Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer

Kennzahl: _____

Kennwort: _____

Arbeitsplatz-Nr.: _____

Herbst 2008

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 Auswahlregeln zu bearbeiten sind

Anzahl der Druckseiten dieser Vorlage: 15

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. A1, B2)!

Bitte wenden!

THEMENSCHWERPUNKT A
(Datenbanksysteme)

AUFGABE 1

1. Modellierung

Eine Fluggesellschaft möchte in einer relationalen Datenbank Informationen über ihre Flugzeuge und Flugrouten speichern:

Die Flugzeuge werden anhand einer eindeutigen Nummer (FNr) identifiziert. Zusätzlich werden Reichweite und Reisegeschwindigkeit gespeichert. Die Flugzeuge teilen sich in zwei disjunkte Gruppen: Passagierflieger und Transportflieger. Bei den Passagierfliegern sollen die jeweils in den Klassen (First, Dynasty, Economy) vorhandenen Sitzplätze gespeichert werden. Bei den Transportfliegern sollen maximales Gewicht, maximale Höhe, maximale Breite und maximale Tiefe der Nutzlast gespeichert werden.
Die Flugzeuge werden verwendet, um vorher festgelegte Touren zu fliegen. Wenn ein Flugzeug eine bestimmte Tour fliegen soll, müssen Abflugzeitpunkt bei Tourbeginn und Ankunftszeitpunkt bei Tourende gespeichert werden. Für die Touren werden jeweils eine eindeutige Nummer (TNr) und die Gesamtlänge der Tour gespeichert.
Jede Tour unterteilt sich in mehrere Teilstrecken. Die Teilstrecken können nur als Bestandteile von Touren auftreten. Für die Teilstrecken jeder Tour werden jeweils aufsteigende Nummern (ab „1“) vergeben um die Flugreihenfolge festzulegen. Für die einzelnen Teilstrecken sollen zusätzlich der Abflugort, Abflugzeitpunkt, Ankunftsort, Ankunftszeitpunkt und die Länge der Teilstrecke gespeichert werden.
Die Orte werden anhand einer eindeutigen Nummer (ONr) identifiziert. Als Zusatzinformationen werden für die Orte Name, Postleitzahl und Land gespeichert.

a) Entwerfen Sie für das beschriebene Szenario ein ER-Diagramm. Bestimmen Sie hierzu:

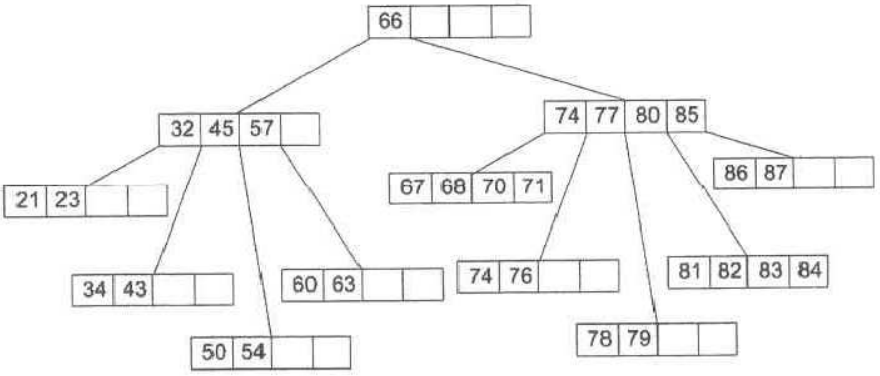
- die Entity-Typen, die Relationship-Typen und jeweils deren Attribute;
- ein passendes ER-Diagramm;
- die Primärschlüssel der Entity-Typen, welche Sie anschließend in das ER-Diagramm eintragen;
- die Funktionalitäten der Relationship-Typen, welche Sie ebenfalls in das ER-Diagramm eintragen.

b) Überführen Sie das ER-Modell aus Aufgabe a) in ein verfeinertes relationales Modell. Geben Sie hierfür die verallgemeinerten Relationenschemata an.

Fortsetzung nächste Seite!

2. Indexstrukturen

Als Indexstruktur einer Datenbank sei folgender B-Baum ($k = 2$) gegeben:



Führen Sie nacheinander die folgenden Operationen aus. Geben Sie die auftretenden Zwischenergebnisse an. Teilbäume, die sich in einem Schritt nicht verändern, müssen nicht erneut gezeichnet werden. Sollten Wahlmöglichkeiten auftreten, so sind größere Schlüsselwerte bzw. weiter rechts liegende Knoten zu bevorzugen.

a) Einfügen von Wert 72

b) Löschen von Wert 32

3. Zerlegungen

Gegeben sei ein Schema $R = (A, B, C, D, E, G)$ und eine FD-Menge

$F := \{E \rightarrow C, C \rightarrow B, CE \rightarrow G, B \rightarrow A\}$.

1. In welcher Normalform ist R bzgl. F?

2. Bestimmen Sie eine 3NF-Zerlegung von R bzgl. F.

3. Ermitteln Sie eine BCNF-Zerlegung von R bzgl. F.

Fortsetzung nächste Seite!

4. SQL

Betrachten Sie die folgenden Relationenschemata zur Verwaltung eines Weinlagers:

Wein:	{ID: integer, Art: string, Lage: string, Jahr: integer, Alc: float}
Weinlager:	{ID: integer, V_Preis: float, Bestand: integer, Min_Bestand: integer}
Weinempfehlung:	{Gericht: string, WeinID: integer}
Weinsorte:	{Art: string, Farbe: string}

Formulieren Sie die folgenden Anfragen in Standard-SQL. Zur Vereinfachung dürfen Sie dabei beliebige Views definieren.

a) Finden Sie alle Gerichte, die zu dem Wein mit dem größten Bestand passen.

b) Geben Sie Art und Durchschnittspreis der Weinsorten aus, deren Durchschnittspreis unter 10 € liegt.

c) Welche Weine werden nie empfohlen?

5. Transaktionen

Ein strikter 2-Phasen-Sperr-Protokoll-Scheduler erhält folgenden, aus den elementaren Operationen der Transaktionen T_1 und T_2 bestehenden Input-Schedule:

$s_{in} = r_1(A)r_1(B)r_2(B)w_2(A)w_1(A)c_1c_2$

Bestimmen Sie einen möglichen Output-Schedule s_{out}, den der Scheduler erzeugt. Beachten Sie hierbei, dass der Output-Schedule die erzeugten *lock*- und *unlock*-Befehle enthalten muss.

AUFGABE 2

1. Transaktionen und Transaktionsverarbeitung

1.1 Erklären Sie die grundlegenden Eigenschaften einer Transaktion (ACID-Prinzip).

1.2 Skizzieren Sie kurz ein Beispiel für das Lost-Update-Problem.

1.3 Was geschieht in den beiden Phasen des 2-Phasen-Sperrprotokolls?

2. ER-Modellierung und Relationenmodell

2.1 ER-Modellierung

Erstellen Sie das Modell eines fiktiven Auftragsverwaltungssystems in E/R-Notation. Attribute von Entitäten und Beziehungen sind anzugeben; Schlüsselattribute werden durch Unterstreichen gekennzeichnet. Die Kardinalitäten von Beziehungen sollen ins Diagramm aufgenommen werden. Führen Sie Surrogatschlüssel nur ein, falls es nötig ist.

„Auftragverwaltungssystem"

In unserem Minisystem müssen Rechnungen mit Rechnungsnummer, Datum, Name und Adresse des Kunden gespeichert werden. Eine Rechnung besteht dabei aus einer oder mehreren Rechnungspositionen.

Kunden werden unterteilt in Privatkunden und Geschäftskunden. Die Zuordnung ist eindeutig und vollständig, d.h. jeder Kunde gehört genau einer der beiden Gruppen an. Bei Privatkunden wird für die leichtere Abrechnung noch die Bankverbindung gespeichert, Geschäftskunden erhalten eine Nummer für ein eigenes Abrechnungskonto.

Eine Rechnungsposition besteht aus der Artikelnummer, der Menge, dem aktuellen Verkaufspreis, der Stückzahl und einer innerhalb einer Rechnung eindeutigen Rechnungspositionsnummer.

Ebenfalls sollen die Artikel verwaltet werden können, die durch eine Artikelnummer, einen Namen und eine Beschreibung gekennzeichnet sind. Zusätzlich wird der Einkaufspreis der Artikel gespeichert.

Artikel können entweder so genannte Einzelartikel sein oder zu Paketartikeln gebündelt werden. Paketartikel beinhalten dabei mehrere Einzelartikel und sollen über eine eigene Artikelnummer ausgewählt werden können. Auch die Einzelartikel in einem Paketartikel müssen separat in eine Rechnung aufgenommen werden können.

Fortsetzung nächste Seite!

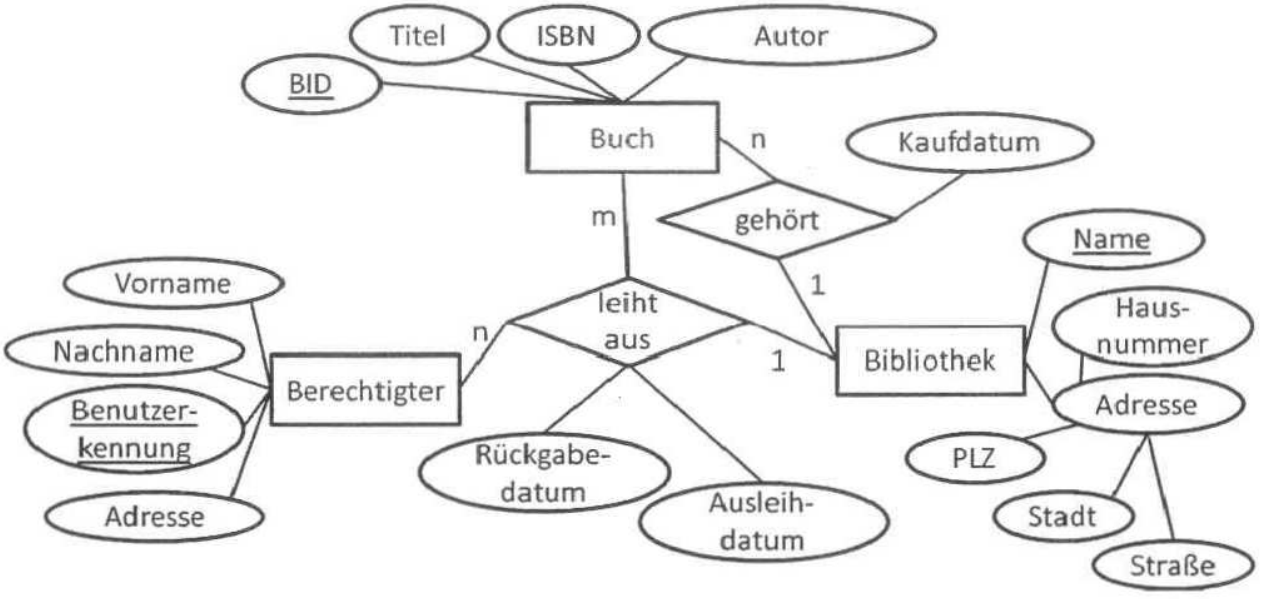
2.2 Relationenmodell: „Bibliothek“

Ausgehend von der gegebenen ER-Darstellung ist ein Relationenschema in dritter Normalform (3. NF) zu entwerfen.

Primärschlüssel werden dabei durch Unterstreichen, Fremdschlüssel durch Nennung der referenzierten Relation inklusive Attribut(e) in eckigen Klammern hinter dem Attributsnamen kenntlich gemacht, z. B.:

```
Haus (Straße, OrtId[Ort(OrtId)])
Ort (OrtId, PLZ, Name)
```

Das Attribut `OrtId` der Relation `Haus` verweist als Fremdschlüssel auf das Attribut `OrtId` der Relation `Ort`.



Fortsetzung nächste Seite!

3. SQL

Bitte beachten Sie: Primärschlüssel werden durch Unterstreichen, Fremdschlüssel durch Nennung der referenzierten Relation inklusive Attribut(e) in eckigen Klammern hinter dem Attributsnamen kenntlich gemacht, z.B.:

```
Haus (Straße, OrtId[Ort(OrtId)])
Ort (OrtId, PLZ, Name)
```

Das Attribut `OrtId` der Relation `Haus` verweist als Fremdschlüssel auf das Attribut `OrtId` der Relation `Ort`.

3.1 Szenario 1: „Online-Movie-Store"

Die Filme eines Online-Movie-Stores sind in folgendem Schema gespeichert:

Online-Movie-Store

```
FILM            (FID, Titel, Erscheinungsjahr, Laufzeit,
                 Drehort[ORT(OID)])

SCHAUSPIELER    (SID, Name, Geburtsjahr)

ORT             (OID, Land, Stadt)

SPIELT_MIT      (Schauspieler[SCHAUSPIELER(SID)], Film[FILM(FID)])
```

Bemerkung: Die Laufzeit eines Films ist als Zahl in Minuten angegeben.

Formulieren Sie die folgenden Anfragen an das oben stehende Schema in SQL:

i) Geben Sie die Namen aller Schauspieler aus, die in der Datenbank enthalten sind. Sortieren Sie die Ausgabe in aufsteigender Reihenfolge.

ii) Welche im Jahr 1996 erschienenen Filme spielen am Drehort Berlin (Deutschland)? Geben Sie Titel und Laufzeit an.

iii) Geben Sie Titel und Erscheinungsjahr aller Filme, in denen Heinz mitgewirkt hat, sortiert nach dem Erscheinungsjahr aus.

iv) Gesucht sind alle Filme, die in einem Jahr erschienen sind, in welchem insgesamt mehr als 4500 Minuten Film produziert worden sind. Außerdem sollen die Filme alle in New York (USA) gespielt haben. Die Titel dieser Filme sind mit Jahr und Gesamtlaufzeit des Jahres auszugeben.

v) In welchen Jahren wurden Filme gedreht, die Harrison und Tommy als Darsteller beinhalten?

Fortsetzung nächste Seite!

3.2 Szenario 2: „Schule modelliert!“

Die Schulverwaltungsdatenbank eines Gymnasiums wird gerade neu aufgebaut und die Mitglieder des Kollegiums erkunden die neuen Möglichkeiten der Datenbank.

```
Fach (Name, Beschreibung)
Schueler (SID, Name, Vorname, Geburtsdatum, Klasse)
Lehrer (LID, Name, Besoldungsstufe, Fachkombination)
Unterricht (LID[Lehrer(LID)], Klasse, Name[Fach(Name)], Tag,
Uhrzeit)
Raum (RID, Raumnummer, Gebaeude, AnzahlPlaetze)
```

Es existieren folgende Fremdschlüsselbeziehungen: Das Attribut `Name` der Relation `Unterricht` bezieht sich auf das Attribut `Name` der Relation Fach. Das Attribut `LID` der Relation `Unterricht` bezieht sich auf das Attribut `LID` der Relation `Lehrer`.
Die Attribute `SID` der Relation `Schueler` und `LID` der Relation `Lehrer` sind Ganzzahlen. Das Attribut `Name` der Relation Fach ist ein Text. Bei dem Attribut `Tag` der Relation `Unterricht` handelt es sich um ein Datum. Das Attribut `Uhrzeit` wird als fünfstellige Zeichenfolge im Format hh:mm angegeben. Das Attribut `AnzahlPlaetze` der Relation `Raum` ist eine dreistellige Ganzzahl. Das Attribut `Geburtsdatum` der Relation `Schueler` wird als Datum gespeichert. Die Datentypen der restlichen Attribute sind dem Kontext zu entnehmen.
Das Attribut `Klasse` der Relation `Unterricht` und das Attribut `Klasse` der Relation `Schueler` haben denselben Wertebereich.
Geben Sie SQL-Anweisungen für folgende Aufgaben an:

i) Erzeugung des beschriebenen Relationenschemas mit Tabellen, Primary Key Constraints und Foreign Key Constraints.

ii) Der Schüler „Kurt Binsen“ (Geburtsdatum: 13.02.1991) hat die Schule gewechselt und soll der Tabelle `Schueler` hinzugefügt werden (Klasse: 10a).

iii) Die Klasse „7b“ soll von der Lehrerin „Frau Schmidt“ (LID: 15) montags um 09:30 Uhr im Raum „02.124“ (RID: 124) unterrichtet werden. „Frau Schmidt“ und Raum „02.124“ sind in den entsprechenden Tabellen bereits eingetragen.

THEMENSCHWERPUNKT B
(Betriebssysteme)

AUFGABE 1

1.1 Die folgenden vier Jobs kommen in einem System zu den angegebenen Zeitpunkten (in Minuten) an:

Jobnummer	Ankunftszeit	Bearbeitungszeit
1	0	8
2	2	2
3	3	5
4	4	2

(a) Zeichnen Sie die Gantt-Diagramme für die Scheduling-Strategien First-Come-First-Served (FCFS), Round Robin (RR mit Zeitquantum=1) und Shortest-Job-First (SJF).

(b) Für jeden Job und jede der in (a) genannten Scheduling-Strategien: Berechnen Sie die Turnaround-Zeit. Berechnen Sie für jede Strategie die Average-Turnaround-Zeit.

1.2 In einem Demand-Paging System mit vier Kacheln sei für einen Prozess die Seitenreferenz-Folge zu bearbeiten:

0, 1, 0, 2, 3, 0, 4, 5, 0, 1

Ermitteln Sie die Anzahl der Seitenfehler, jeweils für die Ersetzungsstrategien

(a) First-In-First-Out,

(b) Optimale Ersetzung (OPT, „Belady's Strategie"), und

(c) Least-Recently-Used (LRU).

Notieren Sie zeilenweise nach jeder Seitenreferenz die komplette Speicherbelegung des Systems und markieren Sie die Zeilen, die nach einem Seitenfehler entstanden. Das erste Laden einer Seite zählt dabei ebenfalls als Seitenfehler.

Fortsetzung nächste Seite!

1.3 In einem Arbeitsspeicher existieren die folgenden nicht-belegten Blöcke in der angegebenen Reihenfolge mit den Größen:

60KB, 36KB, 100KB, 56KB, 72KB

Geben Sie schrittweise die verbleibenden freien Speicherblöcke für die folgenden Anforderungen an (ebenfalls in dieser Reihenfolge) nach Blöcken der Größe:

48KB, 58KB, 64KB, 44KB

Verwenden Sie dabei die Vergabeverfahren:

(a) First-Fit

(b) Best-Fit

(c) Worst-Fit

1.4 Der Dino-Park T-Rex besteht aus einem Dinosaurier-Museum und einem Erlebnispark mit Safari-Rundfahrt. Es gibt im Museum Platz für maximal $m=500$ Besucher und für die Rundfahrt stehen $n=100$ Safari-Wagen zur Verfügung, die jeweils nur einen Besucher aufnehmen können.

Die Besucher werden erst in das Museum eingelassen, falls dort die Maximalzahl m zurzeit nicht überschritten ist, sonst müssen sie vor dem Museum warten. Je nach Interesse verbringen die Besucher einige Zeit im Museum und stellen sich dann an, um einen Safari-Wagen zu bekommen.

Wenn ein leerer Wagen verfügbar ist, lässt dieser den am längsten wartenden Besucher einsteigen und fährt ihn durch den Erlebnispark und dann zum Ausgang des Dino-Parks. Wenn alle n Wagen mit Besuchern unterwegs sind, müssen weitere Besucher am Museumsausgang warten. Falls ein Wagen bereitsteht, einen Besucher aufzunehmen, jedoch kein Besucher wartet, so muss auch der Wagen warten.

Zur Lösung dieser Synchronisierungsprobleme soll jeder Besucher und jeder Safari-Wagen als ein eigener Prozess modelliert werden und es sollen ausschließlich Counting-Semaphoren benutzt werden.

(a) Schildern Sie zunächst, welche Fehler bei Verzicht auf Synchronisierungsmaßnahmen oder bei fehlerhafter Verwendung der Semaphor-Operationen im Dino-Park auftreten können.

Fortsetzung nächste Seite!

(b) Entwerfen Sie eine Lösung in Pseudocode für einen
`process visitor()` und einen
`process car()`

mit den folgenden globalen Semaphoren und den angegebenen Initialisierungen:

```
counting_sema museum_space = 500;
                // Besucherkapazität des Museums
counting_sema cars_waiting = 100;
                // Anzahl der verfügbaren Safari-Wagen
counting_sema visitors_waiting = 0;
                // Anzahl der auf Safari-Wagen wartenden Besucher
```

AUFGABE 2

1. Scheduling-Algorithmen

a) Beschreiben Sie zunächst in ein bis zwei Sätzen die Unterschiede in der Funktion von präemptiven und nicht-präemptiven Schedulingalgorithmen.

b) Betrachten Sie die folgenden Prozesse:

Prozess	Ankunftszeit	benötigte Rechenzeit
P0	0	6
P1	1	4
P2	2	2
P3	3	5
P4	4	3

Führen Sie das Scheduling für dieses Beispiel mit den Algorithmen *(Non-Preemptive) Shortest-Job-First* und *Preemptive-Shortest-Job-First* (auch bekannt als *"Shortest-Remaining-Time-First")* durch.

Stellen Sie den Gesamtablauf je Algorithmus tabellarisch dar, indem Sie für jeden Zeitpunkt den jeweils aktiven Prozess, die getroffene Scheduling-Entscheidung sowie den Zustand der "Ready-Queue" *nach* der Scheduling-Entscheidung angeben. Sie dürfen Zeilen auslassen, in denen sich im Vergleich zur vorigen Zeile nichts ändert.

Gehen Sie dabei von folgender Annahme aus: Sollte das Scheduling-Kriterium keine eindeutige Scheduling-Entscheidung zwischen zwei oder mehreren Prozessen zulassen, so wird von diesen der Prozess mit der *frühesten Ankunfszeit* bevorzugt.

c) Berechnen Sie für obiges Beispiel und NUR für *Preemptive-Shortest-Job First* die durchschnittliche Wartezeit und den Durchsatz. Geben Sie die Durchlaufzeit von Prozess P0 an.

Fortsetzung nächste Seite!

2. Verklemmungen

a) Geben Sie die Bedingungen an, die erfüllt sein müssen, damit eine Verklemmung auftreten kann. Erläutern Sie dabei jede Bedingung etwa in einem Satz.

b) Beschreiben Sie, welche dieser Bedingungen durch geeignete Verfahren verhindert werden können und diskutieren Sie diese Verfahren, indem Sie folgende Fragen für jedes angegebene Verfahren beantworten: Wie ist jeweils die grundlegende Vorgehensweise? Welche Nachteile ergeben sich ggf. durch den Einsatz eines solchen Verfahrens in einem Betriebssystem?

c) Betrachten Sie das "Dining Philosophers" Problem:

Fünf Philosophen sitzen an einem rundem Tisch. Zwischen benachbarten Philosophen liegt jeweils ein Essstäbchen, in der Mitte steht eine große Reisschüssel. Jeder Philosoph wechselt seine Tätigkeit zwischen Essen und Denken ab. Ein Philosoph benötigt zwei Essstäbchen um essen zu können. Um mit Essen zu beginnen nimmt ein Philosoph zunächst das rechts von ihm liegende Essstäbchen, dann das links neben ihm liegende. Nach dem Essen legt der Philosoph beide Essstäbchen ab und denkt wieder. Wenn ein Essstäbchen gerade nicht da liegt, so wartet der Philosoph darauf, dass sein Nachbar es ablegt.

Diskutieren Sie, ob bei dieser Konstellation eine Verklemmung auftreten kann, indem Sie sich auf die o. g. Bedingungen beziehen und zeigen Sie, ob und wie diese Bedingungen im Dining Philosophers Beispiel zutreffen.

d) Die Philosophen aus dem o. g. Beispiel ändern nun nach leidvoller Erfahrung den Ablauf ihrer Sitzungen so, dass der älteste von ihnen immer das linke Stäbchen zuerst nimmt und dann das rechte. Diskutieren Sie in etwa 2-3 Sätzen, welche Auswirkungen dies ggf. auf die Möglichkeit des Auftretens von Verklemmungen hat.

3. Synchronisation

Ein Professor führt Studienberatungen für Studierende durch. Das Vorzimmer des Professors hat Sitzplätze für drei Studierende, die Studienberatung selbst findet immer als Einzelgespräch im Büro des Professors statt. Sind keine beratungsbedürftigen Studierenden anwesend, widmet der Professor sich seiner Forschung. Forscht der Professor gerade, so muss er benachrichtigt werden, wenn ein beratungsbedürftiger Student das Vorzimmer betritt. Wenn im Vorzimmer kein Sitzplatz mehr frei ist, müssen Studierende vor der Tür warten.

Fortsetzung nächste Seite!

Ihnen stehen blockierende (BinarySemaphore) und zählende Semaphore (CountingSemaphore) zur Verfügung, um eine Software zur Synchronisierung der Studienberatung zu implementieren:

```
public interface Semaphore{
   public void wait();
   public void signal();
}

public class BinarySemaphore  implements Semaphore{
    public BinarySemaphore(){...}

}

public class CountingSemaphore implements Semaphore{

  public CountingSemaphore(int N){...}

}
```

Vervollständigen Sie in den folgenden Teilaufgaben a) - c) die Codeschablonen für die Initialisierung, für die Klasse Student und für die Klasse Professor.

An den mit ... gekennzeichneten Stellen können beliebig viele Codezeilen eingefügt werden. Es soll ein möglichst hoher Grad an Parallelität erreicht werden.

a) Initialisierung

Codeschablone für die Initialisierung:

```
// Kontrolliert Zugang zum Vorzimmer des Professors
Semaphore vorzimmer =   ...;

// Der Professor bei der Beratungstätigkeit
Semaphore professor =   ...;

// Die auf den Sitzplätzen wartenden Studierenden
Semaphore studierende =  ...;
```

Fortsetzung nächste Seite!

b) Klasse Student

Codeschablone für die Klasse Student:

```
public class Student extends Thread {
  public void run() {

    ...

    < Student betritt das Vorzimmer des Professors >

    ...
  }
}
```

c) Klasse Professor

Codeschablone für die Klasse Professor:

```
public class Professor extends Thread {
  public void run() {

    ...

    < Professor forscht >

    ...

    < Professor berät einen Studenten >

    ...

  }
}
```