-End-Vermittlung klassifiziert.
Diskutieren Sie kurz die Grundprinzipien der (beiden) Techniken.

2.3 Sie wollen eine längere Datei zu einem entfernten Datenbankrechner übertragen.
Welches Grundprinzip würden Sie bevorzugen?
Begründen Sie Ihre Wahl mit einigen Stichworten.

3. Funktioneller Aufbau von Weitverkehrsnetzen
Die Charakterisierung der unterschiedlichen Teilaufgaben erfolgt meist mit Hilfe des ISO/OSI-Referenzmodells. Normalerweise umfasst das Funktionsmodell für die Endsysteme alle sieben Schichten, für die Vermittlungssysteme die unteren drei Schichten.

3.1 Nennen Sie die Aufgaben von Schicht 3.

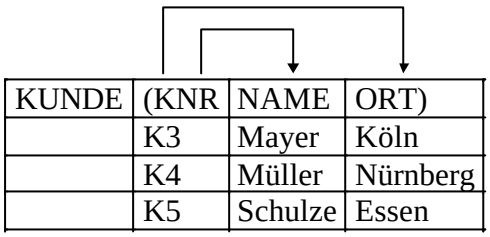
3.2 Wie erklären Sie sich, dass manchmal auch bei den Vermittlungsknoten die Schichten vier bis sieben dargestellt werden? Ist das gerechtfertigt?

Aufgabe 2

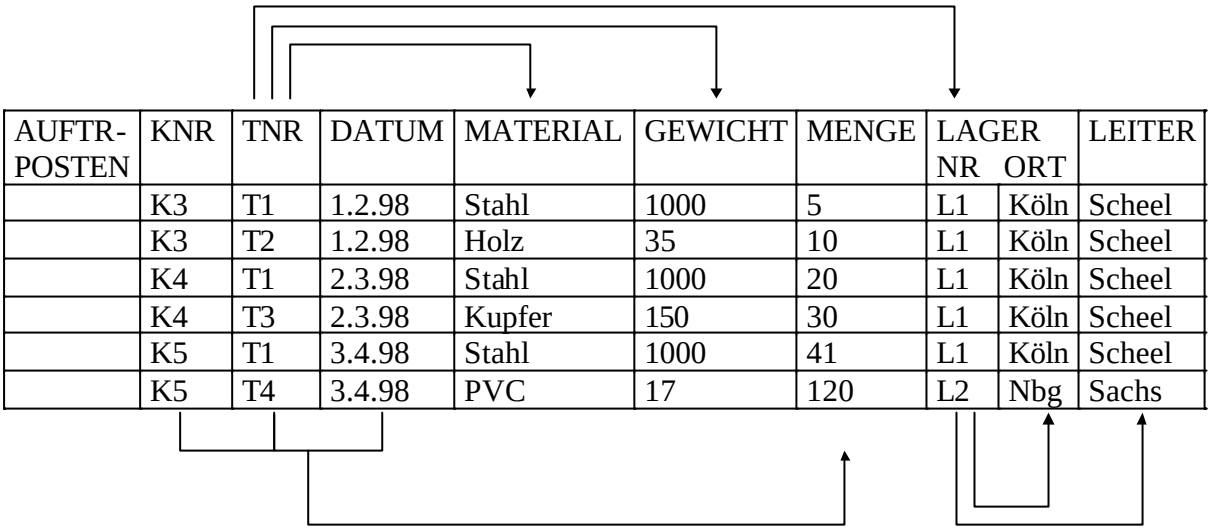
2. Gegeben seien die normalisierten Relationen “KUNDE” und “AUFTR-POSTEN” sowie die funktionalen Abhängigkeiten ihrere Attribute. Primärschlüssel sind unterstrichen.
Man überführe die Relationen in die

1.1 zweite Normalform

1.2 dritte Normalform



KUNDE	(KNR	NAME	ORT)
	K3	Mayer	Köln
	K4	Müller	Nürnberg
	K5	Schulze	Essen



AUFTR-POSTEN	KNR	TNR	DATUM	MATERIAL	GEWICHT	MENGE	LAGER NR	LAGER ORT	LEITER
	K3	T1	1.2.98	Stahl	1000	5	L1	Köln	Scheel
	K3	T2	1.2.98	Holz	35	10	L1	Köln	Scheel
	K4	T1	2.3.98	Stahl	1000	20	L1	Köln	Scheel
	K4	T3	2.3.98	Kupfer	150	30	L1	Köln	Scheel
	K5	T1	3.4.98	Stahl	1000	41	L1	Köln	Scheel
	K5	T4	3.4.98	PVC	17	120	L2	Nbg	Sachs

3. Gegeben seien die Relationen

PERSONAL (	PNR,	NAME,	BERUF,	ANR)
	P1	Müller	Schlosser	A1
	P2	Mayer	Schreiner	A1
	P3	Schulze	Förster	A2
	:	:	:	:

ABTEILUNG (	ANR,	BEZ,	ORT)
	A1	Reparatur	Köln
	A2	Holzbetrieb	München
	:	:	:

Man formuliere die folgenden Anfragen in SQL:

2.1 In welcher Abteilung < ANR, BEZ > arbeitet eine Person mit dem Namen Müller?

2.2 Welche Personen < PNR, NAME > arbeiten in einer Abteilung in Köln?

2.3 Welche Abteilungen < ANR > haben überhaupt kein Personal?

Aufgabe 3

1.1 Lösen Sie das folgende Koordinierungsproblem:
Eine Familie bestehe aus Vater, Mutter und mehreren Kindern. Die Familie besitzt gemeinsam nT Teller, nG Gabeln und nM Messer. Zusätzlich hat der Vater ein eigenes Messer, die Mutter eine eigene Gabel und jedes Kind einen eigenen Teller. Keiner von ihnen gibt seinen eigenen „Gegenstand“ aus der Hand.
Wenn eines der Familienmitglieder essen will, braucht es drei Dnge: Teller, Messer und Gabel, d.h. zum eigenen Gerät jeweils zwei aus dem Gemeinschaftsbesitz.
Lösen Sie den Zugriff auf die gemeinsamen Gegenstände mit einem geeigneten Koordinierungsmechanismus so, dass keine Verklemmung entstehen kann.

Dabei sollen natürlich möglichst viele Personen gleichzeitig essen können.

1.2 Beschreiben Sie, wie Sie bei Ihrer Lösung eine Verklemmug vermieden haben.

1.1 Was versteht man unter Scheduling?
Beschreiben Sie kurz die folgenden Scheduling Algorithmen sowie deren Vor- und Nachteile:
FCFS (First Come First Served), SJF (Shortest Job First) und RR (Round Robin).

1.2 Welcher der angegebenen Algorithmen minimiert die mittlere Wartezeit?

1.3 Welche Aufgabe hat der Prozessumschalter (Dispatcher)?

2.18 Herbst 1999 II

Aufgabe 1

Abteilungsdatenbank

Gegeben sei das folgende Schema einer relationalen Datenbank, in dem beschrieben wird, in welcher Abteilung ein Angestellter arbeitet, sowie welche Produkte zu einer Abteilung gehören:

Angestellter	(AngNr, AngName, Gehalt, AbtNr)
Abteilung	(AbtNr, AbtName, Adresse)
Produkt	(PNr, AbtNr, PName, Preis, Farbe)

1. Entity-Relationship-Diagramm
Geben Sie für das obige Relationenschema ein ER-Diagramm an.
Beschreiben Sie kurz die grafischen Elemente von ER-Diagrammen und deren Bedeutung.

2. Relationale Anfragen
Formulieren Sie die folgenden Anfragen in zwei der folgenden Sprachen:
Relationale Algebra, Tupelkalkül, Bereichskalkül, SQL, Quel.
Verwenden Sie im Tupelkalkül für die Ergebnisrelation die 1-attributige Relation name mit Schema (name) = (Name):

2.1 Bestimme die Namen aller Produkte, die teurer als 100 DM sind.

2.2 Bestimme die Namen der Angestellten, in deren Abteilung blaue Produkte angeboten werden.

3. Aggregatfunktionen
Formulieren Sie die folgenden Anfragen in *SQL* oder in *Quel.*
Verwenden Sie dabei die Aggregatfunktionen avg, count, min oder max.

3.1 Bestimme das Durchschnittseinkommen der Mitarbeiter in der Abteilung ‚Spielzeug'.

3.2 Bestimme zu jeder Abteilung (Nummer genügt) den Preis des teuersten Produktes.

3.3 Bestimme die Namen der Abteilungen, in denen alle Mitarbeiter über 3000 DM verdienen.

Aufgabe 2 Normalformen

Gegeben sei die folgende relationale Datenbank mit den offenen Rechnungen der Kunden eines Versandhauses:

Rechnung	Rnr	Kdnr	Name	Adresse	Positionen	Datum	Betrag
	1	1	Müller	München	3	01.11.94	60
	2	1	Müller	München	2	23.05.95	90
	3	2	Huber	Nürnberg	3	09.03.95	90
	4	2	Huber	Nürnberg	8	14.02.95	70
	5	3	Meier	Hamburg	7	20.06.95	110
	6	4	Meier	München	12	07.04.95	90

1. Beschreiben Sie kurz, welche Redundanzen in der Datenbank vorhanden sind, und welche Anomalien bei einem solchen Relationenschema auftreten können.
2. Erklären Sie kurz, welchen Nachteil Normalisierung allgemein für die Anfragebearbeitung hat.
3. Geben Sie für obige Datenbank alle vollen funktionalen Abhängigkeiten (einschließlich der transitiven) an.
4. Erfüllt die Datenbank die zweite Normalform? Begründen Sie Ihre Antwort.
5. Zeigen Sie, inwiefern die obige Datenbank die dritte Normalform verletzt.
6. Überführen Sie das Schema in die dritte Normalform und geben Sie die resultierende Datenbank an.
7. Geben Sie eine SQL-Anweisung für die Datenbank in dritter Normalform an, die eine Sicht auf das ursprüngliche Schema definiert.
8. Geben Sie für diese Datenbank ein Schema in Boyce/Codd-Normalform an (mit Begründung).

Aufgabe 3

1. Was versteht man unter multiplexen?
2. Beschreiben Sie kurz die Multiplexverfahren, die Sie kennen.
3. Welches Multiplexverfahren verwendet ATM?
4. Welches Multiplexverfahren verwendet Ethernet?

Aufgabe 4 Speicherverwaltung

In dieser Aufgabe sei für die Speicherverwaltung eines Rechensystems einfache Segmentierung vorgesehen. Für jedes Segment eines Prozesses muss ein Eintrag in der Segmenttabelle des Prozesses existieren. Logische und physikalische Adressen haben jeweils eine Länge von 20 Bit, von denen 5 Bit für die Segmentnummer reserviert sind.

1. Welche Informationen über die Segmente müssen in der Segmenttabelle enthalten sein?
2. Zeigen Sie (mit Hilfe einer Zeichnung) die Abbildung einer logischen auf eine physikalische Maschinenadresse.
3. Welche Art der Fragmentierung tritt bei der hier angenommenen einfachen Segmentierung auf?
4. Beschreiben Sie kurz (jeweils ein Satz) die Ihnen bekannten Fragmentierungsarten.
5. Was ist bei den oben angenommenen Werten der Maximalwert für die Segmentgröße?
6. Welche Überprüfungen kann man vornehmen, um festzustellen, ob es sich bei einer angegebenen logischen Adresse um eine gültige Adresse handelt?

Aufgabe 5 Synchronisation

Gegeben sei ein Erzeuger/Verbraucher-System mit einem Erzeuger und zwei Verbrauchern. Der Erzeuger erzeugt fortlaufend natürliche Zahlen, die er in der gemeinsamen Variablen x ablegt (Puffer der Kapazität 1). Die Verbraucher lesen den Wert aus x und verarbeiten ihn.

Geben Sie unter der Verwendung von Semaphoren ein System an, so dass die Prozesse Verbraucher[1] und Verbraucher[2] abwechselnd die vom Erzeuger generierten Werte verarbeiten (nämlich Verbraucher[1] den ersten, dritten, fünften, ... Wert und Verbraucher[2] den zweiten, vierten, sechsten, ... Wert). Dabei soll der Zugriff auf die gemeinsame Variable x unter gegenseitigem Ausschluss erfolgen.

Aufgabe 6 Von-Neumann-Modell

1. Beschreiben Sie kurz die wesentlichen Komponenten des Von-Neumann-Modells. Nennen Sie die wichtigsten Eigenschaften.
2. Beschreiben Sie die einzelnen Phasen der Befehlsverarbeitung innerhalb eines Von-Neumann-Rechners.
3. Wie lange dauert es maximal bis ein Befehl, der gerade zur Bearbeitung ansteht, ausgeführt wird? Geben Sie eine obere Schranke an.
4. Was ist der Von-Neumann-Flaschenhals? Welche prinzipiellen Möglichkeiten gibt es, das Von-Neumann-Modell zu verbessern?

2.19 Herbst 2000 I

Aufgabe 1 Mikroprogrammierung

1. Vergleichen Sie festverdrahtete mit mikroprogrammierter Steuerung.
2. Erklären Sie horizontale und vertikale Mikroprogrammierung, beschreiben Sie die Unterschiedeund leiten Sie daraus Vor- und Nachteile ab.
3. Welche Art der Steuerung erscheint Ihnen für eine RISC-Architektur am besten geeignet? Begründen Sie Ihre Aussage!

Aufgabe 2 Vermittlungsarten

1. Was sind Merkmale von verindungsorientierter und verbindungsloser Übertragung?

2. Erläutern Sie die verschiedenen Vermittlungsarten:
2.1 Leitungsvermittlung bzw. Durchschaltevermittlung
2.2 Nachrichtenvermittlung
2.3 Paketvermittlung

3. Erklären Sie die Begriffe:
3.1 virtuelle Verbindung
3.2 Datagramm

Aufgabe 3 Routing

1. Was versteht man unter Wegewahl (Routing) und welche Ziele werde hierbei verfolgt?
2. Auf welcher/welchen Ebene/n des OSI Referenzmodells findet eine Wegewahl statt?
3. Nennen Sie Methoden zur Wegewahl und klassifizieren Sie diese.
4. Beschreiben Sie den Vorgang des „directory-routing“.

5. Was versteht man unter der „hot-potato“-Technik?

Aufgabe 4 Medienzugriffsverfahren

1. Nennen Sie drei Medienzugriffsverfahren und diskutieren Sie ihre charakteristischen Eigenschaften.
2. Welche Varianten des CSMA/CD-Verfahrens kennen Sie? Warum muss bei dem CSMA/CD-Verfahren der Konfliktparameter K<<1 sein?
   ($K = \frac{\text{max. Signallauf zeit}}{\text{Nachrichte nübertragu ngszeit}}$)
3. Welches Zugriffsverfahren wird in einem FDDI-LAN verwendet? Nennen Sie charakteristische Eigenschaften eines FDDI-LANs.

Aufgabe 5 Client-Server-Konzept

5. Was versteht man unter dem Client-Server-Modell und welchen Einfluss hat es auf die heutigen Rechnerarchitekturen?
6. Welche weiteren Strukturierungsmodelle kennen Sie für verteilte Systeme?

Aufgabe 6

Gegeben seien die Prozesse P_1, ..., P_5, die um vier Arten BM_1, ..., BM_4 von Betriebsmitteln konkurrieren. Alle Betriebsmittel seien exklusiv und nicht entziehbar. Zahlenangaben über Betriebsmittel werden im folgenden Vektor $\vec{v} = \begin{pmatrix} n_1 \\ n_2 \\ n_3 \\ n_4 \end{pmatrix}$ geschrieben, wobei die i-te Vektorkomponente für n Einheiten der Betriebsmittelart BMi (i = 1, ..., 4) steht.
Im Folgenden bezeichne

- $\vec{g}$ die Gesamtzahl aller Betriebsmittel
- $\vec{b}_j$ die Anzahl der Betriebsmittel, die Prozess P_j besitzt
- $\vec{f}_j$ die Anzahl der Betriebsmittel, die Prozess P_j zusätzlich fordert (j = 1, ..., 5)

1. Geben Sie den Prozess-Betriebsmittel-Graphen für die folgenden Werte an:

$$\vec{g} = \begin{pmatrix} 3 \\ 6 \\ 3 \\ 4 \end{pmatrix}$$

$$\vec{b}_1 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \qquad \vec{b}_2 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \qquad \vec{b}_3 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 3 \end{pmatrix} \qquad \vec{b}_4 = \begin{pmatrix} 2 \\ 4 \\ 0 \\ 0 \end{pmatrix} \qquad \vec{b}_5 = \begin{pmatrix} 0 \\ 0 \\ 2 \\ 0 \end{pmatrix}$$

$$\vec{f}_1 = \begin{pmatrix} 0 \\ 2 \\ 2 \\ 2 \end{pmatrix} \qquad \vec{f}_2 = \begin{pmatrix} 2 \\ 6 \\ 2 \\ 0 \end{pmatrix} \qquad \vec{f}_3 = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} \qquad \vec{f}_4 = \begin{pmatrix} 1 \\ 2 \\ 1 \\ 4 \end{pmatrix} \qquad \vec{f}_5 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 2 \end{pmatrix}$$

2. Untersuchen Sie mit Hilfe des üblichen Reduktionsalgorithmus, ob der Zustand in Teilaufgabe 1 verklemmungsbedroht ist. Falls nicht, geben Sie die Reihenfolge an, in der alle Prozesse beendet werden können.

Aufgabe 7

Gegeben sei ein unter Seitenadressierung laufender Prozess P, der in der folgenden Reihenfolge auf seine Seiten {A, ..., E} zugreift:

A, B, C, D, A, B, E, A, B, C, D, E

1. Stellen Sie das Verhalten der folgenden Verdrängungsstrategien dar für den Fall, dass P zu Beginn keine Kacheln im Arbeitsspeicher zugeordnet sind:

1.1 FIFO mit n = 3 Arbeitsspeicherkacheln
1.2 FIFO mit n = 4 Arbeitsspeicherkacheln
1.3 LRU mit n = 4 Arbeitsspeicherkacheln
1.4 Working-Set mit Fenstergröße k = 4

Geben Sie hierzu an, welche Seiten von P sich zu jedem Zeitpunkt im Arbeitsspeicher befinden, und wann Seite-fehlt-Alarme auftreten.
Welche Besonderheit bzgl. der Anzahl der Seite-fehlt-Alarme ergibt sich für die FIFO-Strategie?

2. Zeigen Sie, dass die LRU-Verdrängungsstrategie für jeden gegebenen Seitenreferenzstring eines Prozesses P die folgende Eigenschaft aufweist:
Mit wachsender Anzahl von Kacheln des Arbeitsspeichers nimmt (bei ansonsten gleichen Anfangsbedingungen) die Seite-fehlt-Rate des Prozesses P monoton ab.

Anleitung:
Für eine Arbeitsspeichergröße von $n \in N$ Kacheln bezeichne $S_i^{(n)}$ ($i \in N_0$) die Menge der Seiten des Prozesses P, die sich nach dem i-ten Seitenzugriff von P unter Verwendung der LRU-Strategie im Arbeitsspeicher befinden. Der Einfachheit halber sei $|S_i^{(n)}| = n$ für alle $i \in N_0$ angenommen.

Für jede Seite $s \in S_i^{(n)}$ sei $time_i^{(n)}(s)$ der Zeitstempel von s gemäß LRU-Strategie.
Zeigen Sie zunächst (durch vollständige Induktion) für alle $i \in N_0$ die folgenden Invarianten:

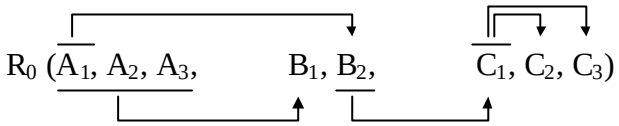
1.1 $S_i^{(n)} \subseteq S_i^{(n+1)}$
1.2 $time_i^{(n)}(s) = time_i^{(n+1)}(s)$ für alle $s \in S_i^{(n)}$
1.3 Für diejenige Seite s mit $\{s\} = S_i^{(n+1)} \setminus S_i^{(n)}$ gilt: $time_i^{(n+1)}(s)$ ist minimal unter allen Seiten aus $S_i^{(n+1)}$.

Dabei sei vorausgesetzt, dass 2.1 bis 2.3 zum Startzeitpunkt i = 0 sind.

Warum folgt aus den obigen Invarianten die zu beweisende Eigenschaft der LRU-Strategie?

Aufgabe 8

Gegeben sei eine Relation R_0 in der ersten Normalform. Der Primärschlüssel ist unterstrichen. Die funktionalen Abhängigkeiten sind durch Pfeile kenntlich gemacht.



Man überführe diese Normalform in die zweite und dritte und begründe die Schritte.

2.20 Herbst 2000 II

Aufgabe 1 Cache Memory

1. Was ist ein Cache Memory?
2. Erklären Sie folgende Einheiten und erläutern Sie deren Zweck:
   Befehls-Cache, Daten-Cache, Primär/Sekundär-Cache
3. Erklären Sie und geben Sie typische Zahlenwerte an für:
   Kapazität, Blockgröße, Trefferverhältnis (hit ratio), miss ratio
   (berücksichtigen Sie nur primäre Cache-Memories)
4. Erklären Sie: Start (Compulsory) Misses, Capacity Misses, Conflict Misses
5. Geben Sie eine Formel für die mittlere Zugriffszeit zum Cache Memory an!
6. Welchen Einfluss hat das Cache Memory auf den Durchsatz am Bus und am Speicher? Wovon hängt der Durchsatz ab?
7. Was geschieht bei Schreiboperationen, wenn der Prozessor ein Cache Memory hat? (Unterscheiden Sie, ob das zu verändernde Wort bereits im Cache Memory steht oder nicht, und berücksichtigen Sie das Konsistenzproblem!)
8. Was spricht für die Adressierung des Cache Memorys in Programm- bzw. Prozessadressen?

Aufgabe 2 Ereignis und Prozesssysteme

1. Definieren Sie: Aktion, Ereignis, Zustandsänderung.
2. Was heißt „atomar“ in diesem Zusammenhang und auf welchen der oben genannten Begriffe trifft dieses Prädikat zu ?
   Was heißt „zeitlich atomar“ und „funktionell atomar“?
3. Beschreiben Sie ein uninterpretierbares Ereignissystem einschließlich der Begriffe Vorrangrelation, Vorbereich, Nachbereich, Speicher.
4. Was ist ein interpretiertes Ereignissystem?
5. Was ist ein uninterpretiertes/interpretiertes Prozesssystem? Was sind Prozesse (im Sinne dieses Modells!), Initialisierungs- und Terminierungsereignisse?
6. Was ist eine Ausführungsfolge eines Prozesssystems?
7. Erklären Sie: konkurrente Prozesse, parallele Prozesse, sequentielle (serielle) Ausführungsfolge, lineares Prozesssystem.

8. Was ist ein determiniertes (man sagt auch: funktionales) Prozesssystem?
9. Was besagt die Bernstein-Bedingung?
10. Ein Prozesssystem umfasse die Aufträge [1..6] und die les-/schreibbaren Objekte [A..G]. Die Vorrangrelation ist ((1,2), (1,3), (1,4), (1,5), (1,6), (2,3), (2,5), (2,6), (3,5), (3,6), (4,6), (5,6)).
    Zeichnen Sie einen Vorranggraphen.
    Die Prozesse verwenden die folgenden Objekte zum Lesen bzw. Schreiben:

Prozess	Lesen	Schreiben
1	A, B, C	A
2	A	C
3	A, C, E	E, F
4	D	D
5	E	G
6	A, B	G

    Ist das Prozesssystem determiniert? Welche Paare kann man in der Vorrangrelation weglassen, ohne dass die Determiniertheit zerstört wird? Weshalb ist die Reduktion der Vorrangrelation von technischem Interesse?

Aufgabe 3 Routing in Rechnernetzen

1. Worin besteht die Aufgabe des Routings in Rechnernetzen und wodurch zeichnen sich gute Lösungen aus (nennen Sie mehrere Kriterien!)?
2. Erklären Sie: Routingtafel, Quellen-Senken-Baum.
3. Skizzieren Sie den Shortest-Delay-Time-First-Algorithmus!
4. Nennen Sie Beispiele für nicht adaptives Routing!
5. Beschreiben Sie Flooding! Mit welchen ergänzenden Maßnahmen und unter welchen Umständen ist Flooding ein sinnvolles Routingverfahren?
6. Was ist Hot-Potato-Routing?
7. Wie unterscheiden sich isoliertes adaptives Routing und gemeinsames adaptives Routing?
8. Erklären Sie OSPF und BGP!

2.21 Frühjahr 2001 I

Aufgabe 1 Kommunikation in Rechnernetzen

Ein lokales, nach außen abgeschottetes Netzwerk soll mit einem Wahlserver versehen werden. Dazu wird auf einem bestimmten Rechner ein Dienst installiert, der von allen anderen Rechnern mittels Fernaufruf (remote procedure call, RPC) genutzt wird. Der Dienst soll zwei Prozeduren zur Verfügung stellen:

- **vote**: Die Prozedur dient zur Stimmenabgabe und enthalte als Parameter den Namen des Kandidaten (als Zeichenkette) und eine Wahlnummer (als ganze Zahl), um mehrfache Stimmenabgabe verhindern zu können.
- **result**: Die Prozedur enthalte als Parameter den Namen eines Kandidaten und als Ergebnisparameter die Zahl der für ihn abgegebenen Stimmen.

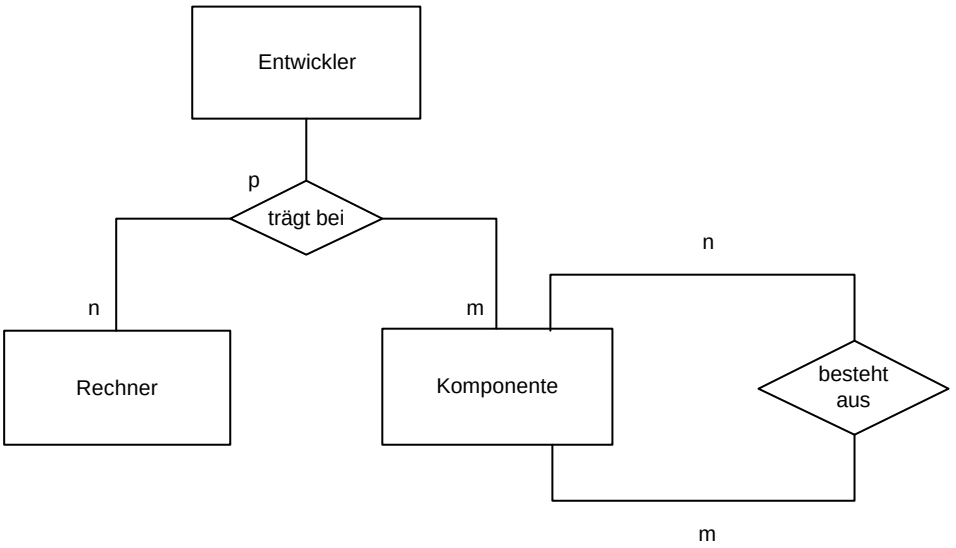
Der Server führe zwei Listen, nämlich ein Wählerverzeichnis zur Verhinderung mehrfacher Stimmenabgabe und eine Stimmenliste, in der zu jedem Kandidaten die für ihn abgegebenen Stimmen mitgezählt werden.

1. Man entwerfe für die Prozedur vote ein Flussdiagramm für den Fall, dass die Fernaufrufe vom Server serialisiert werden.
2. Welche Ergänzungen sind erforderlich, wenn Fernaufrufe nebenläufig abgewickelt werden?
3. Welche der Aufrufsemantiken **‚maybe'**, **‚at least once'** oder **‚at most once'** ist für die Lösung der Aufgabenstellung am besten geeignet? Begründung!
4. Welche zusätzlichen (Schutz-)Maßnahmen wären erforderlich, wenn das Netzwerk Unbefugten zugänglich wäre, und wie könnten sie realisiert werden?

Aufgabe 2

1. Schemaentwurf im Relationenmodell

Die Entwicklung eines großen Programmsystems soll durch eine Datenbank unterstützt werden. Sie soll festhalten, welche Komponenten von den einzelnen Entwicklern beigesteuert werden und auf welchen Rechnern die Komponenten gespeichert sind. Außerdem soll sie Auskunft über die Zerlegung von Komponenten in Teilkomponenten geben. Zur Lösung dieser Aufgabe wurde folgendes Entity-Relationship-Diagramm entwickelt:



1.1 Entwerfen Sie ein Schema nach dem Relationenmodell, das diesem Diagramm entspricht.
Bestimmen Sie die Primär- und Fremdschlüssel.
Entwickler (E) habe die Attribute ENR, Name (ENAME) und Ort (EORT).
Komponente (K) habe die Attribute KNR, Bezeichnung (KBEZ), Programmiersprache (KSPRA) und verwendeter Compiler (KCOMP).
Rechner (R) bestehe aus IP-Adresse (RADR) und Komponentennummer (KNR).
Beitrag (B) wird durch ENR, KNR und Datum (DATUM) beschrieben.
Komponentenliste (KL) hat die Attribute Ober-Komponente (OKN) und Teilkomponente (TKN).

1.2 Drücken Sie die folgenden beiden Anfragen in umgangssprachlicher Formulierung aus:
$\pi_{TKN}(KL \mid><|_{OKN=TKN} \pi_{TKN}(KL \mid><| _{OKN=TKN} \pi_{TKN} \sigma_{OKN='Abrechnung'} KL))$
$\pi_{RADR}(R \mid><|_{KNR=KNR} (B \mid><| \sigma_{EORT='Bangalore'} E))$

2. Bibliotheksdatenbank

Zur Verwaltung des Ausleihwesens benutzt eine Universitätsbibliothek folgende Datenbank:

LESER: L (LSNR, NAME, VORNAME, WOHNORT)
BUCH: B (ISBN, TITEL, VERLAG, EEMPLARE)
EXEMPLAR: E (ISBN, EXPNR, STANDORT)
AUSLEIHE: A (LSNR, ISBN, EXPNR, DATUM)

Formulieren Sie folgende Anfragen in SQL:

1.1 Welche Bücher sind an mehreren Standorten vorhanden?
1.2 Von welchen Buchtiteln sind sämtliche Exemplare ausgeliehen?

3. Algebraische Optimierung

Zur Organisation des Prüfungswesens werde folgende Datenbank benutzt:
Professor: P (PNAME, FBNR)
Student: S (MATRNR, NAME, VORNAME, FBNR)
Fachbereich: F (FBNR, FNAME, DEKAN)
Prüfung: PR (PNAME, MATRNR, NOTE)

1.1 Folgende Anfrage soll nach den Heuristiken „Selektion möglichst früh“ und „Projektion möglichst spät“ optimiert werden.

$\pi_{MATRNR,NAME,VORNAME}\ \sigma_{FNAME='Informatik'}\ S\ |><|\ F$

(Liste der Studenten, die im Fachbereich Informatik eingeschrieben sind.)

1.2 Erläutern Sie an dem Beispiel, inwiefern obige Heuristiken (evtl. Abhängig vom Mengengerüst) zu einer Optimierung der Anfrage führen.

Aufgabe 3

1. Prozess / Thread

1.1 Beschreiben Sie die Gemeinsamkeiten und Unterschiede zwischen einem Prozess und einem Thread.
1.2 Welche Zustände kann ein Thread während seiner Laufzeit annehmen?

2. Koordinierung

Skizzieren Sie ein Programm-Modul (Klasse), das einen Ringpuffer implementiert, mit den Prozeduren (Methoden): **einfügenElement** und **entnehmenElement**. Die Prozeduren sollen von mehreren Threads parallel aufrufbar sein, d.h. Sie müssen insbeondere auf die Koordinierung achten. Als Element können Sie einen beliebigen Datentyp wählen.

1.1 Skizzieren Sie in einer Ihnen geläufigen Programmiersprache (bzw. in einer Programmiersprachen ähnlichen Notation) die verwendeten Datenstrukturen.
1.2 Skizzieren Sie in ähnlicher Notation („prozedural“) die beiden Prozeduren (Methoden) zum Einfügen und Entfernen eines Elements. Der Schwerpunkt soll auch hierbei auf den notwedigen Koordinierungsmaßnahmen liegen.
1.3 Beschreiben Sie die Funktionsweise des von Ihnen verwendeten Koordinierungsmechanismus’.

2.22 Frühjahr 2001 II

Aufgabe 1 Addierwerke und Multiplexer

Ein Halbaddierer bildet für Bit-Paare die Summe mod2 und den Übertrag.

1. Man synthetisiere aus zwei Halbaddierern das Schaltnetz für die Binäraddition (Volladdierer).
2. Man synthetisiere Halb- und Volladdierer ausschließlich aus NAND-Gattern.

Man zeige, wie sich mit Hilfe des Volladdierers

3. ein serielles 4-Bit Addierwerk,
4. ein paralleles 4-Bit Addierwerk aufbauen lässt.
5. Man zeichne das Schaltnetz für einen Multiplexer mit 8 Eingängen ein.

Aufgabe 2 Pipelining

Die Abarbeitung einer Maschineninstruktion erfordere neben dem Zugriff auf die Arbeitsregister, auf den Operanden- und den Instruktionscache die Befehlscodierung und –ausführung.

1. Man zeichne das entsprechende Phasendiagramm.
2. Man skizziere eine Befehlspipeline für dieses Phasendiagramm unter Berücksichtigung von „Internal Forwarding“.
3. Man nenne und beschreibe vier Arten von Pipeline-Konflikten, die den Fluss der Befehlsabarbeitung in einer Pipeline hemmen können.
4. In einer fünf-stufigen Befehlspipeline werden nach jedem bedingten Springbefehl drei „Stall“-Zyklen eingeschoben.
   Was sollen diese Zyklen bewirken? Man zeichne die Phasendiagramme für die Abarbeitung von vier Maschinenbefehlen, von denen der erste ein bedingter Sprung ist, mit und ohne Stallzyklen. Wie sehr vermindern die Stallzyklen die Leistung der Pipeline, wenn 20% der Instruktionen bedingte Sprünge sind?
5. Man bestimme die Beschleunigung (Speed-Up) einer k-stufigen Pipeline für die Abarbeitung von n Instruktionen, wenn keine Hemmnisse auftreten.
   Wie groß ist die maximale Beschleunigung?

Aufgabe 3 Übertragungstheorie

1. Erstellen Sie ein Modell einer Übertragungsstrecke.
2. Welche Rolle erfüllt hierbei die

2.1 Quellcodierung?
2.2 Kanalcodierung?
2.3 Leitungscodierung?

Aufgabe 4 Kanalcodierung

1. Geben Sie die Definition von *Hamming-Distanz* an.
2. Wie groß ist die Hamming-Distanz bei einem Code mit einfacher Paritätsprüfung?
3. Welche Hamming-Distanz hat ein zyklischer Hamming-Code und welche Eigenschaften lassen sich daraus ableiten?
4. Ein zyklischer Hamming-Code wird durch das Generatorpolynom $G(u) = u^3 + u + 1$ festgelegt.

4.1 Welche Eigenschaften besitzt das Generatorpolynom G(u)?
4.2 Wie ist die Anzahl von Nachrichten- und Kontrollstellen für den durch G(u) erzeugten Hamming-Code?

4.3 Geben Sie durch G(u) erzeugte Prüfmatrix an.

Aufgabe 5 Quellcodierung

1. Definieren Sie die folgenden Begriffe:
1.1 Informationsgehalt
1.2 Entscheidungsgehalt
1.3 Redundanz

2. Codieren Sie die Zeichen der folgenden Quelle nach dem Verfahren von Huffman und berechnen Sie die Redundanz.

x_i	x_1	x_2	x_3	x_4	x_5
$p(x_i)$	0,3	0,3	0,2	0,15	0,05

3. Ist es immer möglich eine Quelle redundanzfrei zu codieren?

2.23 Herbst 2001 I

Aufgabe 1
ER-Diagramm, Integritätsbedingungen, Schemaentwurf und SQL-Anfragen

Es soll eine Datenbank für ein Kino- und Film-Auskunftssystem für eine Stadt entworfen werden. Das System soll vergangene und zukünftige Spielpläne enthalten können.

Entity-Mengen:

Regisseure/Innen	mit den Attributen NAME, VORNAME, GEB-DATUM, VITA
Filme	mit dem Attribut TITEL
Schauspieler	mit denselben Attributen wie Regisseure/Innen aber zusätzlich mit dem künstlichen Schlüssel S# vom Typ **integer**
Kinos	mit den Attributen BEZEICHNUNG, STRASSE, HAUSNR, TELEFON-NR

Relationships:

spielt	Ein Schauspieler spielt in einem Film.
führt	Ein Regisseur führt Regie in einem Film.
läuft	Ein Film läuft in einem Kino.

Integitätsbedingungen:

Neben den offensichtlichen Integritätsbedingungen sollen folgende gelten:

I1	In einem Film führt nur 1 Person Regie.
I2	In einem Kino können mehrere Filme laufen, aber nur zu verschiedenen Zeiten (es gibt nur 1 Vorführraum).
I3	TITEL, BEZEICHNUNG sowie die Kombination NAME, VORNAME sind eindeutig für Filme, Kinos bzw. Regisseure. Für Schauspieler sei sie Kombination NAME, VORNAME, GEB-DATUM eindeutig.

1. E/R Diagramm
1.1 Entwerfen Sie für die Datenbank ein E/R Diagramm entsprechend den obigen Spezifikationen und Integritätsbedingungen.

1.2 Geben Sie die Kardinalitäten für die Relationships an.
1.3 Geben Sie für jede Entität die Mengen der Schlüsselkandidaten an.
1.4 Geben Sie die Attribute der Relationships an.
1.5 Geben Sie mindestens zwei verschiedene Varianten für die Relationship **läuft** an, treffen Sie eine Modellierungsentscheidung und begründen Sie Ihre Entscheidung.

2. Relationales Schema
   Geben Sie zu dem entwickelten E/R Diagramm ein relationales Schema an und kennzeichnen Sie durch Unterstreichen die gewählten Primärschlüssel.

3. SQL-Anfragen
   Formulieren Sie für das relationale Schema die folgenden Anfragen bzw. Operationen in SQL:

3.1 Eine Liste aller Filmregisseure
3.2 In welchem Film spielt Meryll Streep?
3.3 NAME, GEB-DATUM und VITA des Regisseurs von „African Queen“
3.4 In welchem Film spielt Meryll Streep gemeinsam mit Robert Redford?
3.5 In welchem Kino mit Tel.Nr. läuft heute der Film „The Straight Story“ und zu welcher Zeit?
3.6 Änderung des Spielplans des Kinos „Media Palast“, so dass morgen um 22:15 Uhr der Film „The Straight Story“ läuft

Aufgabe 2 Betriebsmittelverwaltung und Deadlocks

Eine der wesentlichen Aufgaben eines Betriebssystems ist es, die vorhandenen Hardware- und Software-Betriebsmittel zu verwalten und für einenverklemmungsfreien Ablauf der einzelnen Prozesse zu sorgen.

1. Betriebsmitteleinteilung
   In welche Klassen können Betriebsmittel eingeteilt werden? Geben Sie für jede der genannten Betriebsmittelklassen ein Beispiel an.

2. Bedingungen für Deadlocks
   Bei der Zuteilung von Betriebsmitteln an Prozesse sollte das Auftreten von Deadlocks ausgeschlossen werden. Welche vier Bedingungen sind Voraussetzung für einen Deadlock? Bei welchen der oben genannten Betriebsmittelklassen können diese Bedingungen eintreten?

3. Deadlock-Verhinderung und Deadlock-Vermeidung
   Erklären Sie den Unterschied zwischen Deadlock-Verhinderung und Deadlock-Vermeidung, und nennen Sie jeweils ein Ihnen bekanntes Verfahren.

Aufgabe 3 Prozesssynchronisation

Gegeben sei ein Keller, in den Elemente der Klasse ***Element*** abgelegt werden können. Auf dem Keller seien zwei Methoden definiert: ***füge_ein***, mit der ein Element der Klasse ***Element*** in den Keller eingefügt werden kann und ***entnimm***, mit der ein Element aus dem Keller entnommen werden kann. Es können maximal ***max*** Elemente im Keller abgelegt werden.

Für die Benutzung des Kellers seien folgende Synchronisationsbedingungen gegeben:
- Die Methoden ***füge_ein*** und ***entnimm*** sind wechselseitig ausgeschlossen auszuführen.

- Die Methode ***füge_ein*** darf nur ausgeführt werden, wenn der Keller nicht voll ist, d.h. wenn die Anzahl der Elemente im Keller kleiner ***max*** ist.
- Die Methode ***entnimm*** darf nur ausgeführt werden, wenn der Keller nicht leer ist, d.h. wenn die Anzahl der Elemente im Keller größer 0 ist.

1. Semaphore
   Implementieren Sie die Klasse Keller so, dass die oben genannten Synchronisationsbedingungen durchgesetzt werden. Verwenden Sie zur Durchsetzung der Synchronisationsbedingungen ausschließlich Semaphore (gegeben durch die Klasse Semaphor). Achten Sie dabei auf die korrekte Initialisierung der verwendeten Semaphore.

   Sie dürfen für Ihre Lösung folgende zwei Klassen als gegeben voraussetzen:

```
public class Semaphor
{
     // Konstruktor
     public Semaphor (int init) {…}

     // Methoden
     public void prolog() {…}
     public void epilog() {…}
}

public class Element
{
     // Konstruktor
     public Element(...) {...}

     // Methoden
     ...
}
```

2. Eigenschaften eines Semaphors
   Nennen Sie die Eigenschaften eines Semaphors und erklären Sie seine Funktionsweise.

3. Prozesszustandsgraph
   Prozesse, die einen wie oben beschriebenen Puffer nutzen, können im Verlauf ihrer Lebenszeit unterschiedliche Zustände annehmen. Zeichnen Sie einen allgemeinen Prozesszustandsgraphen und markieren Sie die möglichen Übergänge aus Teilaufgabe 1 in diesem Graph.

Aufgabe 4 Hauptspeicher und Festplatte

1. First Fit und Best Fit
   Beschreiben Sie die beiden Speicherverwaltungsstrategien First Fit und Best Fit und nennen Sie jeweils deren Vor- und Nachteile.

2. SSF und SCAN
   Beschreiben und bewerten Sie die beiden Plattenzugriffsstrategien SSF (shortest seek time first) und SCAN (Aufzugsstrategie). Welches Gütekriterium für Festplatten wird heutzutage meist angegeben und wie beurteilen Sie es?

Aufgabe 5

1. Pipelining Prinzip
   Fließband-Ausführung (pipelining) ist ein hauptsächlich in Prozessoren und/oder in ihren Teilwerken angewandtes Ausführungsprinzip von Befehlen (Instruktionen).

1.1 Welches Ziel soll mit der Anwendung dieses Prinzips erreicht werden?

1.2 Was können Sie bzgl. der Ausführungszeit der einzelnen Befehle sagen?

1.3 Angenommen sei eine lineare Pipeline mit k Stufen und identischer Ausführungszeit von T_s Zeiteinheiten (ZE, meist in Taktzeiten angegeben) pro Stufe:
Wie lang ist die (maximale) Ausführungszeit A jedes einzelnen Befehls?

1.4 Zusätzlich angenommen sei die Ausführung von n Befehlen: Gesucht ist eine Formel für die mittlere Ausführungszeit A(k,n) eines Befehls bei idealer Ausführung der n Befehle (d.h. bei Abwesenheit jeglichen Konflikts wie Ressourcenkonflikt, Datenabhängigkeit, Verzweigungen, Unterbrechungen, u.a.). Was ist die größte untere Schranke für die mittlere Ausführungszeit A(k)?

2. Sprünge in Pipelines
   Gegeben sei folgende Pipeline für die Befehlsausführung mit den 5 Stufen S1, S2, S3, S4, S5:

   S1: Befehl holen (BH)
   S2: Befehl dekodieren (BD)
   S3: Operanden holen (OH)
   S4: Befehl ausführen (BA)
   S5: Ergebnis speichern (ES)

   Es gelte die Ausführungsreihenfolge: $\rightarrow$ BH $\rightarrow$ BD $\rightarrow$ OH $\rightarrow$ BA $\rightarrow$ ES $\rightarrow$
   Welche Auswirkung gegenüber dem idealen Ablauf der Befehlsausführung in der Pipeline (ohne organisatorische Gegenmaßnahmen) haben Programmstrukturen wie
   - unbedingte Sprünge (jump address)
   - bedingte Verzweigungen (conditional jump: if (condition) then ...)
   - Schleifen (loops) ?

2.24 Herbst 2001 II

Aufgabe 1

1. Erklären Sie die Begriffe Speicher- und Leitungsvermittlung. Stellen Sie die Vor- und Nachteile der beiden Vermittlungsarten gegenüber.
2. Was heißt ATM? Beschreiben Sie kurz die Grundidee von ATM und die erwarteten Vorteile gegenüber klssischer Verfahren.
3. Welche Art der Vermittlung benutzt ATM? Begründen Sie Ihre Antwort.

Aufgabe 2

1. Normalformen

   Für die Anwendung in der Anlageabteilung einer Bank sei ein relationales Datenbankschema gegeben. Die verwendeten Attribute sind:
   Anlageberater (AB), Büro des Beraters (Büro), Anleger, Aktienname (Aktie), Anzahl der gekauften Aktien und die für eine bestimmte Aktie gezahlte Dividende.

Die Relationenschemata sehen wie folgt aus:

ANLAGE (Aktie, Anzahl, Anleger, Dividende)
mit den funktionalen Abhängigkeiten
AB → Dividende sowie Anleger, Aktie → Anzahl

BERATUNG (Anleger, AB, Büro)
mit den funktionalen Abhängigkeiten AB → Büro sowie Anleger → AB

1.1 Welchen Normalformen (2.NF, 3.NF, BCNF) genügen die Relationenschemata und welchen nicht? Begründen Sie kurz Ihre Antworten.
1.2 Können beim Arbeiten mit dieser Datenbank Anomalien auftreten und, wenn ja, welche (jeweils mit einem kurzen Beispiel)?
1.3 Bringen Sie die Relationen in BCNF und begründen Sie Ihre Transformationen.

2. SQL

Ein Flugbuchungssystem enthalte unter anderem folgende Relationen:

- Flughafen (FlughafenNummer, FlughafenName, FlughafenOrt, Land)
- Flug (FlugNummer, FluggesellschaftKürzel, FlugzeugNummer)
- Flugintervall (FlugintervallNummer, FlugNummer , Startflughafen Nummer , Zielflughafen Nummer , Abflugzeit, Ankunftszeit)
- Buchung (PassagierNummer , Fluginterv allNummer , BuchungsklasseNummer)
- Buchungsklasse (BuchungsklasseNummer, BuchungsklasseBeschreibung)
- Flugpreis (Fluginterv allNummer , BuchungsklasseNummer , Preis)

2.1 Formulieren Sie eine Anfrage, die für die vorhandenen Flugintervalle eine Tabelle der Namenspaare der Start- und Ziel-Flughäfen nach Start-Flughäfen sortiert liefert.
2.2 Ermitteln Sie die Flugintervallnummern, die zwei Orte (!) ‚Start' und ‚Ziel' direkt verbinden, zusammen mit den zugehörigen Abflugs- und Ankunftszeiten. ‚Start' und ‚Ziel' sollen Parameter sein.
2.3 Ermitteln Sie zu einem als Parameter angebbaren Flugintervall, wie viele Plätze in den verschiedenen Buchungsklassen belegt sind. (Buchungsklassen ohne Buchungen brauchen nicht aufgeführt zu werden.)
2.4 Ermitteln Sie zu einem angebbaren Flug den Gesamtpreis der getätigten Buchungen.

Aufgabe 3

Sie wollen mit Ihrem Computer eine (Modell)Eisenbahn steuern. Dazu besitzt die Eisenbahn eine (mäßig) intelligente Steuereinheit. Diese Steuereinheit sei an eine der (seriellen oder parallelen) Schnittstellen des Rechners angeschlossen. Die Steuereinheit selbst verhält sich passiv, d.h. alle Aktivitäten müssen vom Rechner ausgelöst werden.

Um die Eisenbahn zu steuern, muss also der Rechner ein „Schreib-Kommando“ über die Schnittstelle an die Steuereinheit senden, das etwa folgende Form haben könnte:
Write <adresse> <wert> (mit Adresse wird eine bestimmte Weiche oder Lokomotive angesprochen und auf den mitgesendeten Wert gesetzt).
Zum Abfragen eines Zustandes (z.B. eine bestimmte Weichenstellung) muss der Rechner zunächst ein „Lese-Kommando“ der Form: „Read <adresse>“ an die Steuerung senden und dann an der Schnittstelle auf den als Antwort gesendeten Wert warten.

1. Das Betriebsssystem Ihres Rechners stelle Ihnen primitive Kommandos (Systemaufrufe) zur Verfügung, mit denen Sie auf die Schnittstelle schreiben und aus ihr lesen können.
   Skizzieren Sie in einer programmiersprachenähnlichen Notation ein Modul (Klasse), das die zum Betreiben der Eisenbahn notwendigen Prozeduren (Methoden) Open, Close und Lesen und Schreiben von „Variablen“ (Adressen) der Eisenbahnsteuereinheit realisiert, etwa in der Form:
- **open** („eventuell Angabe der Schnittstelle“)
- **close ( )**
- **write (adresse, wert)**
- **wert = read (adresse)**

2. Berücksichtigen Sie, dass die entworfenen Prozeduren von mehreren Prozessen (zumindest von mehreren Threads) gleichzeitig aufgerufen werden können. Erläutern Sie anhand einer read-Operation, welche Probleme dabei auftreten können, und geben Sie eine Lösung für dieses Problem an.

3. Wenn ein Zug fährt, werden beim Überfahren bestimmter Gleisabschnitte (Meldegleise) Kontakte ausgelöst und von der Eisenbahn-Steuereinheit erkannt. Was können (müssen) Sie tun, um die Bewegung eines Zuges zu verfolgen?
   Skizzieren Sie ein Programmsystem, das anzeigt, welche Gleisabschnitte gerade belegt sind, und mit dem man gleichzeitig auch andere Funktionen der Eisenbahn steuern kann.