Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer

Kennzahl: _____

Kennwort: _____

Arbeitsplatz-Nr.: _____

Herbst 2014

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:	**16**

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

Bitte wenden!

Themenschwerpunkt A
(Datenbanksysteme)

Teilaufgabe 1:

Aufgabe 1:

Entwerfen Sie ein ER-Modell für die folgende Miniwelt, die sich mit Zügen, Verbindungen, Orten usw. befasst. Geben Sie Kardinalitäten in (*min, max*)-Notation an. Sie brauchen Schlüsselattribute nicht zu kennzeichnen.

- Ein *Zugtyp* hat eine Bezeichnung, eine Höchstgeschwindigkeit und eine Kapazität.
- Es gibt *Strecken*, die einen Start- und ein Zielort haben.
- Ein *Zug* hat einen Namen und ist von einem bestimmten Zugtyp. Ein Zug kann auf einer Strecke verkehren und jede Strecke wird von mindestens einem Zug bedient. Jeder Zug, der auf einer Strecke fährt, bekommt für diese Fahrt eine eigene Bezeichnung.
- *Bahnunternehmen* besitzen Züge und bieten Verbindungen auf bestimmten Strecken an.
- *Reisende* nutzen auf Strecken angebotene Züge. Der Preis für die Strecke mit dem Zug soll protokolliert werden. Reisende haben Namen. Ein Zug verkehrt nur auf einer Strecke, wenn mindestens 10 Reisende mitfahren.

Aufgabe 2:

Wir betrachten die Schüler-Datenbank mit den folgenden Tabellen:

- Schüler(Snr, Vorname, Nachname, Geburtsdatum, Klasse): *Persönliche Angaben zu den Schülern.*
- Abschlussnote(Snr, Fach, Note): *Notenspiegel der Schüler.*
- Raum(Rnr, Typ, Sitzplätze): *Typ und Anzahl der Sitzplätze der Räume.*
- Raumbelegung(Klasse, Fach, Rnr): *Raumbelegung nach Klassen und Fächern.*

Erstellen Sie in SQL folgende Anfragen:

a) Bestimmen Sie alle Räume und die Anzahl ihrer Sitzplätze, in denen die Klasse '7b' Unterricht hat.

b) Bestimmen Sie alle Paare von Schülern (jeweils gegeben durch die beiden Schülernummern), die in dieselbe Klasse gehen und am selben Tag Geburtstag haben.

c) Bestimmen Sie die Durchschnittsnote des Schülers mit der Nummer '1'.

d) Bestimmen Sie die Fächer, in denen der Schüler mit der Nummer '1' seine schlechtesten Noten hat.

Fortsetzung nächste Seite!

Augabe 3:

Wir erweitern die Datenbank aus Aufgabe 2 um die folgenden beiden Tabellen:

- Fach(Fach, Fachbetreuer).
- Klasse(Klasse, Klassleiter)

a) Geben Sie ein ER-Diagramm für die erweiterte Datenbank an, in dem möglichst viele Relationen als Relationships modelliert sind.

b) Geben Sie geeignete Create Table-Statements mit Primär- und Fremdschlüsselbedingungen zur Erzeugung der Tabellen Schüler, Abschlussnote und Fach an.

c) Fügen Sie mit Hilfe eines SQL-Statements ein Attribut Ausbildungsrichtung mit einem geeigneten Wertebereich zur Tabelle Schüler hinzu.

d) Löschen Sie alle Abschlussnoten von Schülern der Klasse '9a'.

Aufgabe 4:

Gegeben seien die funktionalen Abhängigkeiten

$$
\begin{aligned}
F = \{ \quad & B \rightarrow AC, && \text{(i)} \\
& BE \rightarrow C, && \text{(ii)} \\
& F \rightarrow AE, && \text{(iii)} \\
& D \rightarrow EF \quad \} && \text{(iv)}
\end{aligned}
$$

der Relation $R(A,B,C,D,E,F)$.

a) Berechnen Sie eine kanonische Überdeckung F_C von F. Geben Sie alle Zwischenschritte und die angewendeten Transformationen an.

b) Geben Sie alle Kandidatenschlüssel für das Relationenschema R (jeweils mit Begründung) sowie die Nichtschlüsselattribute an.

c) In welcher Normalform ist R? Geben Sie eine Begründung an.

Teilaufgabe 2:

Aufgabe 1:

1.1 Beantworten Sie folgende Fragen.

a) Welche Auswirkungen hat eine Transaktion, wenn sie mit Abort abgebrochen wird? Begründen Sie Ihre Antwort.
b) Grenzen Sie die Begriffe Datenbankverwaltungssystem und Datenbank voneinander ab.
c) Was ist das Ergebnis einer Projektion und was einer Selektion (= Restriktion) in der relationalen Algebra?
d) Was bedeutet die Abkürzung SQL ausgeschrieben? Worin unterscheidet sich diese Sprache von Programmiersprachen wie Java/C/Basic/... und welche Vorteile ergeben sich daraus?
e) Wozu dient der Primärschlüssel und welche Eigenschaften muss er erfüllen?

1.2 Geben Sie für jede der folgenden Aussagen an, ob sie richtig oder falsch ist und begründen Sie Ihre Antwort.

a) Das Anlegen von Views führt zu Redundanzen in der Datenbank.
b) Bei einer 1:1-Beziehung braucht nur eine der beiden Relationen einen Primärschlüssel.
c) In einer Relation sind die Tupel nicht zwingend in der Einfügereihenfolge abgelegt.
d) Eine Datenbankoperation muss die Datenbank in einem konsistenten Zustand hinterlassen.
e) Die SQL-Anweisung *select* gehört zur Data-Definition-Language (DDL).

Aufgabe 2:

a) E/R-Modell

Im Folgenden finden Sie die Beschreibung der Inventarverwaltungssoftware eines Unternehmens. Erstellen Sie zu dieser ein ER-Diagramm. Geben Sie dabei auch die Kardinalitäten an. Verwenden Sie nur die im Folgenden genannten Attribute als Schlüsselkandidaten:

Mitarbeiter, denen eine eindeutige Personalnummer zugeordnet ist, arbeiten in genau einer Abteilung, die firmenweit durch eine Abteilungsnummer identifiziert wird und an ein oder mehreren Standorten untergebracht ist. Jeder Standort hat eine Straße, eine Hausnummer, eine Postleitzahl sowie einen Ort und wird durch diese Adresse eindeutig gekennzeichnet. An den einzelnen Standorten kann es Räume geben, die durch eine Raumnummer bezeichnet werden, die am jeweiligen Standort eindeutig ist.
Jeder Mitarbeiter kann Gegenstände besitzen, für die er alleine verantwortlich ist. Alle Gegenstände haben eine eindeutige Inventarnummer und sind von einem bestimmten Inventartyp, der eine eindeutige Bezeichnung hat (z.B. Laptop Modell 500). Es gibt zwei Arten von Inventartypen: Bewegliches Inventar und unbewegliches Inventar. Zu beweglichen Inventartypen wird das Gewicht gespeichert. Unbewegliche Inventartypen haben eine Position, die aus der geographischen Breite und der geographischen Länge besteht.
Gegenstände können von einem Raum in einen anderen Raum umgezogen werden. Für Umzugswünsche kann eine Priorität vergeben werden, um festzulegen, wie dringend der Gegenstand am neuen Ort gebraucht wird. Es kann nur einen Umzugswunsch pro Gegenstand geben.

Fortsetzung nächste Seite!

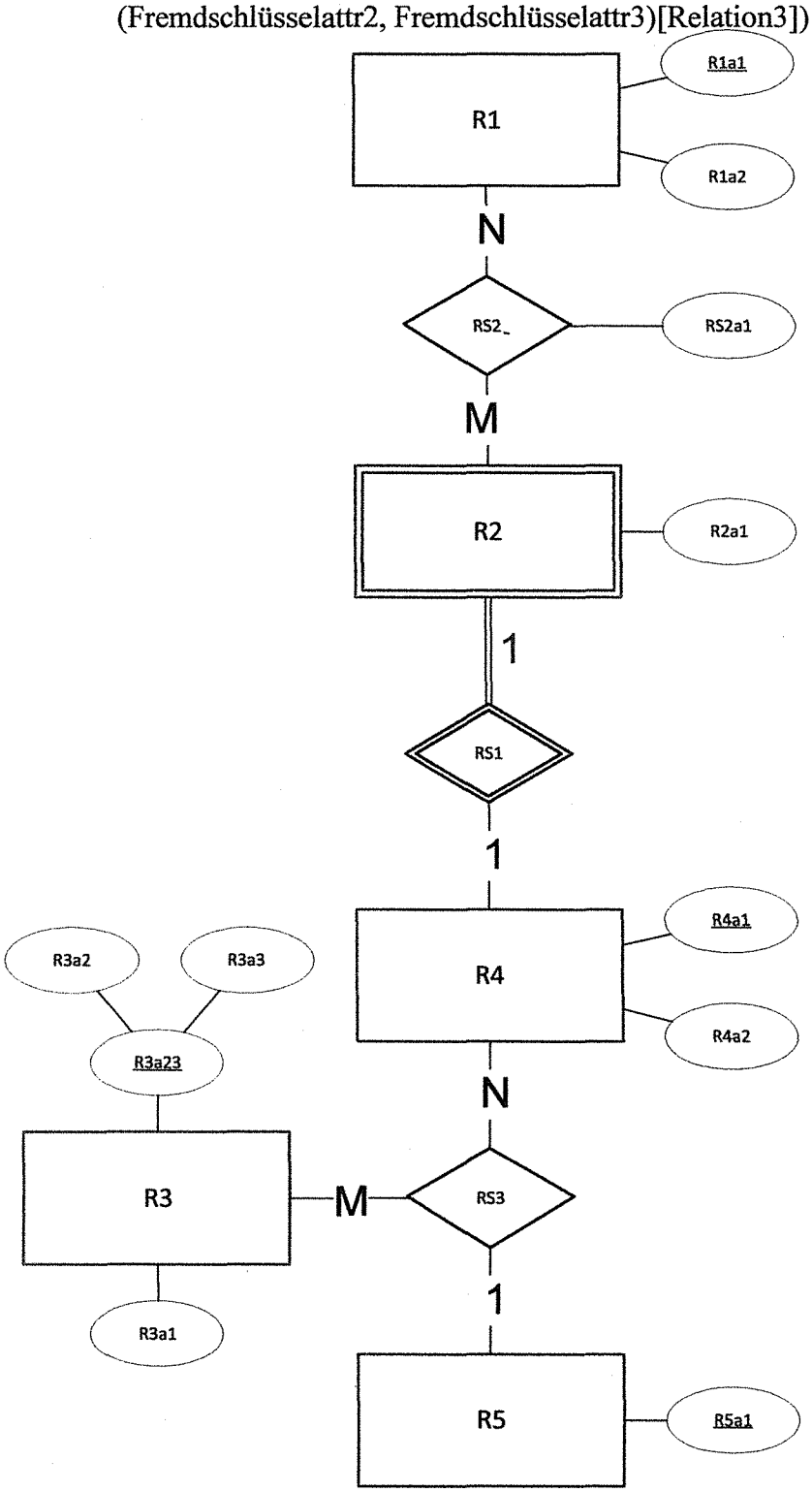
b) Abbildung auf Relationen

Entwerfen Sie zum untenstehenden E/R-Diagramm ein Relationenschema in dritter Normalform (3. NF) mit möglichst wenigen Relationen.

Verwenden Sie dabei folgende Notation:

Primärschlüssel werden durch Unterstreichen gekennzeichnet, Fremdschlüssel durch die Nennung der Relation, auf die sie verweisen, in eckigen Klammern hinter dem Fremdschlüsselattribut. Attribute zusammengesetzter Fremdschlüssel werden durch runde Klammern als zusammengehörig markiert.

Beispiel: Relation1 (Primärschlüssel, Attribut1, Attribut2, Fremdschlüsselattr1[Relation2], (Fremdschlüsselattr2, Fremdschlüsselattr3)[Relation3])



Fortsetzung nächste Seite!

Aufgabe 3:

Gegeben ist folgender Auszug aus einer Datenbank zur Verwaltung von Kochrezepten:
`Autor(ID, Vorname, Nachname)`
`Rezept(Rezeptnummer, Name, Verfasser[Autor], Beschreibung)`
`Zutat(Bezeichnung, Einkaufspreis)`
`Rezept_Zutat(Rezept[Rezept], Zutat[Zutat], Menge)`
Primärschlüssel sind unterstrichen, Fremdschlüssel dadurch gekennzeichnet, dass hinter ihnen in eckigen Klammern der Name der Relation angegeben ist, auf die sie sich beziehen.
Verwenden Sie bei der Formulierung der Anfragen nur standardkonformes SQL. Die Verwendung von Views ist erlaubt und empfohlen.

a) Schreiben Sie SQL-Statements, die die Tabellen zu den oben angegebenen Relationen anlegen. Sie können beliebige sinnvolle Datentypen wählen. Denken Sie auch an die Schlüssel und referentiellen Beziehungen.
b) Schreiben Sie ein SQL-Statement, das Vor- und Nachname des Autors mit der ID 42 ausgibt.
c) Schreiben Sie ein SQL-Statement, das alle Autoren, die weniger als 10 Rezepte verfasst haben, ausgibt, sortiert nach der Anzahl der von ihnen verfassten Rezepte. Ausgegeben werden soll die Anzahl der verfassten Rezepte, sowie Vor- und Nachname des Autors. Auch Autoren, die keine Rezepte verfasst haben, sollen in der Ausgabe enthalten sein.
d) Schreiben Sie eine SQL-Anfrage, die die Namen aller Rezepte zusammen mit der Anzahl verschiedener für sie benötigter Zutaten ausgibt, absteigend sortiert nach Anzahl der Zutaten.
e) Schreiben Sie eine SQL-Anfrage, die die Top-Ten der Rezepte mit den höchsten Zutatenkosten ausgibt. Die Kosten eines Rezepts berechnen sich als Summe über das Produkt aus Menge und Einkaufspreis aller Zutaten. Ausgegeben werden soll der Name des Rezepts, die Kosten und der Rang (1 = teuerstes Rezept bis 10), aufsteigend sortiert nach Rang. Gehen Sie davon aus, dass keine zwei Rezepte gleiche Kosten aufweisen.
   Verwenden Sie keine nicht-standardisierten Erweiterungen, wie `rownum` und ebenfalls nicht das Schlüsselwort `fetch`!

Aufgabe 4:

a) Von wann bis wann werden von einer Transaktion verwendete Sperren gehalten (dynamisches Sperren)? Begründen Sie, warum genau diese Zeitpunkte sinnvoll sind.
b) Bei der Transaktionsverarbeitung kann es zu Verklemmungen (Deadlocks) kommen. Erklären Sie anhand eines Beispiels, wie diese entstehen können. Erläutern Sie, wie ein Datenbankverwaltungssystem diese Situation erkennen und was es tun kann, um eine Verklemmung aufzulösen! Gehen Sie auch auf die entstehenden Konsequenzen ein.

Themenschwerpunkt B
(Betriebssysteme)

Teilaufgabe 1:

Aufgabe 1:

Beantworten Sie folgende Fragen zum Thema Prozesse.

1. Erläutern Sie kurz den Einfluss der Schedulingstrategie auf die Wahl eines geeigneten Prozesszustandsmodelles.
2. In den folgenden Fragen soll das 7–Zustands–Prozessmodell betrachtet werden.
   a) Skizzieren Sie das 7–Zustands–Prozessmodell. Kennzeichnen Sie alle Übergänge durch Pfeile. Beschriften Sie alle Pfeile mit der Aktion, die bei dem entsprechenden Übergang ausgeführt wird.
   b) Welche zusätzlichen Möglichkeiten ergeben sich im 7–Zustands-Prozessmodell, die im 5–Zustands-Prozessmodell nicht vorhanden sind? Geben Sie für jeden der zusätzlichen Zustandsübergänge ein Beispiel an.
3. Welche Problematik kann bei der Anwendung der nicht-präemptiven Schedulingstrategien First-Come First-Serve entstehen?
4. Erklären Sie kurz den Begriff Dispatching.
5. Erläutern und begründen Sie kurz. ob präemptive oder nicht-präemptive Schedulingstrategien weniger Verwaltungsoverhead erzeugen.

Fortsetzung nächste Seite!

Aufgabe 2:

In dieser Aufgabe sollen drei Scheduling-Strategien untersucht werden: die nicht-präemptive Strategie FCFS (First Come First Served), die nicht-präemptive Strategie SJF (Shortest Job First) und die präemptive Strategie RR (Round Robin). Dazu seien die folgenden Prozesse mit ihren Ankunftszeitpunkten und Rechenzeiten (in beliebigen Zeiteinheiten) gegeben.

Prozess	Ankunftszeitpunkt	Rechenzeit
P_1	0	3
P_2	1	5
P_3	2	4
P_4	3	2
P_5	5	3

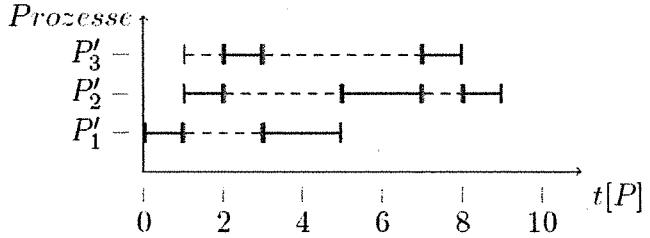
Gehen Sie davon aus, dass jeder Prozess sein Zeitquantum stets vollständig ausnutzt d.h. kein Prozess gibt den Prozessor freiwillig frei (Ausnahme: bei Prozessende).

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 ein beliebiges Scheduling der drei Prozesse P'_1, P'_2 und P'_3:



Bearbeiten Sie unter den gegebenen Voraussetzungen nun die folgenden Aufgaben:

1. Erstellen Sie entsprechend dem Beispiel ein Diagramm für die **nicht-präemptive Strategie FCFS**, das für die Prozesse P_1–P_5 angibt, wann welchem Prozess Rechenzeit zugeteilt wird und wann die Prozesse jeweils terminieren. Kennzeichnen Sie zudem für jeden Prozess seine Ankunftszeit.

2. Erstellen Sie entsprechend dem Beispiel ein Diagramm für die **nicht-präemptive Strategie SJF**, das für die Prozesse P_1–P_5 angibt, wann welchem Prozess Rechenzeit zugeteilt wird und wann die Prozesse jeweils terminieren. Kennzeichnen Sie zudem für jeden Prozess seine Ankunftszeit.

3. Erstellen Sie entsprechend dem Beispiel ein Diagramm für die **präemptive Strategie RR**, das für die Prozesse P_1–P_5 angibt, wann welchem Prozess Rechenzeit zugeteilt wird und wann die Prozesse jeweils terminieren. Die Dauer einer Zeitscheibe betrage 2 Zeiteinheiten. Gehen Sie davon aus, dass jeder Prozess die Dauer seiner Zeitscheibe stets vollständig ausnutzt, sofern er nicht terminiert. Terminiert ein Prozess aber vor Ablauf seiner Zeitscheibe, gibt er den Prozessor zum Zeitpunkt der Terminierung sofort frei. Trifft genau nach Ende einer Zeitscheibe ein neuer Prozess ein, so wird der neue Prozess vor dem gerade aktiven in die Warteschlange eingereiht.

Fortsetzung nächste Seite!

4. Berechnen Sie als Dezimalzahl mit einer Nachkommastelle die mittlere Verweil- und Wartezeit für die drei Verfahren FCFS, SJF und RR.

5. Warum ist der Einsatz von SJF in einem realen System in der Regel nicht realisierbar?

6. Welchen weiteren Nachteil hat SJF in Bezug auf die Verweildauer von Prozessen?

Fortsetzung nächste Seite!

Aufgabe 3:

Gegeben seien eine Menge an Seiten $N = \{ 0, 1, 2, 3, 4 \}$ und eine Menge der im Arbeitsspeicher zur Verfügung stehenden Seitenrahmen $F = \{ f_0, f_1, f_2 \}$. Auf die Seiten wird in der folgenden Reihenfolge zugegriffen:

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

Ein Seitenfehler liegt immer dann vor, wenn sich eine referenzierte Seite nicht im Arbeitsspeicher befindet. Der Arbeitsspeicher ist zu Beginn leer.

1. Bei der Paging-Strategie FIFO (First In, First Out) wird bei einem Seitenfehler diejenige Seite im Arbeitsspeicher ersetzt, deren Zeitpunkt der Zuordnung zu einem Seitenrahmen am längsten zurück liegt.

   Ermitteln Sie die Anzahl der Seitenfehler für die Seitenersetzungsstrategie FIFO, indem Sie alle Veränderungen im Speicher dokumentieren. Erstellen Sie dazu eine Tabelle mit dem Schema:

Zeit	Referenzierte Seite	f_0, t	f_1, t	f_2, t	Summe Seitenfehler
⋮	⋮	⋮	⋮	⋮	⋮

   Ordnen Sie dabei jeder referenzierten Seite den entsprechenden Seitenrahmen f_i ($i \in \{0, \ldots, 2\}$) zu und dokumentieren Sie den Zeitpunkt t der Zuordnung. Geben Sie zudem nach jedem Seitenzugriff die aktuelle Summe an Seitenfehlern an.

   Achtung: Bereits in den Hauptspeicher geladene Seiten dürfen nicht von einem Seitenrahmen in einen anderen verschoben werden!

2. Ermitteln Sie die Anzahl der Seitenfehler für die Seitenersetzungsstrategie LRU (Least Recently Used), indem Sie eine Tabelle mit dem bekannten Schema aus Teilaufgabe 1 erstellen. Es sollen alle Seitenzugriffe seit dem Laden einer Seite berücksichtigt werden. Ordnen Sie in der Tabelle wieder jede referenzierte Seite dem entsprechenden Seitenrahmen f_i ($i \in \{0, \ldots, 2\}$) zu und dokumentieren Sie den Zeitpunkt t eines Seitenzugriffs. Geben Sie zudem nach jedem Seitenzugriff die aktuelle Summe an Seitenfehlern an.

   Achtung: Bereits in den Hauptspeicher geladene Seiten dürfen nicht von einem Seitenrahmen in einen anderen verschoben werden!

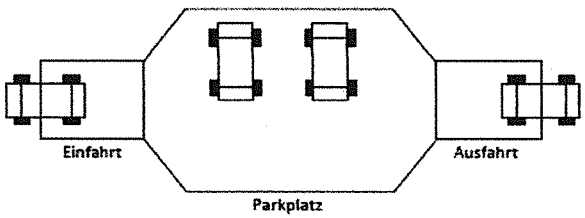
3. Nun soll theoretisch untersucht werden, wieviele Seitenfehler für das Beispiel aus Teilaufgabe 2 entstehen, wenn das System 1, 2, 3, 4 oder 5 Seitenrahmen besäße. Der zugehörige *Distance String* lautet ∞ ∞ ∞ ∞ 0 3 3 ∞ 3 2 4 4 1 4 3 2 . Der Rechenweg muss klar nachvollziehbar sein.

4. Welchen Effekt im Zusammenhang mit der Anzahl der Seitenrahmen eines Systems beschreibt die *Belady's Anomalie*? Kann diese Anomalie mit der Seitenersetzungsstrategie LRU (Least Recently Used) auftreten?

Fortsetzung nächste Seite!

Aufgabe 4:

In dieser Aufgabe soll ein Parkplatz simuliert werden, in dem aufgrund des Platzmangels die Anzahl der aktuell parkenden Autos zu jedem Zeitpunkt maximal `maxAutos` betragen darf. Die einzelnen parkenden Autos (die als Java Threads realisiert werden sollen) besitzen Tickets mit einer eindeutigen Ticketnummer. Sie können den Parkplatz durch eine Einfahrt befahren, indem sie ihr Ticket an einen dort angebrachten Ticketleser einer Schranke halten. Am anderen Ende des Parkplatzes können die parkenden Autos über die Ausfahrt wieder herausfahren, indem Sie ihr Ticket an den Ticketleser der dort installierten Schranke halten, welche anschließend das Passieren erlaubt. Der Parkplatz ist somit nur in eine Richtung passierbar. Die Schranken sind so konstruiert, dass immer nur genau ein Auto passieren kann, nachdem der Fahrer sein Ticket an den jeweiligen Ticketleser gehalten hat. Auf dem Parkplatz dürfen die Auto für eine beliebige Zeitspanne verweilen – somit können Autos den Parkplatz in einer anderen Reihenfolge verlassen, als sie ihn befahren haben. In folgender Abbildung ist ein solcher Parkplatz mit einer maximalen Anzahl an gleichzeitigen Besuchern von `maxAutos = 3` dargestellt, wobei sich gegenwärtig nur `anzahlBesucher = 2` Auto auf dem Parkplatz befinden. Ein Auto ist gerade dabei die Schranke an der Ausfahrt zu passieren und ein Auto wartet auf den Einlass an der Einfahrt.



Im Folgenden soll schrittweise eine Klasse `Parkplatz` implementiert werden. Die Beispielimplementierungen der Klassen `Auto` und `Simulation` sollen Ihnen verdeutlichen, wie die Klasse `Auto` verwendet werden kann:

Die Klasse `Simulation`:

```
1  public class Simulation {
2      private Parkplatz my_parkplatz;
3      private Auto[] autos;
4
5      public Simulation() {
6          my_parkplatz = new Parkplatz(3);
7          autos = new Auto[10];
8          for(int i = 0; i < autos.length; i++) {
9              autos[i] = new Auto(my_parkplatz, i);
10             autos[i].start();
11         }
12     }
13
14
15     public static void main(String[] args) {
16         new Simulation();
17     }
18 }
```

Die Klasse `Auto`:

```
1  import java.util.Random;
2
```

Fortsetzung nächste Seite!

```
3  public class Auto extends Thread {
4      private Parkplatz my_parkplatz;
5      private int ticket_nummer;
6      private Random generator;
7
8      public Auto(Parkplatz parkplatz, int ticket_nummer) {
9          this.my_parkplatz = parkplatz;
10         this.ticket_nummer = ticket_nummer;
11         generator = new Random();
12     }
13
14     public void run() {
15             try {
16                 Thread.sleep(generator.nextInt(2000) + 500);
17                 my_parkplatz.einfahren(ticket_nummer);
18                 Thread.sleep(generator.nextInt(2000) + 500);
19                 my_parkplatz.ausfahren(ticket_nummer);
20             }
21             catch(InterruptedException ie) {}
22     }
23 }
```

Der Klasse Parkplatz soll folgendes Interface zugrunde liegen:

```
1  interface Parkplatz {
2      // Konstruktor
3      public Parkplatz(int maxBesucher);
4
5      // Methoden
6      public synchronized void einfahren(int ticket_nummer);
7      public synchronized void ausfahren(int ticket_nummer);
8  }
```

1. Was sind die kritischen Bereiche bei diesem Problem?

2. Überlegen Sie sich die notwendigen Attribute der Klasse `Parkplatz` und implementieren Sie deren Konstruktor. *Kommentieren Sie Ihre Lösung ausführlich!*

3. Implementieren Sie die Methode `einfahren(int ticket_nummer)`, welche das Passieren der Schranke an der Einfahrt des Parkplatzes modelliert, sowie die Methode `ausfahren(int ticket_nummer)`, welche im Gegenzug das Verlassen des Parkplatzes über die Schranke an der Ausfahrt modelliert. Beachten Sie bei der Implementierung, dass alle oben genannten Bedingungen eingehalten werden. Schreiben Sie die Methode `einfahren(int ticket_nummer)` so, dass sie die eindeutige Ticketnummer des einfahrenden Autos sowie die Anzahl der sich gerade auf dem Parkplatz befindlichen Autos (inklusive des einfahrenden Autos) auf der Konsole ausgibt. Schreiben sie analog die Methode `ausfahren(int ticket_nummer)` so, dass sie die Ticketnummer des Autos, das gerade die Schranke am Ausgang passiert sowie die Anzahl der sich gerade auf dem Parkplatz befindlichen Autos (exklusive des Autos das den Parkplatz gerade verlässt) auf der Konsole ausgibt. Sie können davon ausgehen, dass die Methoden `einfahren(int ticket_nummer)` bzw. `ausfahren(int ticket_nummer)` immer in einer sinnvollen Reihenfolge aufgerufen werden (siehe Beispielimplementierung der Klasse Auto). *Kommentieren Sie Ihre Lösung ausführlich!*

4. Erläutern Sie kurz, warum Ihre Lösung des wechselseitigen Ausschlusses die Bedingung der *Mutual Exclusion* erfüllt.

Teilaufgabe 2:

Aufgabe 1:

Eine der wesentlichen Aufgaben eines Betriebssystems ist es, die vorhandenen Hardware- und Software-Betriebsmittel zu verwalten und für einen verklemmungsfreien Ablauf der einzelnen Prozesse zu sorgen.

(a) In welche Klassen können Betriebsmittel eingeteilt werden? Geben Sie für jede der genannten Klassen ein Beispiel an.

(b) 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?

(c) Welche drei grundsätzlichen Verfahren gibt es, um mit der Deadlock-Problematik umzugehen? Beschreiben Sie jedes Verfahren. Wie ist jeweils die grundsätzliche Vorgehensweise? Nennen Sie den Namen eines Algorithmus, der bei einem dieser Verfahren zum Einsatz kommt.

(d) Fünf Philosophen sitzen an einem runden Tisch, in dessen Mitte ein immer voller Teller mit Nudeln steht. Rechts und links von jedem Philosophen liegt jeweils eine Gabel. Zum Essen benötigt ein Philosoph zwei Gabeln. Die Philosophen sind nur mit zwei Dingen beschäftigt: Denken und Essen. Ist einer hungrig, so greift er zuerst nach der linken und dann nach der rechten Gabel und fängt an zu essen. Ist eine Gabel belegt, so wartet er, bis sie wieder frei ist. Hat der Philosoph sich satt gegessen, so legt er die Gabeln wieder an ihren Platz und fährt fort mit Denken.

Betrachten Sie unten stehenden Pseudocode, der einen einzelnen Philosophen simuliert. Erläutern Sie, wie es in dieser Lösung zu einem Deadlock kommen könnte, indem Sie die Bedingungen aus Aufgabenteil (b) nachweisen. Zu welcher Klasse nach Aufgabenteil (a) gehören die vorhandenen Betriebsmittel? Neben einem Deadlock könnte noch ein weiteres Problem auftauchen. Erläutern Sie kurz welches.

```
N=5; Procedure phil(i:INTEGER)
     WHILE (TRUE) DO
          think;                        //Philosoph denkt nach
          take_fork(i);                 //nimmt die linke Gabel
          take_fork((i+1) mod N);       //nimmt die rechte Gabel
          eat;                          //isst
          put_fork(i);                  //legt die linke Gabel zurück
          put_fork((i+1) mod N);        //legt die rechte Gabel zurück
     END;
END phil;
```

Fortsetzung nächste Seite!

Aufgabe 2:

Zu einem bestimmten Zeitpunkt T kommen folgende Jobs in der gegebenen Reihenfolge in einer Rechenanlage mit einer CPU zur Bearbeitung an. Der zusätzliche Aufwand beim Umschalten zwischen den Jobs soll vernachlässigt werden.

Job	Bearbeitungszeit
1	7
2	3
3	6
4	10
5	1
6	2

(a) Beschreiben Sie unter Verwendung von Gantt-Diagrammen (s. u.) die Bearbeitung der Jobs für die Schedulingverfahren *First-come-first-serve*d (FCFS), *Shortest-Job-First* (SJF) und *Round-Robin* mit Zeitquantum=3 (RR).

Beispiel für ein Gantt-Diagramm:

Zeit	1	2	3	4	5	6	8	9
Prozessor	Job#1			Job#2	Job#3			leer

(b) Geben Sie für alle drei Schedulingverfahren den „Turnaround" (Gesamtzeit im System) eines jeden Jobs an.

(c) Geben Sie für alle drei Schedulingverfahren die Anfangswartezeiten, d.h. die Zeit eines jeden Jobs an, die er warten muss, bis er zum ersten Mal die CPU zugeteilt bekommt.

(d) Welches Schedulingverfahren besitzt (angewandt auf das obige Beispiel und über alle Jobs gerechnet) die geringste durchschnittliche Gesamtwartezeit, d.h. bei welchem Verfahren ist die Zeit der Inaktivität der Prozesse durchschnittlich am geringsten?

Fortsetzung nächste Seite!

Aufgabe 3:

(a) Beschreiben Sie knapp die Speicherverwaltungsstrategien First-Fit, Best-Fit und Worst-Fit. Nennen Sie zu jeder Strategie mindestens einen Vorteil und einen Nachteil.

(b) Gegeben sei zusätzlich noch Fragment-Fit: die zu speichernden Daten können in Fragmente aufgeteilt und auf verschiedene Speicherbereiche verteilt werden. Dabei wird der zuerst gefundene Speicherblock gefüllt und ggf. überschüssige Daten in die nächsten freien Speicherbereiche geschrieben. Nennen die Vor- und Nachteile dieses Verfahrens gegenüber den oben genannten First-Fit, Best-Fit und Worst-Fit.

(c) Gegeben sei die Liste mit den folgenden, freien Speicherblöcken (in genau dieser Reihenfolge, wobei 1MB = 1024KB). Diese Blöcke liegen nicht nebeneinander im RAM.

800KB 1MB 260KB 120KB 5MB

Es werden nun an das Betriebssystem in folgender Reihenfolge Speicheranforderungen gestellt:

1000KB 550KB 90KB 160KB 4MB 900KB

Wie teilt jede der vier Strategien First-Fit, Best-Fit, Worst-Fit und Fragment-Fit den Speicher zu? Zur vollständigen Lösung gehört die Liste aller freien Blöcke nach jeder erfüllten Anforderung.

(d) Beantworten Sie bezogen auf das obige Beispiel die folgenden Fragen mit knapper Begründung: Welches Verfahren schneidet bzgl. Erfüllung der Anfragen am schlechtesten ab? Welches Verfahren arbeitet bzgl. der Speicherfragmentierung am effizientesten?

Fortsetzung nächste Seite!

Aufgabe 4:

(a) Definieren Sie den Begriff „Prozess“ in Betriebssystemen.

(b) Übertragen Sie die gegebene Grafik auf Ihr Blatt, fügen Sie die Prozesszustände „rechnend“, „blockiert“ und „rechenbereit“ ein und beschriften Sie die Übergänge entsprechend der Ereignisse, durch die sie ausgelöst werden.