Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer

Kennzahl: _____

Kennwort: _____

Arbeitsplatz-Nr.: _____

Herbst 2016

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: **26**

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 ausgewählten Teilaufgaben anzugeben (z. B. A2, B1)!

Bitte wenden!

Themenschwerpunkt A
(Datenbanksysteme)

Teilaufgabe 1

1. ER-Modellierung

Gegeben seien folgende Informationen:

- Krankenhäuser bestehen aus Stationen. Zu jedem Krankenhaus ist dessen Adresse (wodurch es identifiziert werden kann) und die Anzahl an Betten bekannt. Stationen besitzen einen Namen, der nur innerhalb eines Krankenhauses eindeutig ist.
- Stationen bestehen wiederum aus Zimmern, welche nummeriert sind. Eine solche Nummer ist innerhalb einer Station eindeutig.
- Jedes Krankenhaus beschäftigt Personal. Dabei wird festgehalten, welches Gehalt bezahlt wird. Personal kann in verschiedenen Krankenhäusern beschäftigt sein.
- Personal ist durch eine Personal-Nummer gekennzeichnet und kann unter anderem in Ärzte und Krankenpfleger unterteilt werden. Zu einem Arzt ist sein Fachbereich bekannt und von welchen Krankenpflegern er Vorgesetzter ist. Kein Krankenpfleger kann zugleich Arzt sein, aber durchaus mehrere vorgesetzte Ärzte haben.
- Eine Station kann von mehreren Ärzten gleitet werden. Ein Arzt kann ebenso mehrere Stationen leiten. Außerdem ist bekannt, ob und in welchem Zimmer ein Arzt sein Büro hat. Kein Arzt muss sich sein Büro mit einem anderen Arzt teilen.
- Personal arbeitet auf Stationen in Schichten. Eine Schicht kann über Datum und Zeitraum eindeutig identifiziert werden. Eine Person kann in einer Schicht auf nur einer Station arbeiten.

a) Erstellen Sie für das oben gegebene Szenario ein geeignetes ER-Diagramm.
Verwenden Sie dabei – wenn angebracht – das Prinzip der Spezialisierung. Kennzeichnen Sie die Primärschlüssel der Entity-Typen, totale Teilnahmen und schwache Entity-Typen.
Zeichnen Sie die Funktionalitäten der Relationship-Typen in das Diagramm ein.

b) Überführen Sie Ihr in Aufgabe a) erstelltes Modell in ein verfeinertes relationales Schema.
Kennzeichnen Sie die Schlüssel durch Unterstreichen. Datentypen müssen nicht angegeben werden.

Fortsetzung nächste Seite!

2. SQL und relationale Algebra

Gegeben sei der folgende Ausschnitt aus dem Schema einer Schulverwaltung:

Person : {[
  ID : INTEGER,
  Name : VARCHAR(255),
  Wohnort : VARCHAR(255),
  Typ : CHAR(1)
]}

Klasse : {[
  Klassenbezeichnung : VARCHAR(20),
  Schuljahr : INTEGER,
  Klassenlehrer : INTEGER
]}

Unterricht : {[
  Klassenbezeichnung : VARCHAR(20),
  Schuljahr : INTEGER,
  Lehrer : INTEGER,
  Fach : VARCHAR(100)
]}

Klassenverband : {[
  Schüler : INTEGER,
  Klassenbezeichnung : VARCHAR(20),
  Schuljahr : INTEGER
]}

Hierbei enthält die Tabelle *Person* Informationen über Lehrer (Typ 'L') und Schüler (Typ 'S'); andere Werte für Typ sind nicht zulässig. *Klasse* beschreibt die Klassen, die in jedem Schuljahr gebildet wurden, zusammen mit ihrem Klassenlehrer. In *Unterricht* wird abgelegt, welcher Lehrer welches Fach in welcher Klasse unterrichtet; es ist möglich, dass derselbe Lehrer mehr als ein Fach in einer Klasse unterrichtet. *Klassenverband* beschreibt die Zuordnung der Schüler zu den Klassen.

a) Schreiben Sie eine SQL-Anweisung, die die Tabelle *Unterricht* mit allen ihren Constraints (einschließlich Fremdschlüsselconstraints) anlegt.

b) Definieren Sie ein geeignetes Constraint, das sicherstellt, dass nur zulässige Werte im Attribut Typ der (bereits angelegten) Tabelle *Person* eingefügt werden können.

c) Schreiben Sie eine SQL-Anweisung, die die Bezeichnung der Klassen bestimmt, die im Schuljahr 2015 die meisten Schüler haben.

d) Schreiben Sie eine SQL-Anweisung, die die Namen aller Lehrer bestimmt, die nur Schüler aus ihrem Wohnort unterrichtet haben.

e) Schreiben Sie eine SQL-Anweisung, die die Namen aller Schüler bestimmt, die immer den gleichen Klassenlehrer hatten.

f) Schreiben Sie eine SQL-Anweisung, die alle Paare von Schülern bestimmt, die mindestens einmal in der gleichen Klasse waren. Es genügt dabei, wenn Sie die ID der Schüler bestimmen.

g) Formulieren Sie eine Anfrage in der relationalen Algebra, die die ID aller Schüler bestimmt, die mindestens einmal von 'Ludwig Lehrer' unterrichtet wurden.

h) Formulieren Sie eine Anfrage in der relationalen Algebra, die Namen und ID der Schüler bestimmt, die von allen Lehrern unterrichtet wurden.

Beachten Sie bei der Formulierung der SQL-Anfragen, dass die Ergebnisrelationen keine Duplikate enthalten dürfen. Sie dürfen geeignete Views definieren.

Fortsetzung nächste Seite!

3. Entwurfstheorie

Gegeben sei folgendes relationale Schema R in erster Normalform:

$$R : \{[A,B,C,D,E,F]\}$$

Für R gelte folgende Menge FD funktionaler Abhängigkeiten:
$FD = \{$

$$\begin{aligned} AC &\rightarrow B, \\ DEF &\rightarrow BC, \\ F &\rightarrow AB, \\ D &\rightarrow F, \\ BC &\rightarrow E \end{aligned}$$

$\}$

a) Bestimmen Sie alle Kandidatenschlüssel von R mit FD.
Hinweis: Die Angabe von Attributmengen, die keine Kandidatenschlüssel sind, führt zu Abzügen.

b) Prüfen Sie, ob R mit FD in 2NF bzw. 3NF ist.

c) Bestimmen Sie mit folgenden Schritten eine kanonische Überdeckung FD_C von FD:
   i. Führen Sie eine Linksreduktion von FD durch. Geben Sie die Menge funktionaler Abhängigkeiten nach der Linksreduktion an (FD_L).
   ii. Führen Sie eine Rechtsreduktion des Ergebnisses der Linksreduktion (FD_L) durch. Geben Sie die Menge funktionaler Abhängigkeiten nach der Rechtsreduktion an (FD_R).
   iii. Bestimmen Sie eine kanonische Überdeckung FD_C von FD auf Basis des Ergebnisses der Rechtsreduktion (FD_R).

d) Zerlegen Sie R mit FD_C mithilfe des *Synthesealgorithmus* in 3NF.

e) Prüfen Sie *für alle Relationen* der Zerlegung aus d), ob sie jeweils in BCNF sind.

Fortsetzung nächste Seite!

4. Transaktionen

a) Erläutern Sie kurz die Konzepte *Atomarität* und *Isolation.*
b) Betrachten Sie den folgenden Schedule S:

T_1	T_2	T_3
$r_1(x)$		
	$r_2(x)$	
	$r_2(y)$	
	$w_2(x)$	
	c_2	
$r_1(y)$		
		$r_3(y)$
		$w_3(y)$
		c_3
$w_1(y)$		
c_1		

Geben Sie den Ausgabeschedule (einschließlich der Operationen zur Sperranforderung und -freigabe) im strikten Zweiphasen-Sperrprotokoll für den obigen Eingabeschedule S an.
Hinweis: Sollte während der Ausführung ein Deadlock auftreten, führen Sie ein Rollback einer der beteiligten Transaktionen durch.

Fortsetzung nächste Seite!

Teilaufgabe 2

1. Konzeptioneller Entwurf

Im Folgenden ist die Beschreibung einer Fotoverwaltung gegeben. Erstellen Sie zu dieser ein ER-Diagramm. Kennzeichnen Sie die Primärschlüssel durch passendes Unterstreichen und geben Sie die Kardinalitäten in Min-Max-Notation an. Modellieren Sie keine Attribute oder Entitytypen, die nicht aus dem Text hervorgehen.
Es werden Fotos verwaltet, die über eine eindeutige ID identifiziert werden und einen Aufnahmezeitpunkt sowie eine Beschreibung besitzen. Den Fotos können Stichworte zugeordnet werden, die einen eindeutigen Namen haben. Stichworte sind hierarchisch organisiert, d.h. ein Stichwort kann maximal einem anderen Stichwort untergeordnet sein. Jedes Foto wurde von genau einem Fotografen aufgenommen und es können Models auf ihm abgebildet sein. Zu Fotografen und Models werden Name, Vorname, Postleitzahl und Wohnort gespeichert. Zu Fotografen wird zusätzlich ihre Mitgliedsnummer im Verband deutscher Fotografen abgelegt, durch die sie eindeutig identifiziert werden können. Models gehören genau einer Agentur an und haben eine Mitgliedsnummer, die innerhalb ihrer Agentur eindeutig ist. Agenturen haben einen eindeutigen Namen sowie eine E-Mail-Adresse. Ein Fotograf kann auch gleichzeitig Model sein. Models können der Veröffentlichung von bestimmten Fotos in bestimmten Medien zustimmen. Zu jedem Medium werden ein eindeutiger Name sowie eine Beschreibung abgelegt. Zu einer Zustimmung wird das Datum gespeichert, an dem sie erfolgt ist.

Fortsetzung nächste Seite!

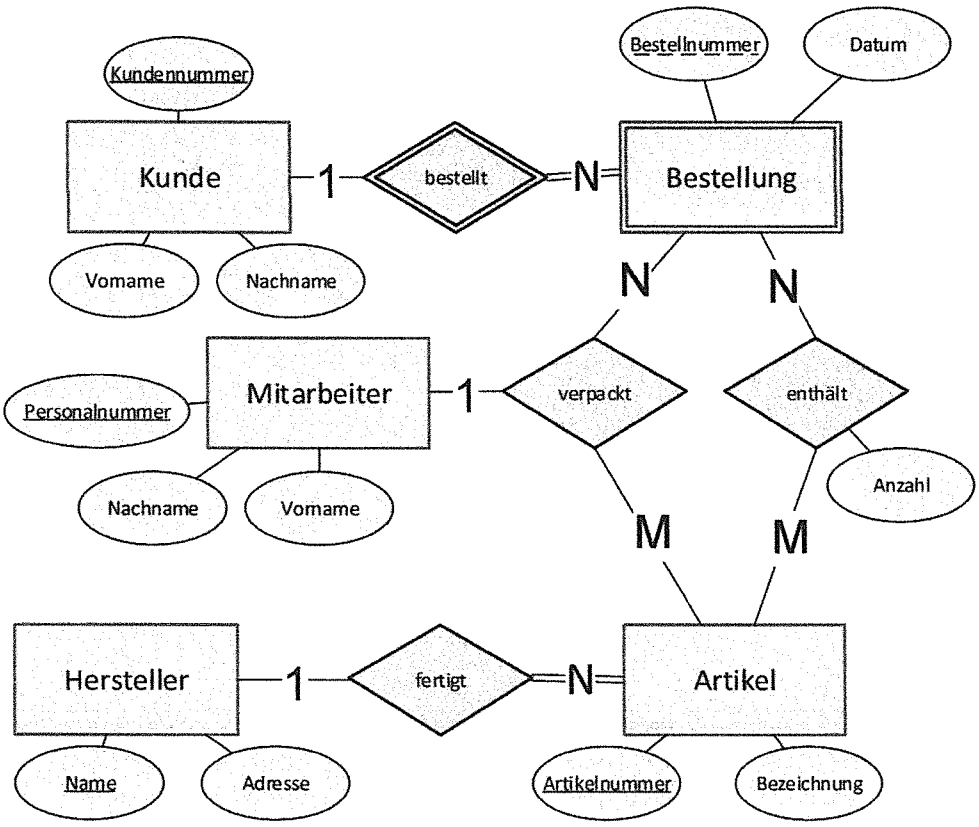
2. Relationenschema

Entwerfen Sie zum untenstehenden ER-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, Fremdschlüssel[Relation2])



Fortsetzung nächste Seite!

3. Normalformen

a) Gegeben ist eine vollständige Extension der Relation Bestellungen mit dem zusammengesetzten Primärschlüssel {Kundennummer, Bestellnummer}:

Kundennummer	**Bestellnummer**	Kundennachname	Bestelldatum	Lieferdatum
1	1657	Frank	13.02.2015	15.02.2015
2	1306	Weizenbaum	14.01.2015	18.01.2015
3	2369	Weizenbaum	09.08.2014	10.01.2015
2	9847	Weizenbaum	21.11.2013	25.11.2013
5	2569	Schmitt	05.09.2015	05.09.2015
7	3457	Nürnberger	07.10.2012	12.12.2012
8	4468	Haberkorn	31.08.2015	01.10.2015

Geben Sie an, in welcher Normalform sich die Relation Bestellungen befindet. Begründen Sie weiterhin in zwei bis drei Sätzen, dass alle Bedingungen dieser Normalform erfüllt sind und dass nicht alle Bedingungen der nächsthöheren Normalform erfüllt sind.

b) Nennen Sie die Bezeichnungen der drei Anomalien, zu denen nicht normalisierte Relationenschemata führen können, und erläutern Sie die Anomalien in jeweils ein bis zwei Sätzen. Sie können dazu Beispiele aus der in der vorhergehenden Teilaufgabe gegebenen Relation nutzen.

Fortsetzung nächste Seite!

4. SQL

Gegeben sind folgende Relationen aus einer Personalverwaltung:

```
Mitarbeiter (MitarbeiterID, Vorname, Nachname, Vorgesetzter[Mitarbeiter],
  AbteilungsID[Abteilung], Telefonnummer, Gehalt)
Abteilung (AbteilungsID, Bezeichnung)
```

a) Schreiben Sie eine SQL-Anfrage, die Vor- und Nachnamen der Mitarbeiter aller Abteilungen mit Bezeichnung „Buchhaltung" ausgibt, absteigend sortiert nach MitarbeiterID.

b) Schreiben Sie eine SQL-Anfrage, die die Nachnamen aller Mitarbeiter zusammen mit dem Nachnamen ihres jeweiligen direkten Vorgesetzten ausgibt. Mitarbeiter ohne Vorgesetzten sollen in der Ausgabe ebenfalls enthalten sein. In diesem Fall soll der Nachname des Vorgesetzten NULL sein.

c) Schreiben Sie eine SQL-Anfrage, die die 10 Abteilungen ausgibt, deren Mitarbeiter das höchste Durchschnittsgehalt haben. Ausgegeben werden sollen der Rang (1 = höchstes Durchschnittsgehalt bis 10 = niedrigstes Durchschnittsgehalt), die Bezeichnung sowie das Durchschnittsgehalt der Abteilung.
Gehen Sie davon aus, dass es keine zwei Abteilungen mit gleichem Durchschnittsgehalt gibt. Sie können der Übersichtlichkeit halber Views oder With-Anweisungen verwenden. Verwenden Sie jedoch **keine** datenbanksystemspezifischen Erweiterungen wie limit oder rownum.

d) Schreiben Sie eine SQL-Anfrage, die das Gehalt aller Mitarbeiter aus der Abteilung mit AbteilungsID 42 um 5% erhöht.

e) Alle Abteilungen mit der Bezeichnung „Qualitätskontrolle" sollen zusammen mit den Datensätzen ihrer Mitarbeiter gelöscht werden. ON DELETE CASCADE ist für keine der Tabellen gesetzt. Schreiben Sie die zum Löschen notwendigen SQL-Anfragen.

f) Alle Mitarbeiter sollen mit SQL-Anfragen nach den Telefonnummern anderer Mitarbeiter suchen können. Sie dürfen jedoch das Gehalt der Mitarbeiter nicht sehen können. Erläutern Sie in zwei bis drei Sätzen eine Möglichkeit, wie dies in einem Datenbanksystem realisiert werden kann, ohne die gegebenen Relationen, die als Tabellen abgelegt sind, zu verändern. Sie brauchen hierzu keinen SQL-Code schreiben.

Fortsetzung nächste Seite!

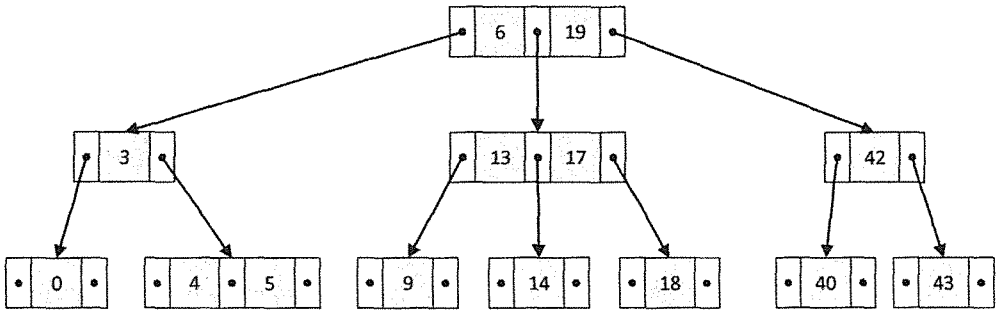
5. Physische Datenorganisation

a) Erläutern Sie die wesentliche Eigenschaft eines Tupel-Identifikators (TID) in ein bis zwei Sätzen.

b) Fügen Sie in einen anfangs leeren B-Baum mit k = 1 (maximal 2 Schlüsselwerte pro Knoten) die im Folgenden gegebenen Schlüsselwerte der Reihe nach ein. Zeichnen Sie den Endzustand des Baums nach jedem Einfügevorgang. Falls Sie Zwischenschritte zeichnen, kennzeichnen Sie die sieben Endzustände deutlich.

3, 7, 13, 11, 9, 10, 8

c) Gegeben ist der folgende B-Baum:



Die folgenden Teilaufgaben sind voneinander unabhängig.

i. Löschen Sie aus dem gegebenen B-Baum den Schlüssel 3 und zeichnen Sie den Endzustand des Baums nach dem Löschvorgang. Falls Sie Zwischenschritte zeichnen, kennzeichnen Sie den Endzustand deutlich.

ii. Löschen Sie aus dem (originalen) gegebenen B-Baum den Schlüssel 17 und zeichnen Sie den Endzustand des Baums nach dem Löschvorgang. Falls Sie Zwischenschritte zeichnen, kennzeichnen Sie den Endzustand deutlich.

iii. Löschen Sie aus dem (originalen) gegebenen B-Baum den Schlüssel 43 und zeichnen Sie den Endzustand des Baums nach dem Löschvorgang. Falls Sie Zwischenschritte zeichnen, kennzeichnen Sie den Endzustand deutlich.

6. Transaktionen

a) Nennen Sie in einem Satz den Unterschied zwischen dem strengen Zwei-Phasen-Sperrprotokoll und dem Zwei-Phasen-Sperrprotokoll.

b) Beim unkontrollierten Mehrbenutzerbetrieb kann es zum Problem der verlorengegangenen Änderungen (lost update) kommen. Erläutern Sie anhand eines Beispiels in drei bis vier Sätzen, wie dieses Problem auftreten kann und welche Folgen sich daraus ergeben.

Themenschwerpunkt B
(Betriebssysteme)

Teilaufgabe 1

Aufgabe 1:

a) Was versteht man unter einem Prozess und was ist der Unterschied zu einem Programm?

b) Das Betriebssystem muss in Zusammenhang mit einem Prozess Ressourcen verwalten. Welche Ressourcen sind minimal erforderlich, um einen Schutz gegenüber Operationen von anderen Prozessen (hierbei geht es nicht um den Schutz gegenüber anderen Benutzern) zu gewährleisten und Operationen auf Dateien zu ermöglichen?

c) Ressourcen und weitere Daten zu einem Prozess werden vom Betriebssystem in der Prozesstabelle verwaltet. Nennen Sie 5 weitere typische Einträge und erläutern Sie, zu welchem Zweck diese Daten vom Betriebssystem benötigt werden.

d) Erläutern Sie den Unterschied zwischen den Konzepten *Prozess* und *Thread*. Nennen Sie die unterschiedlichen Arten von Threads und erläutern Sie die Unterschiede und Vor- und Nachteile der Thread-Arten.

e) Welche Operationen bietet die Systemschnittstelle von UNIX-/Linux-Systemen zur Erzeugung von Prozessen und zum Laden von Programmen? Beschreiben Sie die Funktionsweise der einzelnen Operationen und was im Betriebssystem dabei jeweils abläuft.

f) Skizzieren Sie, wie eine Shell in einem UNIX- oder Linux-System funktioniert (vom Einlesen eines Kommandos über das Ausführen bis zur Ausgabe des nächsten Prompt-Symbols). Wie geht die Shell in diesem Zusammenhang vor, um Kommandos im Vordergrund bzw. Hintergrund auszuführen?

Was tut die Shell dabei, um z.B. die Standardeingabe des auszuführenden Kommandos auf eine Datei umzuleiten (wie bei `Kommando <Datei1`)?

Fortsetzung nächste Seite!

Aufgabe 2:

a) Was versteht man in Betriebssystemen unter einem Semaphor? Beschreiben Sie die elementaren Operationen auf Semaphoren im Detail.

b) Man kann Semaphore sowohl für den gegenseitigen Ausschluss als auch zur Synchronisierung von mehreren Abläufen (einen Ablauf vor Ausführung der Operation B verzögern, bis ein anderer Ablauf einen bestimmten Punkt - d. h. Operation A wurde ausgeführt - erreicht hat) einsetzen. Beschreiben Sie jeweils eine typische Situation von solchen Anwendungsfällen und skizzieren Sie den Einsatz von Semaphoren (unter Angabe der Semaphor-Werte und -Operationen) in diesen Fällen in einer programmiersprachenähnlichen Form.

c) Was versteht man unter einem verdrängenden Scheduling-Verfahren mit dynamischen Prioritäten?

d) Beim Einsatz von Semaphoren bei mehreren Prozessen und verdrängendem Scheduling mit dynamischen Prioritäten kann es zu dem Phänomen der Prioritätsinversion kommen. Was versteht man darunter? Skizzieren Sie in einem Diagramm mit drei Prozessen eine Situation, in der das Phänomen auftritt. Was könnte man zur Vermeidung des Phänomens tun?

Fortsetzung nächste Seite!

Aufgabe 3:

a) Die Dateisysteme von UNIX-/Linux-Systemen (z. B. EXT2 oder EXT3) und Windows (z. B. NTFS) weisen eine Reihe von gemeinsamen Konzepten auf, unterscheiden sich aber auch in vielen Details. Ein gemeinsames Konzept sind die hierarchisch aufbaubaren Dateikataloge. Beschreiben Sie die Funktion eines Dateikatalogs (*Directory*), wie eine baumförmige Hierarchie aufgebaut werden kann und wie schließlich in solch einem System Dateien abgelegt, benannt und anhand des Namens gefunden werden können.

b) Die Verwaltungsdatenstruktur zu einer Datei in UNIX/Linux ist der sog. *Inode*, das Pendant in Windows-NTFS ist ein Eintrag in der Master-File-Table (MFT).
Vergleichen Sie die Konzepte *Inode* und *MFT-Eintrag*. Welche Daten werden darin typischerweise gespeichert? (Nennen Sie 5 Beispiele und erläutern Sie, wozu diese Daten jeweils benötigt werden). Beschreiben Sie zwei Vorteile des NTFS-Konzepts. Was versteht man in diesem Zusammenhang unter dem Konzept eines Datenstroms (*Streams*)?

c) Wie funktioniert die Verwaltung der Datenblöcke einer Datei in einem FAT-Dateisystem? Geben Sie die Belegung der FAT für drei Dateien an, wobei
Datei 1 in den Blöcken 8, 1, 5, 9;
Datei 2 in den Blöcken 12, 2, 11, 4, 7;
Datei 3 in den Blöcken 3, 6, 10;
gespeichert ist.

d) Moderne Dateisysteme werden typischerweise als Log-basierte (oder *Journaling*) Dateisysteme realisiert. Beschreiben Sie die grundlegende Funktionsweise eines solchen Dateisystems und welche Vorteile es mit sich bringt.

Fortsetzung nächste Seite!

Teilaufgabe 2

Aufgabe 1: Prozesse

1. Erläutern Sie kurz den Unterschied zwischen Multiprogramming und Multiprocessing.
2. Worin unterscheiden sich die Aufgaben des Schedulers und des Dispatchers?
3. Erläutern Sie kurz, ob der Einsatz des 5-Zustands-Prozess-Modells mit einem nicht-präemptiven Scheduling-Verfahren empfehlenswert ist.
4. 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.

Fortsetzung nächste Seite!

Aufgabe 2: Scheduling-Strategien

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

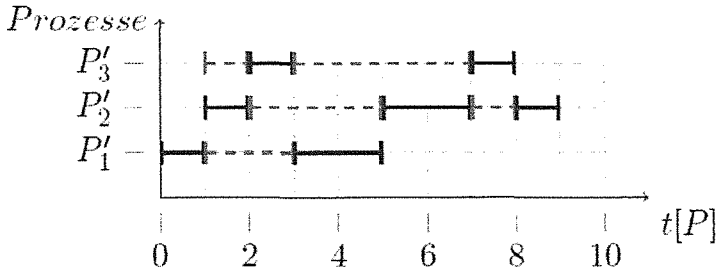
Prozess	Ankunftszeitpunkt	Rechenzeit
P_1	0	2
P_2	0	2
P_3	2	2
P_4	4	3
P_5	5	4
P_6	6	3
P_7	8	1

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:



Fortsetzung nächste Seite!

Bearbeiten Sie unter den gegebenen Voraussetzungen nun die folgenden Aufgaben:

1. Erstellen Sie entsprechend dem Beispiel ein Diagramm für die **nicht-präemptive Strategie SJF** das für die Prozesse P_1-P_7 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 **präemptive Strategie SRPT** das für die Prozesse P_1-P_7 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_7 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.

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

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

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

Fortsetzung nächste Seite!

Aufgabe 3: Petrinetze

In dieser Aufgabe soll eine Modifikation des Erzeuger-/Verbraucherproblems als Petrinetz modelliert werden. Beim klassischen Erzeuger-/Verbraucherproblem liegt folgende Situation vor:

- Es gibt zwei Prozesse. Einen Erzeuger (E) und einen Verbraucher (V).
- E produziert Resultate (R), die zunächst in einem gemeinsamen Speicher (S) der Kapazität max abgelegt werden.
- V entnimmt die Resultate dem gemeinsamen Speicher S und verbraucht diese.
- V darf nur mittels des gemeinsamen Speichers S auf ein Resultat R zugreifen.
- E darf nur bei freien Plätzen ein Resultat R im gemeinsamen Speicher S ablegen.
- Der Speicher darf nicht gleichzeitig von E und V verändert werden.

Dieses klassische Problem wird nun durch Einführen der folgenden zusätzlichen Bedingung modifiziert:

- V darf ein Resultat nur aus dem Speicher S entfernen, wenn danach noch **mindestens zwei Resultate** darin enthalten sind.

Bearbeiten Sie unter Berücksichtigung der gegebenen Anforderungen nun die folgenden Teilaufgaben.

1. Modellieren Sie das beschriebene modifizierte Erzeuger-/Verbraucherproblem durch ein Petrinetz. Erweitern Sie dazu den nachfolgend angegebenen Rahmen um die minimal notwendigen Stellen, Marken, Kanten und Transitionen. Versehen Sie alle Stellen und Transitionen mit aussagekräftigen Bezeichnern. Gehen Sie von folgenden Bedingungen aus:

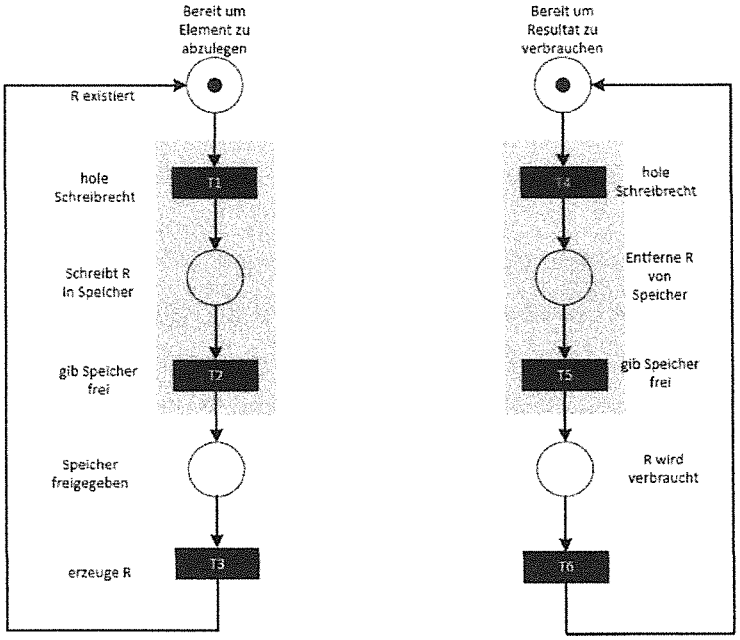
   - Die Kapazität max des gemeinsamen Speichers S beträgt 3.
   - Zu Beginn befindet sich bereits ein fertiges Resultat im gemeinsamen Speicher S.

Hinweis: Überlegen Sie sich zunächst die Funktionsweise des klassischen Erzeuger-/Verbraucherproblems und leiten Sie darauf basierend die nötigen Anpassungen her.

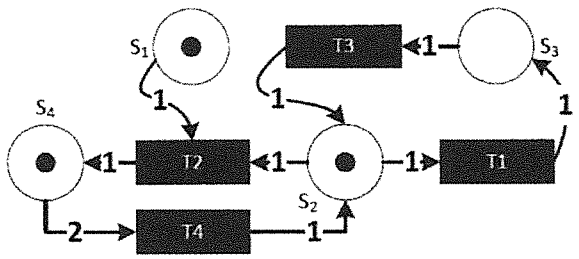
(Beachten Sie die Abbildung auf der nächsten Seite!)

Fortsetzung nächste Seite!

Der vorgegebene Rahmen ist:



2. Gegeben sei nun folgendes Petrinetz:



Leiten Sie den Erreichbarkeitsgraphen für dieses Petrinetz her.

3. Begründen Sie jeweils separat, ob im Petrinetz aus Teilaufgabe 2 ein Deadlock bzw. ein partieller Deadlock entstehen kann.

Fortsetzung nächste Seite!

Aufgabe 4: Seitenersetzungsstrategien

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 = 2, 4, 3, 0, 2, 4, 1, 3, 3, 4, 1, 0, 1, 4, 1, 2
```

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 ($i \in \{0,1,2\}$) 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 LFU (Least Frequently Used), indem Sie eine Tabelle mit dem bekannten Schema aus Teilaufgabe 1 erstellen. Statt der Zeit t soll allerdings nun für jeden Seitenrahmen die Anzahl der Seitenzugriffe seit dem Laden einer Seite festgehalten werden. Ordnen Sie in der Tabelle wieder jede referenzierte Seite dem entsprechenden Seitenrahmen f_i ($i \in \{0,1,2\}$) zu und dokumentieren Sie die Anzahl der Seitenzugriffe. Geben Sie zudem nach jedem Seitenzugriff die aktuelle Summe an Seitenfehlern an. Falls mehrere Seitenrahmen als Kandidaten für die neue Seite infrage kommen, so soll diejenige mit der kleinsten Indexnummer gewählt werden.

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

3. Nun sollen die theoretischen Eigenschaften der Seitenersetzungsstrategie Least Frequently Used (LFU) untersucht werden. Dazu soll untersucht werden, wie viele Seitenfehler für das Beispiel aus Teilaufgabe 2 unter Verwendung von LFU entstehen, wenn das System 1, 2, 3, 4 oder 5 Seitenrahmen besäße. Der zugehörige *Distance String* lautet ∞ ∞ ∞ ∞ 3 3 ∞ 4 0 2 2 4 1 2 1 4. 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 LFU (Least Frequently Used) auftreten?

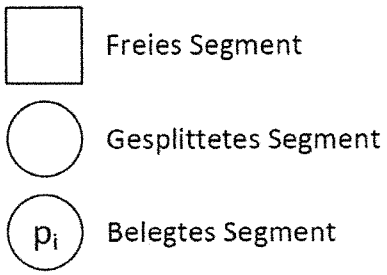
Fortsetzung nächste Seite!

Aufgabe 5: Buddy-Systeme

Gegeben sei ein mobiles System mit einem Hauptspeicher von 2 GB = 2048 MB, der byteweise adressiert wird. Zur Speicherverwaltung werden Buddy-Systeme eingesetzt. Dabei wird immer die am weitesten links stehende Speicherzelle geteilt, wenn ein neuer Prozess eingefügt wird. Die minimale Buddy-Größe soll 128 MB betragen. Bearbeiten Sie nun folgende Aufgaben:

1. Es werden 5 Prozesse der Reihe nach in das Buddy-System eingefügt:
   a) p_1: 510 MB
   b) p_2: 382 MB
   c) p_3: 76 MB
   d) p_4: 248 MB
   e) p_5: 426 MB

Zeichnen Sie insgesamt 5 Buddy-Bäume, die jeweils den Zustand der Speicherbelegung darstellen, nachdem ein weiterer der 5 Prozesse eingefügt wurde. Es muss genau ersichtlich sein, ob es sich um ein freies, ein gesplittetes bzw. um ein belegtes Segmente des Hauptspeichers handelt. Kennzeichnen Sie die entsprechenden Segmente eines Buddy-Baums mit den folgenden Symbolen:

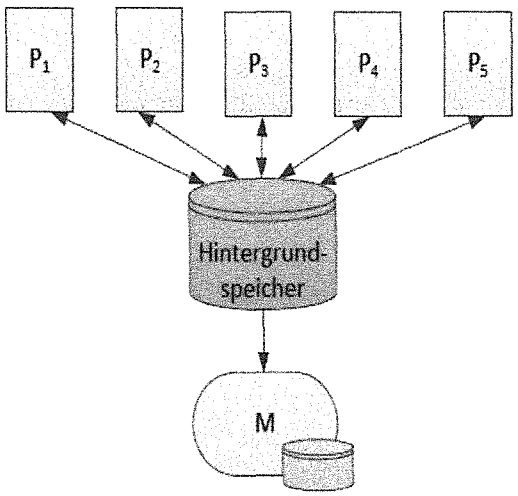


2. Wieviele Bits werden benötigt, um ein Byte im gegebenen Speicher adressieren zu können? Ergänzen Sie im letzten Buddy-Baum der Teilaufgabe 1 (also dem Buddy-Baum nach dem Einfügen von p_5) für alle freien, belegten und gesplitteten Segmente die ersten 6 Bits ihrer Speicheradresse.
3. Wieviel Speicher würde jedem der 5 Prozesse nach dem Buddy-Verfahren tatsächlich zur Verfügung stehen? Geben Sie den maximal zur Verfügung stehenden Speicherplatz für jeden der eingefügten Prozesse an.
4. Die eingefügten Prozesse p_1 bis p_5 benötigen insgesamt 1642 MB. Durch die Verwendung von Buddy-Systemen weicht der tatsächlich zugeordnete Hauptspeicher von diesem Wert ab. Wie viele von den ursprünglichen 2048 MB stehen für weitere Prozesse somit nur noch zur Verfügung und wieviel Speicher wird letztendlich verschwendet? Wie nennt man den für diese Verschwendung verantwortlichen Effekt?
5. Ein neuer Prozess p_6 mit einem Speicherbedarf von 240 MB soll gestartet werden. Ist es möglich diesen Prozess einem Buddy zuzuweisen? Falls ja, zeichnen sie den aktualisierten Buddy-Baum. Falls nein, erläutern Sie den Grund.
6. Zunächst terminiert der Prozess p_3 und danach der Prozess p_4. Zeichen Sie den aktualisierten Buddy-Baum nach jeder der beiden Prozessterminierungen. Kennzeichnen Sie dabei freie, gesplitteten bzw. belegten Segmente wieder mit den zuvor verwendeten Symbolen.

Fortsetzung nächste Seite!

Aufgabe 6: Koordination von Threads

In dieser Aufgabe soll der Betrieb eines Terminals in Java simuliert werden, auf dem insgesamt 5 Prozesse aktiv sind. Es wird angenommen, dass die Prozesse P_1 bis P_5 spezielle Rechenprozesse sind, die regelmäßig große Datenmengen in speziellen *Zugriffsphasen* auf den gemeinsamen Hintergrundspeicher schreiben. Aus Performanzgründen dürfen zu einem Zeitpunkt jedoch maximal 3 Rechenprozesse gleichzeitig den Hintergrundspeicher beschreiben. In den *Zwischenphasen* greifen die Rechenprozesse nicht auf den Hintergrundspeicher zu und führen spezielle Berechnungen durch. In regelmäßigen Abständen soll der aktuelle Stand des Hintergrundspeichers über ein Bandlaufwerk gesichert werden, wobei exakt der Zustand zu einem dedizierten Zeitpunkt gespeichert werden soll. Diese Datensicherung führt ein weiterer Datensicherungsprozess M durch.
Der Datensicherungsprozess beendet sich im Ruhezustand, wenn aktuell keine Sicherung nötig ist. Er geht in den Wartezustand über, wenn eine Sicherung nötig ist und meldet einen Sicherungswunsch an. Kein Rechenprozess darf dann mehr einen Schreibzugriff beginnen. Sobald alle noch laufenden Schreibzugriffe beendet wurden, beginnt der Datensicherungsprozess seine Arbeit und sichert die Daten auf ein Bandlaufwerk. Während der Datensicherung dürfen die Prozesse P_1 bis P_5 nicht auf die Festplatte zugreifen und müssen warten. Gleichzeitig gilt, dass M keine Festplattenzugriffe machen darf, solange einer der Prozesse P_1 bis P_5 aktuell noch auf die Festplatte zugreift. Das beschriebene Szenario ist in folgender Abbildung nochmals dargestellt:



Im Folgenden soll eine Klasse Speicher zur Simulation des Hintergrundspeichers implementiert werden.
Die Beispielimplementierungen der Klassen Rechenprozess, Datensicherungsprozess und Terminal soll Ihnen verdeutlichen, wie die Klasse Speicher verwendet werden kann:

Fortsetzung nächste Seite!

Die Klasse Terminal:

```
1  public class Terminal {
2      private Speicher my_speicher;
3      private Rechenprozess[] rechenprozesse;
4      private Datensicherungsprozess my_datensicherungsprozess;
5
6      public Terminal() {
7          my_speicher = new Speicher(3);
8          rechenprozesse = new Rechenprozess[5];
9          for(int i = 0; i < rechenprozesse.length; i++) {
10             rechenprozesse[i] = new Rechenprozess(my_speicher, i);
11             rechenprozesse[i].start();
12         }
13         my_datensicherungsprozess = new Datensicherungsprozess(my_speicher);
14         my_datensicherungsprozess.start();
15     }
16
17
18     public static void main(String[] args) {
19         new Terminal();
20     }
21 }
```

Die Klasse Rechenprozess:

```
1  import java.util.Random;
2
3  public class Rechenprozess extends Thread {
4      private Speicher my_speicher;
5      private int id;
6      private Random generator;
7
8      public Rechenprozess(Speicher speicher, int id) {
9          this.my_speicher = speicher;
10         this.id = id;
11         generator = new Random();
12     }
13
14     public void run() {
15             try {
16                 Thread.sleep(generator.nextInt(2000) + 500);
17                 my_speicher.schreibzugriffBeginnen(id);
18                 Thread.sleep(generator.nextInt(2000) + 500);
19                 my_speicher.schreibzugriffBeenden(id);
20             }
21             catch(InterruptedException ie) {}
22     }
23 }
```

Fortsetzung nächste Seite!

Die Klasse Datensicherungsprozess:

```
1  import java.util.Random;
2
3  public class Datensicherungsprozess extends Thread {
4          private Speicher my_speicher;
5          private Random generator;
6
7          public Datensicherungsprozess(Speicher speicher) {
8                  this.my_speicher = speicher;
9                  generator = new Random();
10         }
11
12         public void run() {
13                 while (true) {
14                         try {
15                                 Thread.sleep(generator.nextInt(2000) + 500);
16                                 my_speicher.sicherungBeginnen();
17                                 System.out.println("Daten werden gesichert");
18                                 Thread.sleep(generator.nextInt(2000) + 500);
19                                 my_speicher.sicherungBeenden();
20                                 Thread.sleep(1000);
21                         } catch (InterruptedException ie) {
22                         }
23                 }
24         }
25 }
```

Bearbeiten Sie nun die folgenden Aufgaben:

1. Was versteht man allgemein unter einem kritischen Bereich?
2. Implementieren Sie den Konstruktor der Klasse Speicher. Verwenden Sie dabei den Coderahmen am Ende der Aufgabe und *kommentieren Sie Ihre Lösung ausführlich.* Die Klassenattribute sind dorts bereits deklariert und müssen durch den Konstruktur initialisiert werden.

Fortsetzung nächste Seite!

3. Implementieren Sie die Methode `schreibzugriffBeginnen(int id)`, welche den Beginn des Zugriff eines Rechenprozesses auf den Hintergrundspeicher modelliert, sowie die Methode `schreibzugriffBeenden(int id)`, welche im Gegenzug das Beenden des Zugriffs eines Rechenprozesses auf den Hintergrundspeicher modelliert. Beachten Sie dazu, dass alle oben genannten Anforderungen beachtet werden müssen. Insbesondere:

   - Maximal 3 Rechenprozesse dürfen den Hintergrundspeicher gleichzeitig beschreiben.
   - Sobald der Datensicherungsprozess einen Sicherungswunsch hat, darf kein Rechenprozess den Hintergrundspeicher mehr beschreiben, bis der Datensicherungsprozess fertig ist und der Hintergrundspeicher wieder freigegeben ist.

   Ergänzen Sie dazu den Coderahmen am Ende der Aufgabe und *kommentieren Sie Ihre Lösung ausführlich.*
   Hinweis: Sie können davon ausgehen, dass die Methoden `schreibzugriffBeginnen(int id)` bzw. `schreibzugriffBeenden(int id)` immer in einer sinnvollen Reihenfolge aufgerufen werden (siehe Beispielimplementierung der Klasse Rechenprozess).

4. Implementieren Sie nun die Methoden für den Datensicherungsprozess. Vervollständigen Sie dazu den Coderahmen für die Methoden `sicherungBeginnen()` und `sicherungBeenden()` in dem Coderahmen am Ende der Aufgabe.
   Hinweis: Sie können davon ausgehen, dass die Methoden `sicherungBeginnen()` und `sicherungBeenden()` immer in einer sinnvollen Reihenfolge aufgerufen werden (siehe Beispielimplementierung der Klasse Datensicherungsprozess).

5. Zeigen Sie zwei kritische Bereiche in Ihrem Programm auf. Wie wird hier sichergestellt, dass die Bedingung der Mutual Exclusion erfüllt ist?

Fortsetzung nächste Seite!

Folgender Coderahmen steht Ihnen zur Bearbeitung der Teilaufgaben 2), 3) und 4) zur Verfügung:

```
1  public class Speicher {
2          private static int maxRechenprozesse;
3          private int anzahlRechenprozesse;
4          private boolean hat_sicherungswunsch;
5
6
7
8          public Speicher(                                                    ) {
9
10
11
12
13
14
15
16
17
18         }
19
20         public synchronized void schreibzugriffBeginnen(int id) {
21                 // Wird durch einen Rechenprozess aufgerufen.
22                 try {
23                    while (                                                  ) {
24
25
26
27
28                        }
29                 } catch (InterruptedException ie) {
30                 }
31
32
33
34
35
36
37
38                 System.out.println("Rechenprozess " + id + " ==> Speicher ("
39                                 + anzahlRechenprozesse + " schreibend)");
40
41
42
43
44         }
45
46         public synchronized void schreibzugriffBeenden(int id) {
47                 // Wird durch einen Rechenprozess aufgerufen.
48
49
50
51
```

Fortsetzung nächste Seite!

```
52
53
54                  System.out.println("Speicher ==> Rechenprozess " + id + " ("
55                                  + anzahlRechenprozesse + " schreibend)");
56
57
58
59
60
61
62          }
63
64          public synchronized void sicherungBeginnen() {
65                  // Wird durch den Datensicherungsprozess aufgerufen.
66                  this.hat_sicherungswunsch = true;
67                  try {
68                     while (                                                ) {
69
70
71
72
73                     }
74                  } catch (InterruptedException ie) {
75                  }
76                  System.out.println("Datensicherungsprozess ==> Speicher ("
77                                  + anzahlRechenprozesse
78                                  + " Rechenprozesse schreibend)");
79
80
81
82
83
84          }
85
86          public synchronized void sicherungBeenden() {
87                  // Wird durch den Datensicherungsprozess aufgerufen.
88                  this.hat_sicherungswunsch = false;
89
90
91
92
93
94
95
96                  System.out.println("Speicher ==> Datensicherungsprozess ("
97                                  + anzahlRechenprozesse
98                                  + " Rechenprozesse schreibend)");
99
100
101
102
103
104
105
106
107          }
108
109 }
```