Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer

Kennzahl: _____

Kennwort: _____

Arbeitsplatz-Nr.: _____

Frühjahr 2013

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: **22**

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
(Datenbanksysteme)

Teilaufgabe 1:

1. Allgemeinwissen

a) Definieren Sie den Begriff Datenbanksystem und grenzen Sie die Funktionalitäten eines Datenbanksystems von denen eines Dateisystems ab.
b) Nennen Sie die Phasen des Datenbankentwurfsprozesses und charakterisieren Sie diese kurz.
c) Nennen und beschreiben Sie kurz die vier wesentlichen Eigenschaften von Transaktionen.
d) Nennen und erläutern Sie kurz vier Beispiele für Integritätsbedingungen.

2. ER-Modellierung und Relationenmodell

a) Erstellen Sie ein ER-Diagramm bestehend aus Entitäts- und Beziehungstypen sowie Attributen für folgendes Szenario. Geben Sie auch die Kardinalitäten mit an. Verwenden Sie bei Entitäten und Attributen ausschließlich die beschriebenen, soweit dies möglich ist. Verwenden Sie keine künstlichen Schlüssel, es sei denn, diese sind der Aufgabenstellung eindeutig zu entnehmen. Die Kardinalitätseinschränkungen können Sie entweder in der (min,max)-Notation oder der einfachen Notation nach Chen (1:1, 1:N, M:N) angeben.

Als **Szenario** dient eine Mitarbeiterverwaltung, die folgendermaßen beschrieben ist:

1. Mitarbeiter werden über ihre Sozialversicherungsnummer identifiziert. Sie haben einen Namen, bestehend aus Vor- und Nachnamen, sowie ein Geburtsdatum und ein Alter, das aus dem Geburtsdatum errechnet wird.
2. Ein Mitarbeiter kann der Vorgesetzte anderer Mitarbeiter sein.
3. Jeder Mitarbeiter ist einer Abteilung zugeordnet. Eine Abteilung wird über ihren Namen und eine Nummer identifiziert (z.B. „Entwicklung“, „3“). Zu einer Abteilung werden mehrere Orte gespeichert, an denen die Abteilung angesiedelt ist.
4. Mitarbeiter können Abteilungen leiten. Jede Abteilung wird von einem Mitarbeiter geleitet, aber nicht jeder Mitarbeiter muss eine Abteilung leiten.
5. Abteilungen können Projekte realisieren. Ein Projekt wird durch eine Nummer identifiziert und findet an einem Ort statt.
6. Jeder Mitarbeiter arbeitet an einem oder mehreren Projekten. An jedem Projekt muss mindestens ein Mitarbeiter arbeiten. Zusätzlich wird gespeichert, ab wann ein Mitarbeiter an einem Projekt arbeitet.

Hinweis: In der Formulierung der Beziehungen sind die Kardinalitätseinschränkungen genau zu beachten. Im Allgemeinen gilt: Alles was nicht explizit eingeschränkt ist, muss im Modell auch uneingeschränkt bleiben (keine Einschränkungen in eine Beschreibung hineininterpretieren). Alles was explizit eingeschränkt ist, darf im Modell nicht einfach uneingeschränkt bleiben (alle formulierten Einschränkungen sind zu erkennen).

Fortsetzung nächste Seite!

b) Erstellen Sie zu dem E/R-Diagramm aus Teil (a) ein Relationenschema. Seien Sie dabei so vollständig wie möglich – bspw. Berücksichtigung totaler Partizipationen. Vermeiden Sie unnötiges Ausprägen von Relationen bei allen Beziehungen.

Beispiel für die Notation:

```
Relationenname (Primärschlüssel, Attribut1, Attribut2, …,
    Fremdschlüssel[AndereRelation],
    (FremdschlüsselattributA,
    FremdschlüsselattributB)[DritteRelation])
    Attribut2 NOT NULL
```

3. Normalformen

Gegeben sei die nachfolgende relationale Datenbank mit unterstrichenen Schlüsselattributen. Sie enthält Informationen über Opernaufführungen. Relation "Opernvorstellungen":

Oper	Komponist	Datum	Ort	Opernhaus	Plätze	Dirigent	Dauer	Solist
Rigoletto	Verdi	08.04.2013	Wien	Staatsoper	2298	Lopez-Cobos	2:00	Polenzani
Don Carlos	Verdi	09.04.2013	Wien	Staatsoper	2298	de Billy	4:00	Youn
Rigoletto	Verdi	11.04.2013	Dresden	Semperoper	1300	Fisch	2:00	Berrugi
Macbeth	Verdi	16.04.2013	Mailand	Scala	2000	Gergiev	2:45	Vassallo
Carmen	Bizet	20.05.2013	Wien	Staatsoper	2298	de Billy	2:45	Garanca
Carmen	Bizet	02.06.2013	Wien	Staatsoper	2298	de Billy	2:45	Alagna
Don Carlos	Verdi	12.10.2013	Mailand	Scala	2000	Luisi	4:00	Pape

Gegeben seien ebenso die folgenden funktionalen Abhängigkeiten:

FD1: Oper → Komponist, Dauer
FD2: Ort → Opernhaus, Plätze
FD3: Oper, Ort → Dirigent
FD4: Oper, Datum, Ort → Solist

a) Beschreiben Sie kurz, welche Redundanzen in der Datenbank vorhanden sind und welche Anomalien auftreten können. Geben Sie ein Beispiel für jede Anomalie an.

b) Welcher Normalform genügt das angegebene Relationenschema? Begründen Sie Ihre Entscheidung.

c) Erläutern Sie kurz, welchen Nachteil Normalisierung allgemein für die Anfragebearbeitung haben kann.

Fortsetzung nächste Seite!

4. Anfrageverarbeitung

a) Zeichnen Sie für folgendes SQL-Statement einen nicht-optimierten Anfragegraphen.

```
SELECT e1.name AS ename, e1.salary, dept.name AS dname,
       AVG (e2.salary) AS avg_salary
  FROM employee e1, employee e2, dept
 WHERE e1.dept_id = dept.id
   AND e2.dept_id = e1.dept_id
   AND dept.name LIKE 'Sales %'
 GROUP BY e1.name, e1.salary, dept.name
 HAVING AVG(e2.salary) > 2*e1.salary
```

b) Wie kann die relationale Algebra zur Optimierung von SQL-Anfragen verwendet werden?

Fortsetzung nächste Seite!

Teilaufgabe 2:

Aufgabe 1: Relationale Algebra und SQL

Gegeben sei folgendes Schema:

- Züge:
  {[ZugNr: integer, AnzahlWagons : integer]}
- Städte:
  {[StadtNr: integer, Name: string]}
- Bahnhöfe:
  {[BahnhofNr: integer, Name: string, LiegtInStadtNr: integer]}
- Verbindungen:
  {[ZugNr: integer, BahnhofNrVon: integer, BahnhofNrNach: integer, AbfahrtsZeit: date, AnkunftsZeit: date]}

Hinweis:
Ein Element $\in$ date hat der Einfachheit halber die Form: „5:32PM“. Wir nehmen an, dass Verbindungen jeden Tag gelten.

Aufgabe 1.1:
Geben Sie tabellarisch eine Beispielausprägung mit mindestens zwei Zeilen pro Tabelle des angegeben Schemas an.

Aufgabe 1.2

1. Finden Sie die Züge, welche morgens um 5 Uhr von dem Bahnhof „Berlin Ostbahnhof“ zu dem Bahnhof „München Hauptbahnhof“ fahren **(in relationaler Algebra).**
2. Finden Sie die Städte, von denen mindestens ein Zug mit mehr als drei Wagons zu einem Bahnhof in Mannheim fährt **(in relationaler Algebra).**
3. In welcher Stadt liegt der Bahnhof „Coesfeld Schulzentrum“? **(in relationaler Algebra)**
4. Finden Sie die Namen der Städte an denen zur gleichen Uhrzeit mehr als ein Zug ankommt **(in relationaler Algebra).**

Fortsetzung nächste Seite!

Aufgabe 1.3:

Beantworten Sie folgende Fragen in SQL:

1. An welchen Uhrzeiten fahren Züge von „München Hauptbahnhof" nach „Berlin Ostbahnhof"?
2. Finden Sie den Zug oder die Züge mit den meisten Wagons.
3. Welche Bahnhöfe (BahnhofNr angeben) sind mit dem Bahnhof "München Hauptbahnhof" direkt verbunden? Geben Sie die Bahnhöfe duplikatfrei aus. „München Hauptbahnhof" selbst darf nicht im Ergebnis vorkommen.
4. Finden sie alle Bahnhöfe (BahnhofNr), die mit **genau zweimal** Umsteigen von dem Bahnhof „München Hauptbahnhof" aus erreicht werden können.
5. Geben Sie die Namen der Bahnhöfe an, die überdurchschnittlich oft als Ziel vorkommen. Geben Sie außerdem zu jedem der Bahnhöfe die Anzahl der ankommenden Züge an. Beachten Sie hierbei, dass auch Bahnhöfe, die von keinem Zug erreicht werden können, den Durchschnitt beeinflussen.

Aufgabe 1.4:

Gegeben sei folgende SQL Anfrage:

```
SELECT z.ZugNr FROM
        Zug z, Bahnhof b1, Bahnhof b2, Verbindung v
WHERE
        v.BahnhofNrVon = b1.BahnhofNr AND
        v.BahnhofNrNach = b2.BahnhofNr AND
        b1.Name = "München" AND
        b2.Name = "Berlin" AND
        z.ZugNr = v.ZugNr
```

Übertragen Sie das gegebene SQL Statement in die relationale Algebra und optimieren Sie den Ausdruck logisch. Verwenden Sie die Operatorbaum-Syntax.

Fortsetzung nächste Seite!

Aufgabe 2: ER Modellierung

Sie sind beauftragt, bei der Entwicklung des ersten Prototypen eines sozialen Netzwerkes die Datenmodellierung zu übernehmen. Der erste Prototyp soll folgende Anforderungen erfüllen:

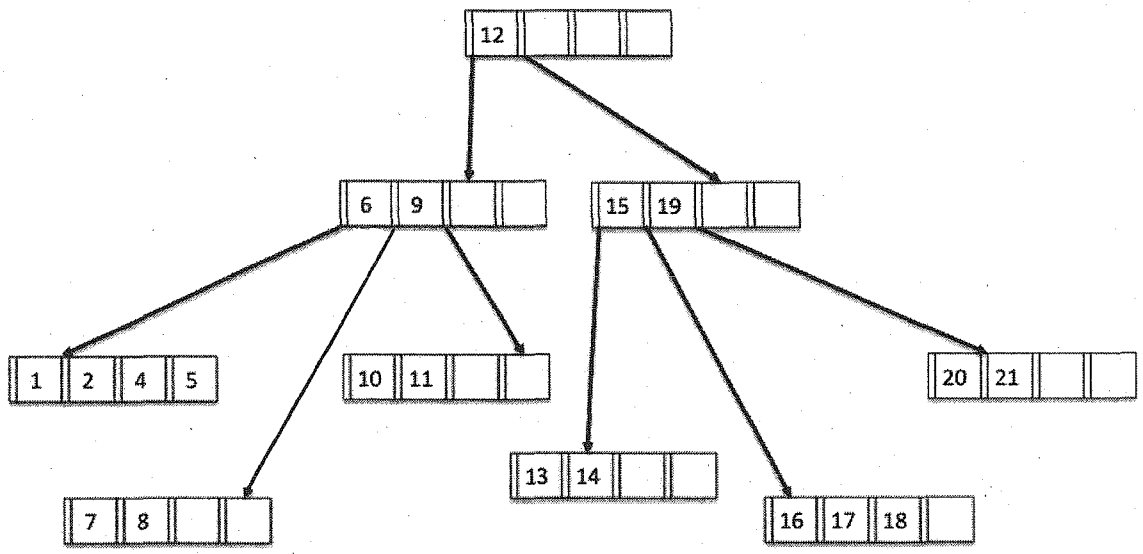
Benutzer können sich mit ihrem Nutzernamen und Kennwort anmelden. Ein Benutzer hat eine sogenannte Wall, auf die andere Benutzer Texte schreiben können. Benutzer können Freunde von anderen Benutzern sein. Außerdem können Benutzer sich in verschiedenen Gruppen anmelden. Gruppen haben ebenfalls eine sogenannte Wall, auf der Gruppenmitglieder Nachrichten hinterlassen können.

Erstellen Sie ein ER-Diagramm, welches die obengenannten Anforderungen erfüllt. Denken Sie daran Ihrem ER-Diagramm Funktionalitätsangaben hinzuzufügen.

Fortsetzung nächste Seite!

Aufgabe 3: B-Baum

Gegeben sei der folgende B-Baum



1. Was bedeutet k bei einem B-Baum mit Grad k? Geben Sie k für den obigen B-Baum an.
2. Was sind die Vorteile von B-Bäumen im Vergleich zu binären Baumen?
3. Wozu werden B-Bäume in der Regel verwendet und wieso?
4. Fügen Sie den Wert **3** in den B-Baum ein, und zeichnen Sie den vollständigen B-Baum nach dem Einfügen und möglichen darauf folgenden Operationen.
5. Entfernen Sie aus dem **ursprünglichen** B-Baum den Wert **19**. Zeichnen Sie das vollständige Ergebnis nach dem Löschen und möglichen darauf folgenden Operationen. Sollte es mehrere richtige Lösungen geben, reicht es eine Lösung zu zeichnen.

Themenschwerpunkt B
(Betriebssysteme)

Teilaufgabe 1:

Aufgabe 1:

Synchronisation mit Semaphoren

a) Gegeben sei folgende informelle Definition der Operation `V` auf Semaphore `s` durch Pseudocode. Die Semaphore sei symbolisch durch `int s` dargestellt. Es wird als gegeben angenommen, dass die Operation atomar ist und wechselseitig ausgeschlossen ausgeführt wird.

```
public void V (int s) {
   s = s + 1;
   if (s <= 0) { Führe genau einen der bezüglich s
     wartenden Prozesse in den Zustand rechenwillig über }
}
```

Geben Sie eine analoge Definition der Operation `P` an.

b) Gegeben sei ein System mit drei Prozessen V1, V2, E und vier boolschen Semaphoren R, S, T, U (jeweils mit 1 initialisiert). Die drei Prozesse laufen folgendermaßen ab:

```
Process V1
{
 while (TRUE)
 {
  P(S);
  P(T);
  P(U);
  <Aktion V1>
  V(T);
  V(S);
  V(U);
 }
}
```

```
Process V2
{
 while (TRUE)
 {
  P(R);
  P(T);
  P(U);
  <Aktion V2>
  V(R);
  V(U);
  V(T);
 }
}
```

```
Process E
{
 while (TRUE)
 {
  P(U);
  P(T);

  <Aktion E>
  V(T);
  V(U);

 }
}
```

Ist in dem gegebenen Prozesssystem eine Verklemmung möglich? Begründen Sie Ihre Antwort.

Fortsetzung nächste Seite!

c) Ein Obsthändler hat ein Geschäft für Obst mit nur einem Parkplatz. Beliefert wird er von einem Lieferanten L, der bei jeder Lieferung drei Kisten Obst bringt. Kunden K holen jeweils eine Kiste Obst ab. Zum Anliefern und Abholen muss der einzige Parkplatz benutzt werden. Für sich alleine betrachtet laufen die beteiligten Prozesse folgendermaßen ab:

```
Process K
{
 while (TRUE)
 {
  <zum Parkplatz fahren>;
  <1 Kiste Obst abholen>;
  <Parkplatz verlassen>;
 }
}
```

```
Process L
{
 while (TRUE)
 {
  <zum Parkplatz fahren>;
  <3 Kisten Obst anliefern>;
  <Parkplatz verlassen>;
 }
}
```

Synchronisieren Sie diese zwei Prozesse, indem Sie in Pseudocode-Notation in geeigneter Weise Semaphore deklarieren und Semaphor-Operationen einfügen.

Die Pseudocode-Operation zum Deklarieren eines Semaphors mit Namen `name` mit Startwert `n` laute:

```
name(n);
```

Die Pseudocode-Operationen zum Aufrufen von P und V auf der Semaphore mit Namen `name` lauten:

```
P(name);
```

```
V(name);
```

Sperrphasen sind möglichst kurz zu halten und folgende Bedingungen müssen erfüllt sein:

i) Der Parkplatz darf nur von einem Prozess gleichzeitig benutzt werden.

ii) Ein Kunde K darf nur zum Parkplatz fahren, wenn noch mindestens eine Kiste Obst im Geschäft ist.

iii) Das Geschäft hat beschränkte Kapazität; es kann höchstens 12 Kisten Obst aufnehmen.

iv) Der Lieferant L darf nur zum Parkplatz fahren, wenn er seine Lieferung vollständig abladen kann.

v) Zu Beginn sei das Geschäft leer und der Parkplatz frei.

d) Listen Sie alle benötigten Semaphoren auf und erklären Sie ihren Zweck bzw. ihre Pragmatik.

e) Fügen Sie in dem oben aufgelisteten Pseudocode des Ablaufs der beteiligten Prozesse an geeigneter Stelle entsprechende Aufrufe zum Deklarieren der Semaphoren und der Aufrufe von P und V ein.

Fortsetzung nächste Seite!

Aufgabe 2:

Scheduling

a) Erläutern Sie kurz allgemein die Aufgabe des *Schedulers* im Betriebssystem.

b) Gegeben seien fünf Prozesse mit ihren jeweiligen Ankunftszeiten, Bedienzeiten und Deadlines:

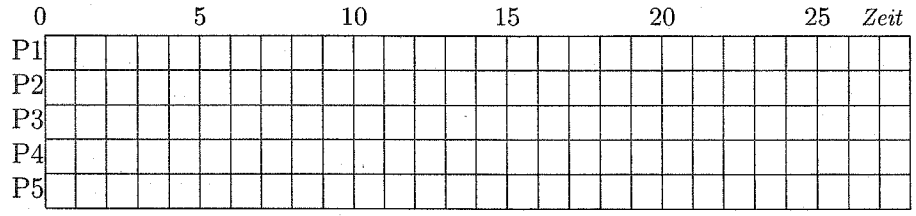
Prozess	*Ankunfszeit*	*Bedienzeit*	*Deadline*
P1	0	4	19
P2	7	5	15
P3	3	4	17
P4	10	2	18
P5	11	3	21

Die *Deadline* ist der Zeitpunkt, an dem ein Prozess spätestens beendet werden sollte. Alle Prozesse sollen aber in jedem Fall komplett bedient werden, auch wenn ihre Deadline bereits abgelaufen ist.

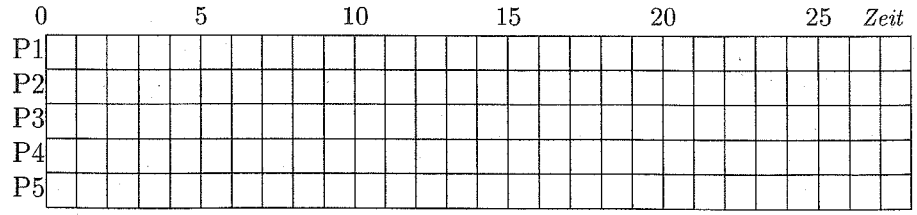
Ein *Kontextwechsel* benötige in diesem System *1 Zeiteinheit.* Der Kontextwechsel ist auch für den ersten rechnenden Prozess zu berücksichtigen.

Übertragen Sie die nachfolgenden Gantt-Diagramme auf Ihr Blatt und tragen Sie für die angegebenen Scheduling-Verfahren ein, welcher Prozess zu welcher Zeit die CPU erhält. Markieren Sie rechenwillige Prozesse mit - und rechnende Prozesse mit `X` in der jeweiligen Spalte.

i) *Shortest Job First (SJF)*: Prozess mit der kürzesten Bedienzeit wird ausgeführt, nicht unterbrechend.



ii) *Earliest Deadline First (EDF)*: Prozess mit der nächsten, in der Zukunft liegenden, Deadline wird ausgewählt, unterbrechend nur beim Eintreffen neuer Prozesse.



Fortsetzung nächste Seite!

c) Berechnen Sie die *mittlere Verweilzeit* $\overline{V}$ für die beiden Strategien der vorangehenden Teilaufgabe. Nutzen Sie zur Berechnung folgende Formel:

- c_i bezeichnet die **Abgangszeit** (der Zeitpunkt seiner fertigen Bearbeitung) des Prozesses Pi und a_i seine Ankunftszeit
- $v_i = c_i - a_i$ bezeichnet die **Verweilzeit** des Prozesses Pi;
- Gibt es $n \in N$ Prozesse, so gilt für die **mittlere Verweilzeit**:

$$\overline{V} = \frac{1}{n} \sum_{i=1}^{n} v_i$$

d) Für welche Art von Systemen erscheint Earliest Deadline First (EDF) prinzipiell sehr gut geeignet?

Fortsetzung nächste Seite!

Aufgabe 3:

Seitenersetzung

a) Gegeben sei ein System mit drei verfügbaren Frames ***F0, F1, F2***, welche zu Beginn leer sind. Tragen Sie für die Seitenanforderungsfolge **1, 2, 3, 4, 5, 1, 2, 5, 1, 2, 3, 4, 5** und die Verfahren **FIFO** (First-In-First-Out), **LRU** (Least-Recently-Used) und **OPT** (Belady's optimale Strategie) die jeweils entstehende Belegung der Speicherkacheln ein.

Übertragen Sie hierfür die unten gegebenen Schemata auf Ihr Blatt und vervollständigen Sie sie. Geben Sie für jedes Verfahren die Anzahl der resultierenden Seitenfehler (engl. page faults) an.

FIFO	**1**	**2**	**3**	**4**	**5**	**1**	**2**	**5**	**1**	**2**	**3**	**4**	**5**
F0													
F1													
F2													

LRU	**1**	**2**	**3**	**4**	**5**	**1**	**2**	**5**	**1**	**2**	**3**	**4**	**5**
F0													
F1													
F2													

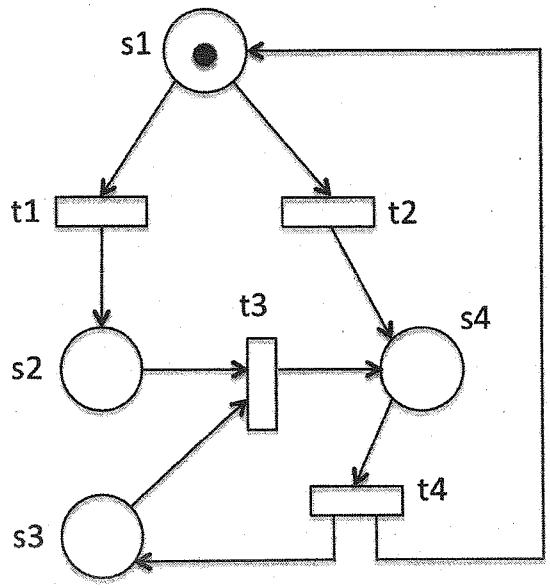
OPT	**1**	**2**	**3**	**4**	**5**	**1**	**2**	**5**	**1**	**2**	**3**	**4**	**5**
F0													
F1													
F2													

b) Die optimale Strategie von Belady minimiert die Anzahl der Seitenfehler. Warum wird sie trotzdem in realen Paging-Systemen nicht eingesetzt?

c) Was versteht man unter Thrashing? Nennen Sie mindestens eine sinnvolle Gegenmaßnahme, die das Betriebssystem ergreifen kann.

Fortsetzung nächste Seite!

Aufgabe 4:
Petri-Netze
Gegeben sei folgendes boolsches Petri-Netz



a) Erläutern Sie zunächst allgemein, was man unter *Aushungern (engl. starvation)* bei Petri-Netzen versteht.

b) Geben Sie den *Erreichbarkeitsgraphen* des gegebenen Petri-Netzes an.

c) Existiert in dem gegebenen Petri-Netz ein *nicht-sequentieller Ablauf*? Falls ja, geben Sie ein Beispiel an. Falls nein, begründen Sie Ihre Antwort.

d) Ist in dem gegebenen Petri-Netz eine *Verklemmung (engl. deadlock)* erreichbar? Begründen Sie Ihre Antwort.

Fortsetzung nächste Seite!

Teilaufgabe 2:

Aufgabe 1:

Prozesse
Beantworten Sie folgende Fragen zum Thema Prozesse.

a) Welche drei Arten von Informationen enthält der Prozesskontrollblock (PCB)? Nennen Sie je ein Bespiel für jede der drei Arten.

b) Was ist der Unterschied zwischen PCB und Prozess-Image?

c) Nennen Sie einen Vorteil der sich ergibt, wenn ein Prozessmodell zwischen vielen verschiedenen Prozesszuständen differenziert. Geben Sie dafür ein Beispiel an!

d) Nennen Sie einen Nachteil der sich ergibt, wenn ein Prozessmodell zwischen vielen verschiedenen Prozesszuständen differenziert. Geben Sie dafür ein Beispiel an!

Fortsetzung nächste Seite!

Aufgabe 2:

Multilevel Feedback Queuing

In dieser Aufgabe sollen Sie ein modifiziertes Multilevel Feedback Queuing (MLFQ) Verfahren anwenden, welches auf unterschiedlichen Leveln verschieden lange Zeitscheiben verwendet. Die drei folgenden Prioritätsklassen stehen zur Verfügung:

Priorität	Verfahren
0	FIFO mit Quantum 1
1	FIFO mit Quantum 2
2	Round Robin mit Quantum 4

Das Quantum gibt die Dauer der Zeitscheibe auf dem entsprechenden Level an.

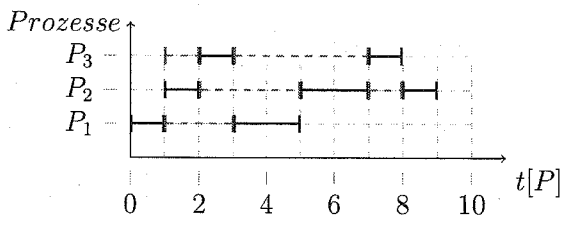
Gehen Sie davon aus, dass jeder Prozess sein Zeitquantum stets vollständig ausnutzt d.h. kein Prozess gibt den Prozessor freiwillig frei (Ausnahme: bei Prozessende).

Trifft ein Prozess zum Zeitpunkt t ein, so wird er direkt zum Zeitpunkt t beim Scheduling berücksichtigt. Wird ein Prozess zum Zeitpunkt t' unterbrochen, so reiht er sich auch zum Zeitpunkt t' wieder in die Warteschlange ein. Sind zwei Prozesse absolut identisch bezüglich ihrer relevanten Werte, so werden die Prozesse nach aufsteigender Prozess–ID in die Warteschlange eingereiht. Diese Annahme gilt sowohl für neu im System eintreffende Prozesse, als auch für den Prozess, dem der Prozessor u.U. gerade entzogen wird!

Beispiel: Es seien folgende Ankunfts– und Rechenzeiten für die drei Beispielprozesse P_1, P_2 und P_3 gegeben:

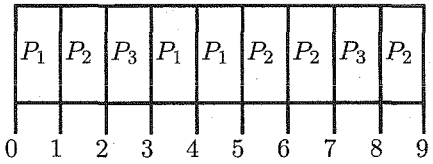
Prozess	Ankunftszeitpunkt	Rechenzeit
P_1	0	3
P_2	1	4
P_3	1	2

Das folgende Diagramm veranschaulicht das Scheduling der drei Prozesse P_1, P_2 und P_3 entsprechend dem oben beschriebenen Verfahren.



Fortsetzung nächste Seite!

Das Scheduling kann man auch mit Hilfe eines Gantt–Charts visualisieren:

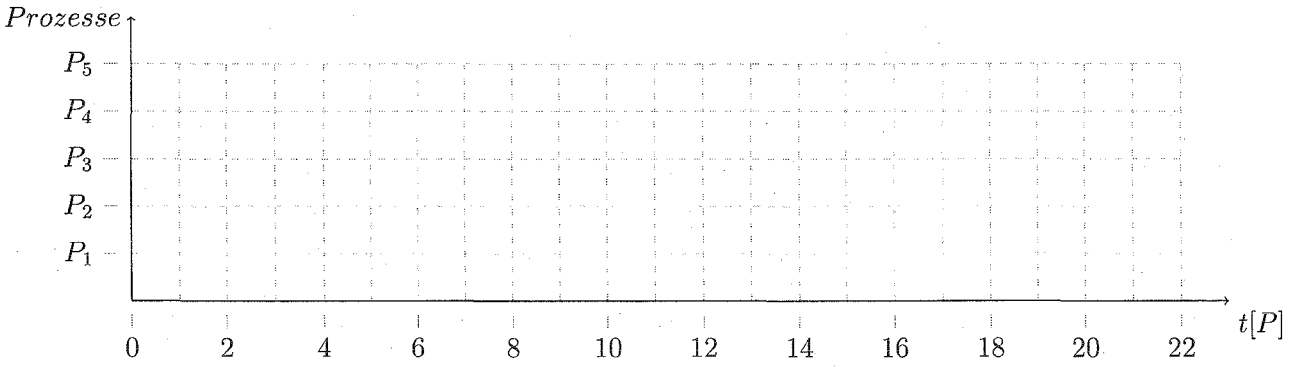


Bearbeiten Sie unter den gegebenen Voraussetzungen nun die folgenden Aufgaben:

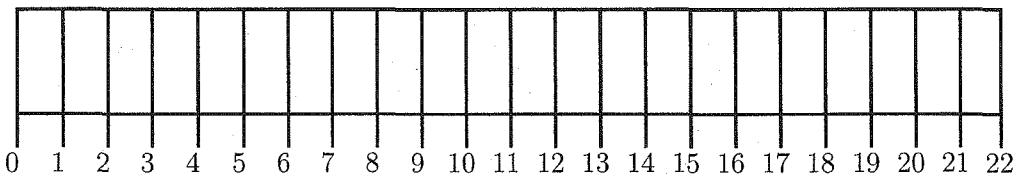
a) Gegeben seien die Prozesse $P_1, \ldots, P_5$ mit den folgenden Ankunfts– und Rechenzeiten:

Prozess	Ankunftszeitpunkt	Rechenzeit
P_1	1	4
P_2	3	4
P_3	3	2
P_4	5	6
P_5	7	5

Übertragen Sie das nachfolgende Diagramm auf Ihr Blatt und veranschaulichen Sie die Zuteilung der Rechenzeit an die Prozesse für die oben beschriebene Variante des MLFQ-Verfahrens im Diagramm. Kennzeichnen Sie zudem für jeden Prozess seine Ankunftszeit.



b) Übertragen Sie das nachfolgende Gantt-Diagramm auf Ihr Blatt. Veranschaulichen Sie die Zuteilung der Rechenzeiten an die Prozesse $P_1, \ldots, P_5$ aus Aufgabe a) des nachfolgenden Gantt-Diagramms.



c) Geben Sie für die Prozesse P_4 und P_5 jeweils die Verweilzeit (Antwortzeit) und die Wartezeit an.

d) Mit welcher Strategie ist Round Robin mit einem Quantum $Q = \infty$ gleichzusetzen?

Fortsetzung nächste Seite!

Aufgabe 3:

Buddy-Systeme

Ein Computer verfüge über einen 2 GByte großen Hauptspeicher, der nach dem Buddy-Verfahren verwaltet wird.

a) Wieviele Bits benötigt man mindestens, um diesen Speicher byteweise zu adressieren?

b) Nacheinander sollen nun die folgenden vier Programme in den Speicher geladen werden:

- P_1: 100 MByte
- P_2: 300 MByte
- P_3: 40 MByte
- P_4: 80 MByte

Zeichnen Sie den Buddy-Baum jedesmal, nachdem eines der Programme in den Speicher geladen wurde. Verwenden Sie dabei folgende Notation:


freies Segment


gesplittetes Segment

von Programm P_i belegtes Segment

Tragen Sie neben jedem freien Segment die Größe des freien Speicherbereichs an. Tragen Sie neben jedem belegten Segment die Größe des allokierten Speicherbereichs sowie die Speicheradressen des Segments an.

Hinweis: Es wird immer das am weitesten links stehende Segment gesplittet und der am weitesten links stehende Buddy belegt.

c) Die Programme aus Teilaufgabe b) benötigen insgesamt 520 MByte Speicherplatz. Damit müssten noch 1528 MByte Speicher nutzbar sein. Warum ist das im Beispiel nicht der Fall? Welcher Effekt kommt hier zum Tragen? Wieviel nutzbarer Speicherplatz steht für weitere Programme insgesamt noch zur Verfügung?

d) Gegeben ist eine weitere Anfrage:

- P_5: 600 MByte

Kann P_5 noch zusätzlich in den Speicher geladen werden? Falls ja, zeichnen Sie den Buddy-Baum nach der Belege-Operation. Falls nein, begründen Sie Ihre Antwort.

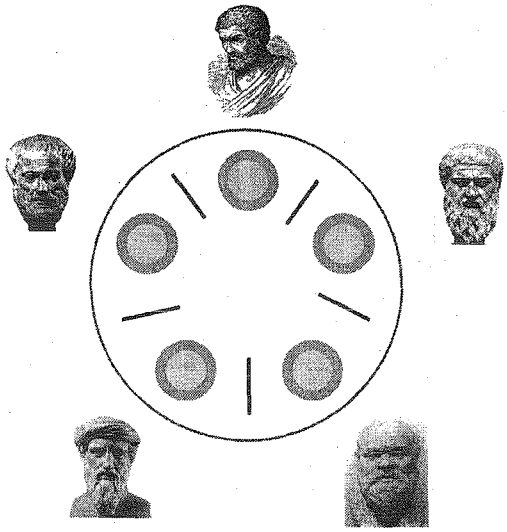
Fortsetzung nächste Seite!

Aufgabe 4:

Koordination von Threads

In dieser Aufgabe soll das bekannte Philosophenproblem (Dining Philosophers) mit Hilfe der Programmiersprache Java implementiert werden.

Beim Philosophenproblem sitzen fünf Philosophen an einem runden Tisch beim Essen (siehe Abbildung). Vor jedem Philosophen befindet sich ein Teller voller Reis und zwischen je zwei Tellern befindet sich ein Stäbchen. Ein Philosoph, der hungrig ist, benötigt sowohl das linke als auch das rechte Stäbchen, um essen zu können. Hat ein Philosoph zwei Stäbchen, so isst dieser, bis er satt ist, erst dann legt er die Stäbchen an die ursprünglichen Plätze zurück, so dass seine Nachbarn davon Gebrauch machen können. Ein Philosoph legt ein Stäbchen niemals zurück, bevor er nicht gegessen hat.



a) Beschreiben Sie, ab wann man sich beim Philosophenproblem im kritischen Bereich befindet und ab wann man diesen wieder verlässt.

b) Wenn man beim Philosophenproblem keine Einschränkung beim Zugriff auf die Stäbchen formuliert, kann es zu einem Deadlock kommen.

   i) Beschreiben Sie informell einen Ablauf für das Philosophenproblem, der zu einem Deadlock fährt.

   ii) Geben Sie zwei Möglichkeiten an, wie sich die Deadlock-Bedingung "Circular Wait" vermeiden lässt, so dass keine Deadlock-Situation im Fall des Fünf-Philosophenproblems auftritt.

Fortsetzung nächste Seite!

c) Im Folgenden sollen Sie das Philosophenproblem in Form der drei Java Klassen `Philosoph`, `Staebchen` und `Dinner` implementieren. Der Quelltext der Klasse `Dinner` ist bereits vorgegeben und sieht folgendermaßen aus:

```
1  public class Dinner {
2     public static void main(String[] args) {
3        Thread[] philosoph = new Thread[5];
4        Staebchen staebchen[] = new Staebchen[5];
5
6        for (int i = 0; i < 5; ++i) {
7           staebchen[i] = new Staebchen();
8        }
9        for (int i = 0; i < 5; ++i) {
10          philosoph[i] =
11             new Philosoph(i, staebchen[(i+4)%5], staebchen[i]);
12          philosoph[i].start();
13       }
14    }
15 }
```

i) Implementieren Sie zunächst die Klasse `Staebchen`. Diese soll die beiden Methoden `get()` (Stäbchen vom Tisch nehmen) und `put()` (Stäbchen an den ursprünglichen Platz zurücklegen) zur Verfügung stellen. Halten Sie sich dazu an den nachfolgenden Code-Rahmen. Referenzieren Sie jeden Teil Ihre Lösung entsprechend den Angaben im Code-Rahmen. Achten Sie darauf, dass der Zugriff auf ein Stäbchen im wechselseitigen Ausschluss erfolgt und keine Race Conditions auftreten können! Verwenden Sie für Ihre Implementierung das vorgegebene Attribut `wirdBenutzt`.

Kommentieren Sie Ihre Lösung ausführlich!

Code-Rahmen der Klasse `Staebchen`. Übertragen Sie den Code-Rahmen mit Ihren Ergänzungen auf Ihr Blatt.

```
1  public class Staebchen {
2
3     // Attribut
4     private boolean wirdBenutzt = false;
5
6     // ============================================================
7     // Teil (1):
8     // Fügen Sie hier den Programmtext der get()-Methode ein.
9     // ============================================================
10    // ...
11
12    // ============================================================
13    // Teil (2):
14    // Fügen Sie hier den Programmtext der put()-Methode ein.
15    // ============================================================
16    // ...
17 }
```

Fortsetzung nächste Seite!

ii) Implementieren Sie nun die Klasse `Philosoph`. Achten Sie darauf, dass Ihre Implementierung frei von Deadlocks ist. Verwenden Sie dazu den nachfolgenden Code-Rahmen (Fortsetzung auf der nächsten Seite!). Verwenden Sie ausschließlich die vorgegebenen Attribute. Achten Sie darauf, dass sich Ihre Implementierung zusammen mit der Klasse `Dinner` fehlerfrei übersetzen lässt.

Kommentieren Sie Ihre Lösung ausführlich!

Hinweis: Die Klasse `Random` dient dazu, Zufallszahlen zu erzeugen. Die Methode `nextInt(int positiveZahl)` liefert eine Zahl zwischen 0 (inklusive) und
`positiveZahl` (exklusive). Mit Hilfe dieser Methode wird die Zeit, die ein Philosoph isst bzw. denkt simuliert.

Code-Rahmen der Klasse `Philosoph`. Übertragen Sie den Code-Rahmen mit Ihren Ergänzungen auf Ihr Blatt.

```
1  import java.util.Random;
2
3  public class Philosoph extends Thread {
4     private static Random rand = new Random();
5
6     // Attribute
7     private Staebchen links;
8     private Staebchen rechts;
9     private int id;
10
11    // ================================================================
12    // Teil (3):
13    // Fügen Sie hier den Programmtext des Konstruktors ein.
14    // ================================================================
15    // ...
16
17
18    // run()-Methode
19    public void run() {
20       try {
21          while(true) {
22             // Philosoph denkt
23             System.out.println("Philosoph "+ id + " denkt");
24             Thread.sleep(rand.nextInt(500) + 500);
25             // ================================================================
26             // Teil (4):
27             // Philosoph ist hungrig.
28             // Fügen Sie hier den Programmtext zum Erhalten der
29             // Stäbchen ein.
30             // ================================================================
31             // ...
32
```

Fortsetzung nächste Seite!

```
33                  // Philosoph isst
34                  System.out.println("Philosoph "+ id + " isst");
35                  Thread.sleep(rand.nextInt(500) + 500);
36                  // ================================================
37                  // Teil (5):
38                  // Philosoph ist satt.
39                  // Fügen Sie hier den Programmtext zum Zurücklegen der
40                  // Stäbchen ein.
41                  // ================================================
42                  // ...
43
44              }
45          }
46          catch(InterruptedException ie) {
47          }
48      }
49  }
```