Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer

Kennzahl: _____

Kennwort: _____

Arbeitsplatz-Nr.: _____

Herbst 2011

66114

Erste Staatsprüfung für ein Lehramt an öffentlichen Schulen

— Prüfungsaufgaben —

Fach: **Informatik (vertieft studiert)**

Einzelprüfung: **Datenbank- und Betriebssysteme**

Anzahl der gestellten Themen (Aufgaben): **4 Aufgaben, von denen zwei gemäß untenstehender Auswahlregel zu bearbeiten sind!**

Anzahl der Druckseiten dieser Vorlage: **13**

Zu den zwei Themenschwerpunkten A (Datenbanksysteme) und B (Betriebssysteme) ist jeweils entweder die Aufgabe 1 oder 2 zu wählen. Auf der Vorderseite des Kopfbogens sind im Feld „gewähltes Thema: Nr." die Nummern der beiden gewählten Aufgaben anzugeben (z. B. A2, B1)!

Bitte wenden!

Themenschwerpunkt A

Aufgabe A1

Datenbanksysteme

1. SQL

Gegeben sei folgendes Schema, welches Zoos, Tiere, die in ihnen eine Zeitlang leben, und die Tierarten der Tiere speichert:

zoos:	{[*zooid, name, stadt*]}	*Zoos*
tiere:	{[*tierid, name, geburtstag, todestag*]}	*Tiere*
lebt_in:	{[*zooid, tierid, von, bis*]}	*Aufenthalte der Tiere in verschiedenen Zoos*
tierarten:	{[*tierartid, name*]}	*Tierarten*
hat_tierart:	{[*tierid, tierartid*]}	*Zuordnung von Tierarten zu Tieren*
tierartenhierarchie	{[*oberartid, unterartid*]}	*Zuordnung von direkten Untertierarten zu Tierarten*

Beachten Sie bei der Bearbeitung dieser Aufgabe folgende Hinweise:

- *Bei allen lebenden Tieren steht im Todestag der Wert NULL.*
- *Ein Tier kann nicht in mehreren Zoos gleichzeitig leben.*
- *Es kann sein, dass ein Tier eine Zeit seines Lebens in keinem Zoo lebt. Dies ist zum Beispiel der Fall, wenn es in Freiheit geboren wurde.*
- *Das Attribut* bis *der Relation* lebt_in *kann den Wert* NULL *besitzen, wenn das Tier mit dem Schlüssel* tierid *zurzeit im Zoo mit dem Schlüssel* zooid *lebt. Es kann aber auch sein, dass bereits ein Datum festgelegt wurde, an dem das Tier den Zoo verlassen wird. Eventuell ist sogar der Verbleib des Tieres in verschiedenen Zoos auf Jahre hinaus in die Zukunft geplant.*
- *Eine Tierart kann eine beliebige Anzahl Untertierarten haben.*
- *Eine Tierart kann eine beliebige Anzahl Obertierarten haben.*

a) Formulieren Sie eine SQL-Anfrage, die *tierid* und *name* aller Tiere zurückgibt, die eine Zeit im Zoo mit der *zooid* 42 gelebt haben, zurzeit in diesem Zoo leben oder voraussichtlich in Zukunft in diesem Zoo leben werden. Stellen Sie sicher, dass Ihre Anfrage keine Duplikate erhält. Sortieren Sie das Ergebnis nach den Namen der Tiere.

b) Formulieren Sie eine SQL-Anfrage, die für jeden Zoo *zooid*, *name* und die Anzahl der Tiere, die zurzeit in diesem Zoo leben, zurückliefert! Zoos, in denen zurzeit keine Tiere leben, sollen nicht im Ergebnis auftauchen. Sortieren Sie das Ergebnis nach der Anzahl der Tiere, die zurzeit im Zoo leben.
Hinweis: CURRENT_DATE liefert in SQL das heutige Datum.

Fortsetzung nächste Seite!

c) Formulieren Sie eine SQL-Anfrage, die *tierid* und *name* aller Tierarten zurückgibt, die einen Umzug von Berlin nach München erlebt haben oder voraussichtlich erleben werden.

Hinweis:
- *Mithilfe des Minus-Operators kann in SQL die Differenz in Tagen zwischen zwei Daten berechnet werden.*
  Der Ausdruck DATE('2012-02-02') – DATE('2011-02-02') liefert beispielsweise das Ergebnis 365.
- *Der Umzug eines Tieres von einem Zoo in einen anderen lässt sich dadurch erkennen, dass das Tier an einem Tag im Ausgangszoo und am darauf folgenden Tag im Zielzoo lebt.*

d) Formulieren Sie eine SQL-Anfrage, die angibt, wie viele Schildkröten am 01.01.1900 in Münchner Zoos gelebt haben.

Hinweis:
- *Sie dürfen Views verwenden, um Teilergebnisse auszudrücken!*
- *DATE('1900-01-01') liefert das Datum 01.01.1900 in SQL.*
- *Es kann sein, dass Schildkröten von damals heute noch leben.*
- *Auch Unterarten von Schildkröten sind Schildkröten (z. B. Meeresschildkröten). Dasselbe gilt für die Unterarten der Unterarten von Schildkröten (z. B. Bastardschildkröten) und so weiter (z. B. Olivbastardschildkröte, Atlantikbastardschildkröte). Folglich ist zur Beantwortung dieser Anfrage eine rekursive (temporäre) View bzw. Common Table Expression zwingend erforderlich.*

2. Modellierung

Für eine Autovermietung wird ein geeignetes ER-Modell zur Verwaltung von Mietwagenreservierungen benötigt.

- Jedes Fahrzeug besitzt eine eindeutige Fahrzeugidentifikationsnummer sowie eine Farbe. Zusätzlich wird für jedes Fahrzeug gespeichert, ob dieses mit einem Navigationsgerät und/oder einer Klimaanlage ausgestattet ist.
- Jedes Fahrzeug gehört genau einem Fahrzeugtyp an. Ein Fahrzeugtyp wird durch die Kombination einer Typschlüsselnummer (TSN) und Herstellerschlüsselnummer (HSN) identifiziert. Zu einem Fahrzeugtyp sind dessen Modellname (z. B. Golf) und dessen Motorbezeichnung (z. B. 2.0 TDI) zu speichern.
- Bei einem Fahrzeugtyp kann es sich um eine Limousine, einen Kombi oder einen Van handeln. Für Vans ist zusätzlich die Anzahl der Sitze zu speichern, für Kombis wird das Kofferraumvolumen und für Limousinen die Anzahl der Türen angegeben. Ein Fahrzeugtyp kann höchstens zu einer dieser Karosserieformen zählen. Es kann jedoch Fahrzeugtypen geben, die keiner der drei genannten Formen entsprechen.

Fortsetzung nächste Seite!

- Ein Fahrzeugtyp gehört mindestens einer Fahrzeugklasse an, die eindeutig durch ihre Klassenbezeichnung (bspw. Mittelklasse, Kompaktklasse, etc.) identifiziert wird. Es kann Fahrzeugtypen geben, die sich nicht eindeutig einer Klasse zuordnen lassen, und daher mehreren Klassen angehören.
- Von Kunden der Autovermietung wird deren Vorname, Nachname sowie deren Geburtsdatum und Anschrift gespeichert. Außerdem erhält jeder Kunde initial eine eindeutige Kundennummer.
- Ein Kunde kann eine Reservierung aufgeben. Dabei kann der Kunde zwischen zwei Arten von Reservierungen wählen. Nutzt er eine Spezialreservierung, so kann er den gewünschten Fahrzeugtyp wählen (z. B. Golf mit 2.0 TSI). Wählt er dagegen die Standardreservierung, so kann er sich lediglich auf eine Fahrzeugklasse (z. B. Kompaktklasse) festlegen. In beiden Fällen gibt eine Reservierung zudem den Zeitraum an, in dem das Auto gemietet werden soll. Eine Reservierung besitzt eine Reservierungsnummer, die jedoch nur bezüglich des jeweiligen reservierenden Kunden eindeutig ist.

a) Entwerfen Sie ein ER-Modell, das die oben beschriebenen Punkte möglichst präzise modelliert.
Entwickeln Sie dazu geeignete Entity-Typen mit entsprechenden Attributen sowie Relationship-Typen, um den logischen Zusammenhang zwischen diesen aufzuzeigen.
Machen Sie Gebrauch von Generalisierungen/Spezialisierung, sofern sinnvoll.
Kennzeichnen Sie die Primärschlüssel im Modell.
Fügen Sie zudem Funktionalitäten in das ER-Diagramm ein.

b) Überführen Sie das in Teilaufgabe a) konstruierte ER-Diagramm in ein verfeinertes Relationsmodell. Geben Sie hierfür die verallgemeinerten Relationenschemata an. Vermeiden Sie dabei redundante Attribute.

3. Normalformen

Gegeben sei folgende Relation

R: {ID, Vorname, Nachname, Geburtsdatum,
Breitengrad, Längengrad, Länge, Breite, Fläche},

die ausgewiesenen Parzellen des Mondes ihre jeweiligen Eigentümer zuweist.

- Ein Eigentümer einer Mondparzelle erhält mit dem erstmaligen Erwerb einer Parzelle eine eindeutige *ID* zugewiesen. Zur Vereinfachung wird hier zu einem Eigentümer lediglich dessen *Vorname, Nachname* und *Geburtsdatum* verwaltet.
- Parzellen sind stets von rechteckiger Gestalt und paralleler Ausrichtung zum Mondäquator und können damit stets eindeutig durch deren südöstlichste Koordinate identifiziert werden. Diese wird in Form des Paares *(Breitengrad, Längengrad)* modelliert.
- Eine jede Parzelle besitzt eine *Länge* (in Ost-West-Richtung) sowie eine *Breite* (in Nord-Süd-Richtung). Zudem wird für eine Parzelle deren *Fläche* gespeichert, die aus der *Länge* und *Breite* der Parzelle resultiert.
- Hier wird davon ausgegangen, dass eine Parzelle stets nur maximal einen Eigentümer haben kann.

Fortsetzung nächste Seite!

a) Geben Sie für das beschriebene Szenario eine Menge funktionaler Abhängigkeiten F an, aus der sämtliche funktionalen Abhängigkeiten der Relation R abgeleitet werden können.

b) Bestimmen Sie eine kanonische Überdeckung F_C für F.

c) Bestimmen Sie anhand der in Teilaufgabe b) entwickelten kanonischen Überdeckung F_C den oder die Schlüsselkandidaten der Relation R.

d) Erfüllt R die erste Normalform? Welches Kriterium muss dafür erfüllt sein?

e) Ist R in 2NF? Begründen Sie Ihre Entscheidung.

f) R erfüllt nicht die dritte Normalform. Wenden Sie daher den Synthesealgorithmus auf R an, um ein neues Relationenschema zu erhalten, das die dritte Normalform erfüllt.

Aufgabe A2

1. Datenmodellierung: ER-Modell, SQL

In einer *Brokering-Datenbank* sollen folgende Informationen gespeichert werden:

- Es gibt Büros an verschiedenen Standorten und jedes Büro hat eine Adresse (Schlüssel) sowie eine Telefonnummer.
- Jeder Broker arbeitet in genau einem Büro; es wird auch gespeichert, seit wann er dort arbeitet. Ein Broker hat verschiedene Telefonnummern, und er wird durch eine Broker-ID (Schlüssel) identifiziert. Jedes Büro wird von einem Broker geleitet; es wird auch gespeichert, seit wann dieser das Büro leitet.
- Ein Konto hat eine Kontonummer (Schlüssel), und es wird der Eröffnungstag und der Status des Kontos gespeichert. Ein Konto wird von einem Broker betreut.
- Ein Kunde wird durch eine Kunden-ID (Schlüssel) identifiziert, und es werden außerdem der Name und das Geburtsdatum des Kunden gespeichert. Ein Kunde kann verschiedene Konten haben, die von den Büros betreut werden.

Wenn das Konto eines Kunden von einem Büro betreut wird, dann soll der das Konto betreuende Broker in diesem Büro arbeiten. Diese Integritätsbedingung I kann im ER-Modell nicht abgebildet werden, sie kann aber im zugehörigen relationalen Modell überprüft und berücksichtigt werden.

a) Erstellen Sie ein ER-Modell, welches das oben dargestellte Szenario möglichst vollständig abbildet. Verwenden Sie wann immer möglich (binäre oder auch höherstellige) Relationships.

b) Übertragen Sie Ihr ER-Modell ins relationale Modell. Erstellen Sie dazu Tabellen mit Hilfe von CREATE TABLE-STATEMENTS in SQL. Berücksichtigen Sie die Fremdschlüsselbeziehungen.

c) Schreiben Sie ein SELECT-Statement, das die obige Integritätsbedingung I überprüft.

d) Geben Sie geeignete INSERT-Statements an, die in alle beteiligten Tabellen jeweils mindestens ein Tupel einfügt, so dass alle Integritätsbedingungen erfüllt sind, nachdem alle Einfügungen ausgeführt werden.

Fortsetzung nächste Seite!

2. Datenbankfragen in SQL

Gegeben seien die folgenden Tabellen aus einer *Fußball-Datenbank*:

- *Spieler (Verein, Rückennummer, Name, Alter, Position),*
- *Verein (Name, Stadt),*
- *Spiel (ID, Heim, Gast, Datum, Tore_Heim, Tore_Gast):*
  Für jedes Spiel werden der Heim- und der Gastverein sowie die von diesen jeweils erzielte Anzahl von Toren gespeichert.
- *Tor (SpielID, Spielzeit, Verein, Rückennummer):*
  Für jedes Tor werden die Spielzeit (eindeutiger Zeitstempel in Form einer reellen Zahl zwischen 0 und 90 zur Repräsentation der vergangenen Minuten) sowie der Spieler (gegeben durch Verein und Rückennummer) gespeichert; wir nehmen an, dass beide Halbzeiten genau 45 Minuten dauern.

Formulieren Sie auf den obigen Tabellen folgende SQL-Anfragen:

a) Bestimmen Sie alle Vereine aus *München.*

b) Bestimmen Sie alle Heimspiele, die der Verein *Bayern München* gewonnen hat.

c) Bestimmen Sie alle Tore, die in der ersten Halbzeit gefallen sind.

d) Bestimmen Sie für jeden Spieler die Anzahl seiner erzielten Tore.

e) Bestimmen Sie, nach Vereinen und Positionen gruppiert, die Anzahl der Spieler auf diesen Positionen.

f) Erstellen Sie mittels eines CREATE VIEW-Statements eine Sicht, die für jedes Spiel und jeden beteiligten Verein die Anzahl der erzielten Punkte enthält.

g) Füllen Sie die Sicht, indem Sie das Ergebnis geeigneter SELECT-Statements einfügen. Ein Sieg soll dabei 3 Punkte bringen, ein Unentschieden 1 Punkt.

h) Bestimmen Sie auf der Basis der Sicht für jeden Verein die Anzahl seiner erzielten Punkte.

3. Funktionale Abhängigkeiten und Zerlegungen

Gegeben sei das Relationenschema $R = (U, F)$ mit der Attributmenge

$$U = \{ A, B, C, D, E \}$$

und der folgenden Menge F von funktionalen Abhängigkeiten:

$$F = \{ AD \rightarrow BC,\ C \rightarrow AD,\ CD \rightarrow E \}$$

a) Geben Sie alle Schlüssel für R an – jeweils mit Begründung.

b) Ist R in 2NF, 3NF bzw. in BCNF? Geben Sie jeweils eine Begründung an.

c) Zerlegen Sie R mittels des Synthesealgorithmus in ein 3NF-Datenbankschema.

d) Zerlegen Sie R bezüglich der funktionalen Abhängigkeit $C \rightarrow AD$ verlustfrei in zwei Relationenschemata $R_i = (U_i, F_i)$, $i = 1, 2$.

Themenschwerpunkt B

Aufgabe B1

Betriebssysteme

1. Semaphore

In ein automatisches Flugbuchungssystem soll eine Gefährdungsanalyse integriert werden. Ein vom Nutzer eingegebener Datensatz mit Buchungsdaten wird hierfür einer automatisierten Gefährdungsanalyse (G) unterzogen, die entweder „bedenklich" oder „unbedenklich" als Antwort zurückliefert. Wird ein Datensatz als „unbedenklich" eingestuft, so wird die Buchung durch das elektronische Buchungssystem (B) durchgeführt. Liefert die Gefährdungsanalyse „bedenklich" als Antwort, so wird der Datensatz durch einen Sicherheitsbeauftragen (S) manuell überprüft; dieser kann je nach Ergebnis die Buchung entweder freischalten (dann wird sie durch dasselbe elektronische Buchungssystem durchgeführt) oder verweigern.

Für die Gefährdungsanalyse steht ein Rechenzentrum mit 50 Servern zur Verfügung. Diese Analyse ist somit für bis zu 50 Datensätze gleichzeitig möglich. Das elektronische Buchungssystem kann bis zu 100 Buchungen gleichzeitig verarbeiten und es steht genau ein Sicherheitsbeauftragter zur Verfügung.

Modellieren Sie den Buchungsprozess mit integrierter Gefährdungsanalyse unter Verwendung von Zähl-Semaphoren (Counting Semaphores), so dass möglichst viele Berechnungen parallel durchgeführt werden können:

a) Definieren Sie in einer objektorientierten Sprache Ihrer Wahl oder in Pseudocode die Schnittstellen (Konstruktor, nach außen sichtbare Methoden) einer Zähl-Semaphore SEM. Eine Implementierung der Methoden ist nicht notwendig.

b) Benutzen Sie Ihr Ergebnis aus dem Aufgabenteil a), um den folgenden Code zu vervollständigen. Ersetzen Sie dabei die Kommentare der Form `/ * Ergaenzung X */` durch geeignete Semaphor-Aufrufe. Einem Kommentar können dabei einer, mehrere oder keine Aufrufe entsprechen.

```
class Buchungssytem {
     Sem G = new SEM /* Ergaenzung 1 */
     Sem B = new SEM /* Ergaenzung 2 */
     Sem S = new SEM /* Ergaenzung 3 */

     public void flugbuchung(Datensatz d) {
          /* Ergaenzung 4 */
          if (gefaehrdungsanalyse(d) == unbedenklich) {
               /* Ergaenzung 5 */
               buchung(d);
          } else { /* if (gefaehrdungsanalyse(d) == bedenklich) */
               /* Ergaenzung 7 */
               if (freigeschaltet(d)) {
                  /* Freischaltung durch Sicherheitsbeauftragten */
                  /* Ergaenzung 8 */
                  buchung(d);
                  /* Ergaenzung 9 */
               }
               /* Ergaenzung 10 */
          }
          /* Ergaenzung 11 */
     }
```

Fortsetzung nächste Seite!

2. Virtueller Speicher

In dieser Aufgabe geht es um eine optimale Wahl der Seitengröße p eines Systems mit Seitenadressierung. Nehmen Sie an, dass der gesamte Speicher des Rechners genau ein Programm enthält. Die durchschnittliche Größe eines Programms sei l Bytes.

a) Was ist die interne Fragmentierung?

b) Im Schnitt bleibe die Hälfte der Seiten unbenutzt. Wie groß ist die durchschnittliche interne Fragmentierung, wenn $p = 1024$ Bytes ist?

c) Welche Information über eine Seite muss in der Seitentabelle mindestens (unabhängig von der gewählten Verdrängungsstrategie) vorgehalten werden? Wovon hängt die Anzahl i der für einen solchen minimalen Seitentabelleneintrag benötigten Bytes ab?

d) Welche Gründe sprechen für die Wahl eines kleinen bzw. großen Wertes von p?

e) Geben Sie die durchschnittliche Größe der Seitentabelle als Funktion von l, i und p an.

f) Sowohl für die interne Fragmentierung als auch für die Seitentabelle wird Speicherplatz verbraucht, der für Nutzdaten nicht zur Verfügung steht. Bestimmen Sie für gegebene i und l den Wert von p, für welchen die Summe aus durchschnittlicher interner Fragmentierung und durchschnittlicher Größe der Seitentabelle minimal ist.

g) Sei $l = 128$ Kilobyte. Für welchen Wert von i ergibt Ihre Lösung aus dem vorherigen Punkt die optimale Seitengröße $p = 1024$ Byte?

3. Deadlocks, Bankier-Algorithmus

a) Nennen Sie die vier Eigenschaften, die ein Betriebssystem besitzen muss, so dass ein Deadlock auftreten kann (ohne Begründung).

b) Beim Bankieralgorithmus wurde der Begriff des „sicheren Zustands“ verwendet. Wie lautet die Definition eines sicheren Zustands?

c) Vier Prozesse (p_1, p_2, p_3, p_4) greifen auf fünf Ressourcenklassen (A, B, C, D, E) zu. Der Vektor verfügbarer Ressourcen ist $V = (6,9,9,7,9)$ und die Maximalanforderungsmatrix ist

$$M = \begin{pmatrix} 3 & 7 & 5 & 2 & 4 \\ 2 & 1 & 8 & 5 & 7 \\ 4 & 4 & 4 & 1 & 3 \\ 3 & 6 & 2 & 4 & 3 \end{pmatrix}$$

Geben Sie für die folgenden Zustände mittels des Bankiersalgorithmus an, ob sie sicher oder unsicher sind. Begründen Sie Ihre Aussage zwei Mal. Geben Sie für die nicht sicheren Zustände an, welche Prozesse an dem potentiellen Deadlock beteiligt sind.

i. Der Zustand Z_1 mit der folgenden Belegungsmatrix:

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

ii. Der Zustand Z_2, der sich aus Zustand Z_1 ergibt, wenn p_1 eine weitere Ressource aus der Klasse B zugeteilt bekommt.

iii. Der Zustand Z_3, der sich aus Zustand Z_1 ergibt, wenn p_4 eine weitere Ressource aus der Klasse B zugeteilt bekommt.

d) Führt jeder unsichere Zustand unweigerlich in einen Deadlock? Begründen Sie Ihre Antwort kurz.

Fortsetzung nächste Seite!

4. Prozess-Scheduling

In dieser Aufgabe geht es um präemptives und nicht-präemptives Scheduling von Prozessen. Gegeben seien vier Prozesse *A, B, C* und *D* mit den folgenden Auskunfts- und Laufzeiten:

Prozess	Auskunftszeit	Laufzeit
A	3	3
B	1	4
C	2	2
D	2	7

Wird ein Prozess unterbrochen, so reiht sich der Restprozess mit dem Zeitpunkt der Unterbrechung als Ankunftszeit in die Warteschlange ein. Kommen zwei Prozesse zeitgleich an, so wird nach Prozessnamen priorisiert (*A* vor *B* vor *C* vor *D*). Geben Sie für die folgenden drei Scheduling-Strategien an, zu welchen Zeiteinheiten welcher Prozess Rechenzeit zugeteilt bekommt. Geben Sie ferner für jeden Prozess *P* die Reaktionszeit r_P := erster Rechenzeitpunkt – Ankunftszeitpunkt sowie die Antwort a_P := letzter Rechenzeitpunkt – Ankunftszeitpunkt an (vergessen Sie dabei nicht, die letzte Zeiteinheit zu berücksichtigen):

a) Shortest Job First ohne Präemption

b) Shortest Job First mit Präemption

c) Round Robin mit Zeitscheibe 2

Aufgabe B2

1. Scheduling

a) Nennen Sie jeweils einen Vor- und einen Nachteil bei der Bevorzugung kurzer bzw. langer Prozesse beim Scheduling!
Worin unterscheiden sich harte und weiche Echtzeitanforderungen?

b) Gegeben sind folgende Prozesse mit ihren Bereitzeitpunkten und ihrem Rechenzeitbedarf. Führen Sie für eine Einprozessormaschine das nicht-preemptive Scheduling gemäß *Highest Response Ratio Next* (HRN) und das preemptive Scheduling gemäß *Round-Robin* (RR) mit einer Zeitscheibenlänge von 1 durch. Der Overhead für den Prozesswechsel ist hier vernachlässigbar.

Prozess	***Bereitzeitpunkt***	***Benötigte Rechenzeit***
A	0	5
B	0	3
C	1	2
D	4	1
E	7	2
F	14	1

Zuerst werden hinzustoßende Prozesse ans Ende der (eventuell leeren) Warteschlange angehängt; erst dann wird der nächste Prozess gemäß der Scheduling-Strategie bestimmt, welcher dem Prozessor zugeteilt wird. Ein Prozess kann also bereits zum Zeitpunkt seiner Ankunft im System ausgeführt werden (z. B. *A* zum Zeitpunkt *0*).

Fortsetzung nächste Seite!

Übertragen Sie die Tabellen auf Ihr Blatt. Tragen Sie in der Zeile Ausführung jenen Prozess ein, der für die entsprechende Zeitscheibe dem Prozessor zugeteilt wird. Die restlichen Zeilen der Tabelle stellen die Ready-Queue dar. Tragen Sie hier die Prozesse ein, die in der Warteschlange auf Zuteilung warten. Geben Sie jeweils beim Prozesswechsel von HRN auch alle $h_i(t)$ an.

Highest Response Ratio Next:

0					5					10					15				
Ausf.																			
Ready-Queue																			

Round Robin:

0					5					10					15				
Ausf.																			
Ready-Queue																			

Geben Sie für jeden Prozess die Verweildauer im System an:

Prozess	A	B	C	D	E	F
HRN						
RR						

c) Gegeben sind folgende periodische, unterbrechbare Prozesse A, B und C auf einer Einprozessormaschine. A benötigt alle 5 Zeiteinheiten 2 Zeiteinheiten Rechenzeit, B benötigt alle 4 Zeiteinheiten 2 Zeiteinheiten Rechenzeit und C benötigt alle 12 Zeiteinheiten 1 Zeiteinheit Rechenzeit. Berechnen Sie die Prozessorauslastung.

Führen Sie das Scheduling gemäß preemptiven Rate-Monotonic-Scheduling (RMS) mit Zeitscheibenlänge 1 durch und zeichnen Sie das entsprechende Gantt-Diagramm soweit sinnvoll!

Orientieren Sie sich an folgendem Muster:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
A																														
B																														
C																														

	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
A																														
B																														
C																														

Was stellen Sie fest?

Fortsetzung nächste Seite!

2. Synchronisation

a) Nennen Sie die vier hinreichenden und notwendigen Bedingungen für Verklemmungen und jeweils eine passende Maßnahme, um diese auszuschließen.

Gegeben sind die folgenden Programme in Pseudocode.

```
(1) Init () {
(2)       max=100;
(3)       size=0;
(4) }

(1) Erzeuger () {
(2)         while(true) {
(3)               Element=baue_neues_Element();
(4)               while(size>=max) {}
(5)               Puffer[size]=Element;
(6)               size=size+1;
(7)         }
(8) }

(1) Verbraucher () {
(2)         while(true) {
(3)               if(size>0) {
(4)                     Element=Puffer[size];
(5)                     size=size-1
(6)                     verarbeite_Element(Element);
(7)               }
(8)         }
(9) }
```

Im System sind *n* Erzeuger vorhanden und produzieren Elemente, die in eine Liste der Größe *max* eingefügt werden. Der Verbraucher ist nur einmal im System vorhanden und entnimmt Elemente aus der Liste und verarbeitet sie.

b) Welche Probleme können bei der gegebenen Implementierung auftreten? Geben Sie jeweils ein Beispiel an.

c) Schützen Sie Erzeuger und Verbraucher vor den in b) genannten Problemen durch die Verwendung eines Monitors.

Geben Sie die **Schnittstelle** des Monitors mit allen benötigten Variablen und Funktionen in Pseudocode **inklusive Kommentaren** an und beschreiben Sie, wie der o. g. Quellcode für Erzeuger und Verbraucher angepasst werden muss, wenn dieser Monitor verwendet werden soll.

Hinweis: Es ist nicht notwendig, hier die Realisierung des Monitors anzugeben.

Fortsetzung nächste Seite!

3 Virtuelle Adressabbildung

a) Probleme der direkten Speicherbelegung:

Problem 1: Welches Problem kann durch die absolute Adressierung von Programmcode entstehen?

Problem 2: Welches Problem kann durch die Zuweisung eines festen Speichersegments an einen Prozess entstehen?

Problem 3: Welches Problem kann durch eine fehlende Regulierung der Speicherzugriffe verschiedener Prozesse entstehen?

Erklären Sie, wieso diese Probleme jeweils durch die Verwendung von virtuellem Speicher gelöst werden.

b) Gegeben sei ein Rechner mit 5 Bit physikalischem und virtuellem Adressraum und 5 Bit breiten Worten. Der Inhalt des physikalischen Speichers ist unten angegeben. Es gelten folgende Annahmen:

- Die Seiten sind jeweils 4 Worte groß.
- Die Basisadresse der Seitentabelle ist `10000`.
- Ein Seitentabelleneintrag besteht aus 5 Bit, die ersten (= höherwertigsten) 3 Bits geben die physikalische Seitenrahmennummer an, das 4. Bit ist das *present-* bzw. *mapped*-Bit und das 5. Bit ist hier nicht von Bedeutung.
- Bei der zweistufigen Adressierung werden 2 Bit zur Indizierung der Seitentabellen der zweiten Stufe verwendet.

Adresse	*Inhalt*	*Adresse*	*Inhalt*	*Adresse*	*Inhalt*	*Adresse*	*Inhalt*
00000	10111	01000	01010	10000	10111	11000	00110
00001	10011	01001	00011	10001	11010	11001	01010
00010	11011	01010	10010	10010	11100	11010	00010
00011	11101	01011	00001	10011	01101	11011	00011
00100	01000	01100	10000	10100	00101	11100	10110
00101	00110	01101	10010	10101	11110	11101	01111
00110	01111	01110	11110	10110	00101	11110	11010
00111	11000	01111	01101	10111	10001	11111	11100

Welche Daten stehen an der Adresse `01011`, wenn Sie diese Adresse als physikalische Adresse, als einstufige und als zweistufige virtuelle Adresse verwenden?

c) Geben Sie eine einstufige, virtuelle Adresse für die physikalische Adresse `10101` an.
Geben Sie eine einstufige, virtuelle Adresse an, welche zu einem Seitenfehler führt.

Fortsetzung nächste Seite!

4. Virtueller Speicher und Seitenverdrängung

a) Um was handelt es sich bei der so genannten Belady-Anomalie?
Welcher Zusammenhang besteht zwischen der Belady-Anomalie und der Keller-Eigenschaft?

b) Gegeben sei eine Maschine mit einem physikalischen Speicher für 4 Seitenrahmen und ein virtueller Adressraum mit 6 Seiten A bis F. Gegeben sei die folgende Referenzkette:

A B C A D E B F A C B E

Geben Sie für die folgenden Seitenverdrängungsstrategien an, welche Seiten sich zu jedem Zeitpunkt im realen Speicher befinden:

- Optimal (nach Belady)
- Least Frequently Used (Der Zugriffszähler einer Seite wird hier bei ihrer Verdrängung auf Null zurückgesetzt.)

Bei Zweideutigkeiten wird alphabetische Ordnung verwendet, das heißt, es wird die Seite entfernt, deren Bezeichner im Alphabet zuerst kommt.

Markieren Sie auch für jede Strategie die Anzahl der auftretenden Seitenfehler.

Orientieren Sie sich an folgendem Muster:

Belady:

A	*B*	*C*	*A*	*D*	*E*	*B*	*F*	*A*	*C*	*B*	*E*

Anzahl der Seitenfehler: _____

Least Frequently Used:

A	*B*	*C*	*A*	*D*	*E*	*B*	*F*	*A*	*C*	*B*	*E*

Anzahl der Seitenfehler: _____

Welcher Nachteil kann durch die Verwendung von LFU entstehen?