Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer

Kennzahl: _____

Kennwort: _____

Arbeitsplatz-Nr.: _____

Herbst 2014

66116

Erste Staatsprüfung für ein Lehramt an öffentlichen Schulen
— Prüfungsaufgaben —

Fach: **Informatik (vertieft studiert)**

Einzelprüfung: **Datenbanksysteme, Softwaretechnologie**

Anzahl der gestellten Themen (Aufgaben): **2**

Anzahl der Druckseiten dieser Vorlage: **15**

Bitte wenden!

Thema Nr. 1

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.

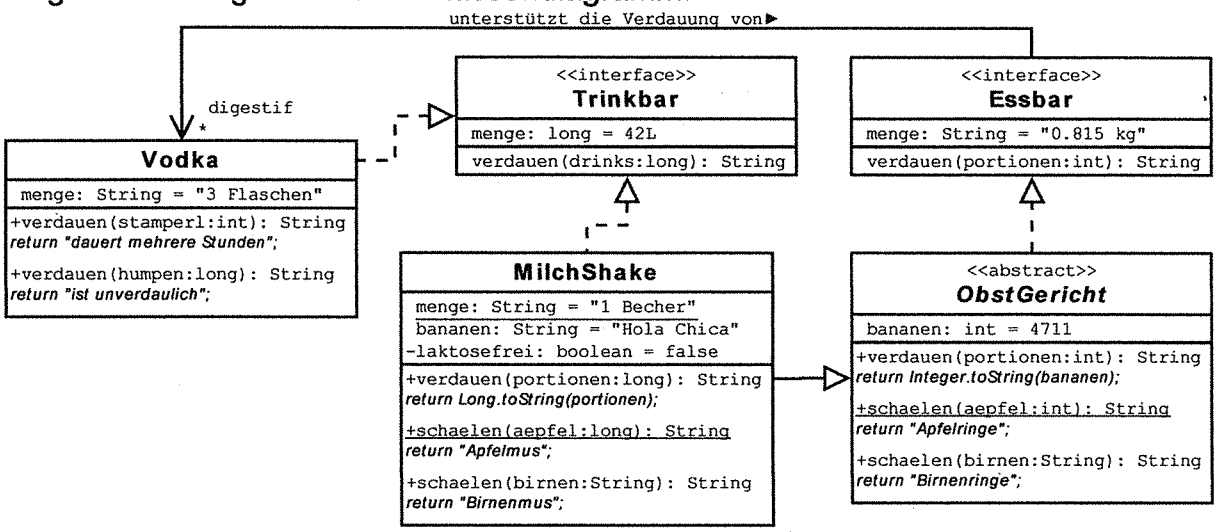
c) In welcher Normalform ist R? Geben Sie eine Begründung an.

Fortsetzung nächste Seite!

Teilaufgabe 2:

Aufgabe 1:

Gegeben sei folgendes UML-Klassendiagramm:



Zur besseren Lesbarkeit ist die *abstrakte* Klasse **ObstGericht** zusätzlich mit dem Stereotyp **<<abstract>>** gekennzeichnet. Die **return**-Zeile der jeweiligen Methode ist als Kommentar unmittelbar unter der Methodensignatur *kursiv* angegeben.

a) Angenommen, die Klasse **Mahlzeit** mit der folgenden Methode liegt mit den Klassen aus dem obigen UML-Diagramm im gleichen **package**. Geben Sie zu jeder **System.out.println**-Zeile die jeweils zu erwartende Ausgabe des Programms während der Ausführung an:

```
public static void main(String[] args) {
        /*****************************/
        Trinkbar tv = new Vodka();
        System.out.println(tv.verdauen(1));
        /*****************************/
        Trinkbar tm = new MilchShake();
        System.out.println(tm.menge);
        System.out.println(tm.verdauen(2));
        /*****************************/
        ObstGericht om = new MilchShake();
        System.out.println(om.menge);
        System.out.println(om.bananen);
        System.out.println(om.verdauen(3));
        System.out.println(om.schaelen(Integer.MAX_VALUE + 123));
        System.out.println(om.schaelen("Kiwi"));
}
```

b) Geben Sie eine Implementierung der Schnittstelle **Essbar** in einer geeigneten objekt-orientierten Programmiersprache Ihrer Wahl an.

c) Geben Sie nun eine Implementierung der Klasse **MilchShake** in der vorangehend gewählten Programmiersprache an.

d) Jemand möchte „Island Breeze“ als Unterklasse von Vodka und MilchShake modellieren, schließlich enthält der genannte Cocktail sowohl Vodka als auch Milch und Banane. Wieso ist das keine gute Idee, wenn Java die Zielsprache ist? In welcher gängigen Programmiersprache wäre das kein Problem?

Fortsetzung nächste Seite!

Aufgabe 2:

In dieser Aufgabe sollen Sie einen abstrakten Datentypen (kurz: ADT) **`Stechuhr`** zur Arbeitszeiterfassung mit folgenden Vereinfachungen spezifizieren und umsetzen:

- Die Stechuhr wird jeweils um Mitternacht ausgelesen und zurückgesetzt – zu einer Zeit, zu der sich kein Mitarbeiter mehr im Gebäude aufhalten darf. Daher muss beim Ermitteln der **`arbeitszeit`** (des heutigen Tages) nur das letzte Paar an Ein- und Ausbuchungsvorgängen berücksichtigt werden.
- Ein- und Ausbuchungszeiten sowie Arbeitszeiten werden in der Einheit „Minuten seit 0:00 Uhr des laufenden Tages" vom Typ **`Long`** verarbeitet und gespeichert.
- Mitarbeiter werden über ihren Personalstammdatensatz identifiziert, der garantiert ein-eindeutig ist und dessen ADT (vereinfacht) wie folgt aussieht:

adt *Mitarbeiter*
sorts *Mitarbeiter, String, Long*
ops

einstellen:	*String* × *Long*	→ *Mitarbeiter*	// Konstruktor
name:	*Mitarbeiter*	→ *String*	
id:	*Mitarbeiter*	→ *Long*	// eindeutige Kennung

axs
name(einstellen(n, i)) = n
id(einstellen(n, i)) = i
id(einstellen(n, i)) = id(einstellen(m, j)) ↔ n = m
end *Mitarbeiter*

adt *Stechuhr*
sorts *Stechuhr, Mitarbeiter, String, Long*
ops

init:	→ *Stechuhr*	// Konstruktor
...		

axs
...
end *Stechuhr*

Für die ADTs String und Long stehen alle bekannten Operationen zur Verfügung.

a) Ergänzen Sie den obigen Ausschnitt der algebraischen Spezifikation des ADTs **`Stechuhr`** um die notwendigen Signaturen und Axiome folgender Operationen:
   i) **`einbuchen`**: vermerkt die Ankunftszeit eines **`Mitarbeiters`** (beide als Parameter übergebenen) in „Minuten seit 0:00 Uhr des laufenden Tages"
   ii) **`ausbuchen`**: wie **`einbuchen`**, nur für das Verlassen des Gebäudes
   iii) **`leeren`**: setzt die **`Stechuhr`** zurück (nach dem Auslesen um Mitternacht)
   iv) **`arbeitszeit`**: bestimmt die heutige Arbeitszeit (Datentyp: Long) eines (als Parameter übergebenen) **`Mitarbeiters`**

   Hinweis: Sie dürfen bei Bedarf zusätzlich auch Hilfsoperationen ergänzen!

b) Geben Sie eine Implementierung des ADT **`Mitarbeiter`** in einer geeigneten objekt-orientierten Programmiersprache Ihrer Wahl an. Der Konstruktor **`einstellen`** des ADTs soll in der implementierten Klasse sowohl durch den Konstruktor selbst als auch durch eine gleichnamige Klassenmethode umgesetzt werden. Stellen Sie in der Klasse **`Mitarbeiter`** außerdem eine Methode **`Long getNewID()`** bereit, die bei jedem Aufruf eine neue eindeutige ID generiert.

Fortsetzung nächste Seite!

Aufgabe 3:

Gegeben seien folgende Java-Methode sowie ihre Vor- und Nachbedingung:

```
public static long funI(final int n) {
      /* P */long a = 1, b = 0;
      for (int i = 0; i < n - 1; i++) {
            b += a;
            a *= 3;
      }/* R */
      long ret = 3 * a - 4 * b;/* Q */
      return ret;
}
```

$P := n \geq 1$

$Q := ret = 3^n - 4 \cdot \sum_{j=0}^{n-2} 3^j$

Zur Vereinfachung nehmen Sie bitte im Folgenden an, dass die verwendeten Datentypen unbeschränkt sind und daher keine Überläufe auftreten können.

a) Welche der folgenden Bedingungen ist eine zum Beweisen der Korrektheit der Methode mittels *wp*-Kalkül (Floyd-Hoare-Kalkül) sinnvolle Schleifeninvariante?

i) $i = 0 \wedge a = 1 \wedge b = 0$

ii) $a = 3^i \wedge b = \sum_{j=0}^{i-1} 3^j \wedge i \leq n-1$

iii) $a = 3^{n-1} \wedge b = \sum_{j=0}^{n-2} 3^j$

iv) *true*

v) $ret = 3^n - 4 \cdot \sum_{j=0}^{n-2} 3^j$

b) Zeigen Sie *formal* mittels *wp*-Kalkül, dass die von Ihnen gewählte Bedingung unmittelbar vor Beginn der Schleife gilt, wenn zu Beginn der Methode die Anfangsbedingung P gilt.

c) Zeigen Sie *formal* mittels *wp*-Kalkül, dass die von Ihnen gewählte Bedingung tatsächlich eine Invariante der Schleife ist.

d) Bestimmen Sie *formal* mittels *wp*-Kalkül die schwächste Zusicherung *R*, die unmittelbar nach dem Ende der Schleife gelten muss, so dass am Ende der Methode die Nachbedingung Q erfüllt wird. Zeigen Sie *formal* mit Hilfe der von Ihnen gewählten Invariante, dass die Zusicherung *R* auch tatsächlich unmittelbar nach der Ausführung der Schleife gilt.

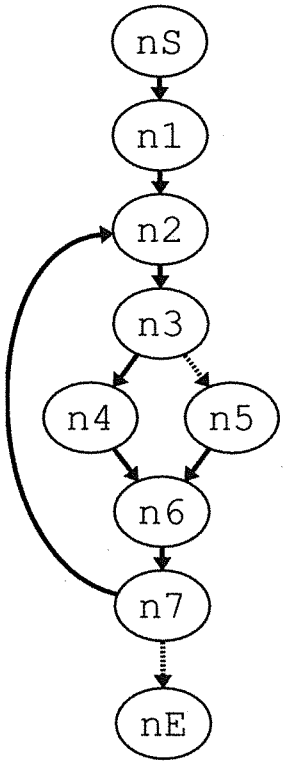
e) Beweisen Sie, dass die Methode immer terminiert. Geben Sie dazu eine Terminierungsfunktion an und begründen Sie kurz Ihre Wahl.

Fortsetzung nächste Seite!

Aufgabe 4:

Gegeben sei folgende Methode **double** p(**double** e) und ihr Kontrollflussgraph:

```
public static double p(double e) {
	double r = 0.0d;
	boolean v = false;
	double n = 1.0d;
	do {
		if (v) {
			r -= 4 / n;
		} else {
			r += 4 / n;
		}
		v = !v;
		n += 2;
	} while (e < 4 / n);
	return r;
}
```



a) Geben Sie je einen Repräsentanten aller Pfadklassen **im Kontrollflussgraphen** an, die zum Erzielen einer vollständigen
   i) Verzweigungsüberdeckung (Branch-Coverage, C_1)
   ii) Schleife-Inneres-Überdeckung (Boundary-Interior-Coverage, $C_{\infty,2}$)

   mit **minimaler** Testfallanzahl und möglichst kurzen Pfaden genügen würden.
b) Bestimmen Sie anhand des Kontrollflussgraphen des obigen Code-Fragments die maximale Anzahl linear unabhängiger Programmpfade, also die zyklomatische Komplexität nach McCabe.
c) Kann für dieses Modul eine 100%-ige Pfadüberdeckung erzielt werden? Begründen Sie kurz Ihre Antwort.
d) Geben Sie zu jedem Knoten die jeweilige Datenflussannotation (defs bzw. c/p-uses, sofern überhaupt vorhanden) für jede betroffene Variable in der zeitlichen Reihenfolge ihres Auftretens zur Laufzeit an.

Thema Nr. 2

Teilaufgabe 1:

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!

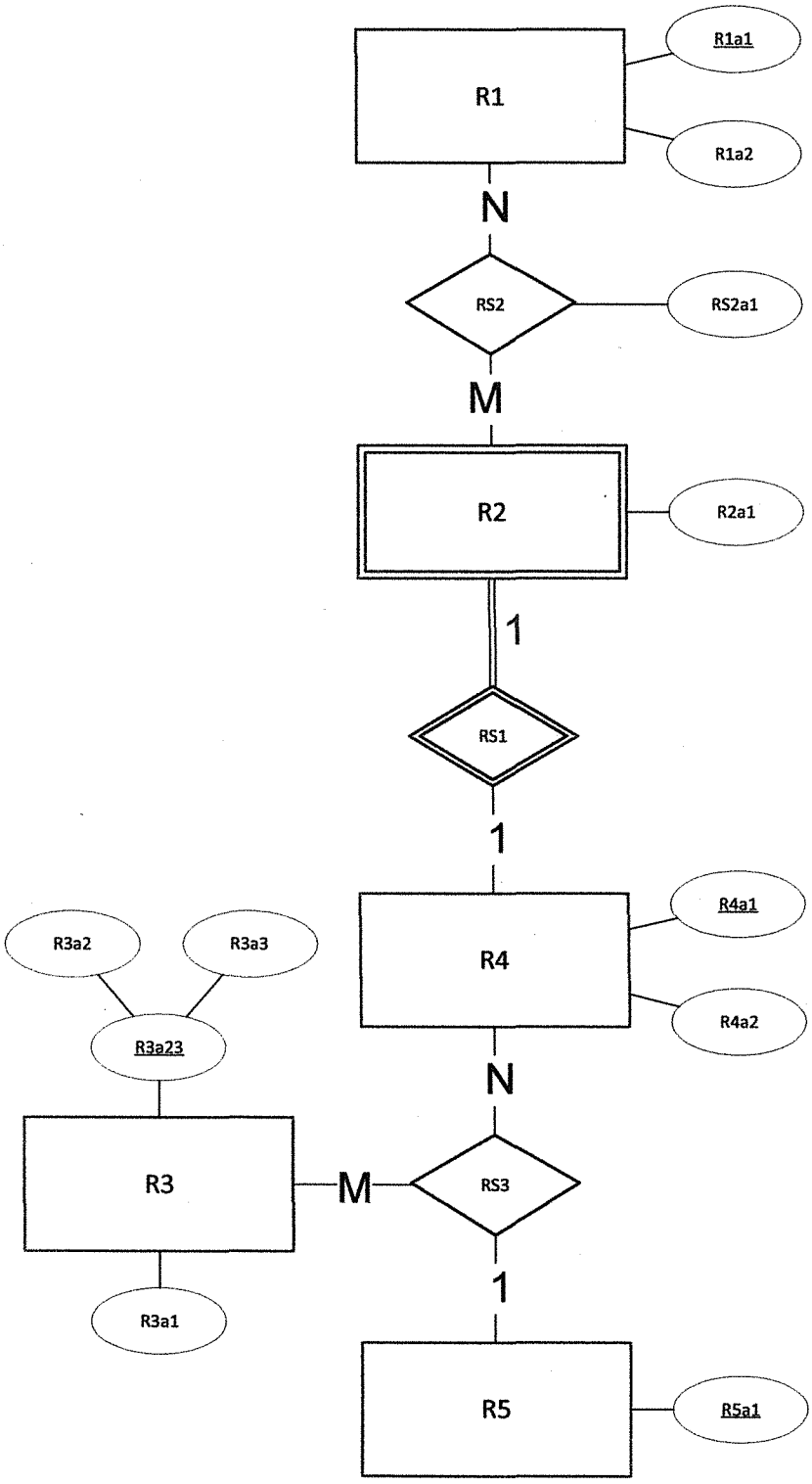
b) Abbildung auf Relationen

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.

Fortsetzung nächste Seite!

Teilaufgabe 2:

Aufgabe 1:

In Abbildung 1 ist eine Version des Algorithmus von Euklid zur Berechnung des größten gemeinsamen Teilers zweier ganzer, positiver Zahlen angegeben. Der Algorithmus nimmt als Eingabe eine ganze Zahl $A > 0$ und eine ganze Zahl $B > 0$. Nach Ausführen des Algorithmus soll der größte gemeinsame Teiler von A und B in der Variable a stehen. Die Funktion $ggt(A,B)$ bezeichnet dabei eine Funktion die den größten gemeinsamen Teiler zweier ganzer, positiver Zahlen A und B berechnet.

```
{A > 0 ∧ B > 0}
a := A;
b := B;
while (a ≠ b) {
   if (a > b) {
      a := a – b;
   } else {
      b := b – a;
   }
}
{a = ggt(A, B)}
```

Abbildung 1

Zeigen Sie die korrekte Funktionsweise des Algorithmus. Führen Sie dabei folgende Schritte aus[1]:

1. Geben Sie die Schleifeninvariante an.
2. Annotieren Sie das Programm unter Verwendung von Hoare Logik.
3. Begründen Sie für jede Anwendung der Konsequenzregel, warum die Implikation gilt.

Fortsetzung nächste Seite!

[1] Sie dürfen dabei die Funktion *ggt* verwenden.

Aufgabe 2:

Sie sollen das Design für ein einfaches Wahlsystem entwerfen. Das System soll dabei die Verteilung der Stimmen auf die einzelnen Parteien ermöglichen. Zusätzlich soll es verschiedene Darstellungen dieser Daten erlauben: Eine Tabelle, in der die Daten gelesen und auch eingegeben werden können, und ein Diagramm als alternative Darstellung der Informationen. Das System soll mit dem Model-View-Controller Muster modelliert werden.

1. Beschreiben Sie das Model-View-Controller Muster:
    a. Beschreiben Sie das Problem, welches das Muster adressiert.
    b. Beschreiben Sie die Aufgaben der Komponenten, die im Muster verwendet werden.
2. Modellieren Sie das System unter Anwendung des Musters:
    a. Entwerfen Sie ein UML Klassendiagramm.
    b. Implementieren Sie eine setup-Methode, die das Objektmodell erstellt.

Fortsetzung nächste Seite!

Aufgabe 3:

Im Folgenden ist ein Algorithmus angegeben, der für eine positive Zahl *until* die Summe aller Zahlen bildet, die kleiner als *until* und Vielfache von vier oder sechs sind. Für nicht positive Zahlen soll 0 zurückgegeben werden. Der Algorithmus soll also folgender Spezifikation genügen:

$$until > 0 \Rightarrow specialSums(until) = \sum\{y \mid 0 < y < until \wedge (y\%4 = 0 \vee y\%6 = 0)\}$$

$$until \leq 0 \Rightarrow specialSums(until) = 0$$

wobei % den Modulo-Operator bezeichnet.

```
public static long specialSums (int until)
   1. long sum = 0;
   2. if (until > 0)  {
   3.     for (int i = 1; i <= until; i++)  {
   4.         if (i % 4 == 0 || i % 6 == 0)  {
   5.             sum += i;
   6.         }
   7.     }
   8. }
   9. return sum;
```

Abbildung 2: Algorithmus specialSums

Beachten Sie, dass der Algorithmus nicht der Spezifikation genügt. Der Fehler liegt in der Bedingung der for-Schleife (Zeile 3). Der Fehler kann jedoch einfach korrigiert werden indem die Bedingung

$$i \leq until \quad \text{in} \quad i < until$$

geändert würde.

1) Zeichnen Sie das zum Programm in Abbildung 2 gehörige Ablaufdiagramm.
2) Schreiben Sie einen Testfall, der das Kriterium „100% Anweisungsüberdeckung“ erfüllt, aber den Fehler trotzdem nicht aufdeckt.
3) Schreiben Sie einen Testfall, der das Kriterium „100% Zweigüberdeckung“ erfüllt, aber den Fehler trotzdem nicht aufdeckt.
4) Schreiben Sie einen Testfall, der den Fehler aufdeckt. Berechnen Sie Anweisungsüberdeckung und Zweigüberdeckung ihres Testfalles.
5) Es ist nicht immer möglich vollständige Pfadüberdeckung zu erreichen. Geben Sie einen gültigen Pfad des Programmes an der nicht erreichbar ist.

Ein Testfall kann als Menge von Paaren dargestellt werden, wobei jedes Paar *(I,O)* die Eingabe *I* und die zu dieser Eingabe erwartete Ausgabe *O* darstellt.

Fortsetzung nächste Seite!

Aufgabe 4:

a) Erläutern Sie den Unterschied zwischen der partiellen Korrektheit und der totalen Korrektheit eines Programms.

b) Geben Sie die allgemeinen Regeln zum Nachweis der partiellen Korrektheit für die zweiseitig bedingte Anweisung und für die Schleife an.

c) Führen Sie für das nachfolgende Programm den Nachweis der partiellen Korrektheit. Verwenden Sie dazu

$$I = \{(x \geq c^2 \wedge (a = (c+1)^2) \wedge (c \geq 0)\}$$

als Schleifeninvariante. Alle Variablen repräsentieren ganze Zahlen.

```
// P = {true}
s1: c = 0;
s2: a = c + 1;
    if (y < 0)   // B1 = (y < 0)
s3: x = -y;
    else
s4: x = y;
    while (a < x + 1) {   // B2 = {a < x + 1}
s5: c = c + 1;
s6: a = a + 2c + 1;
    }
s7: r = 2c;
```

// Q = $\{(r^2 \leq 4x) \wedge (x < (0{,}5r+1)^2)\}$

d) Geben Sie eine geeignete Funktion an, mit der ein Terminierungsbeweis für die Schleife durchgeführt werden kann. Welche Anforderungen muss die Funktion erfüllen?

e) Geben Sie für die obige Schleife jeweils eine zu starke und eine zu schwache Invariante an. Erklären Sie jeweils kurz, welche Beweisteile sich mit den ungeeigneten Invarianten nicht durchführen lassen.

Fortsetzung nächste Seite!

Aufgabe 5:

Lösen Sie folgende Aufgabe in einer objektorientierten Programmiersprache Ihrer Wahl. Ein Regal habe 5 Fächer, in die je 30 Disks passen. Das Regal beinhaltet DVDs, CDs und BDs der Genres Musik, Action, Komödie, Thriller und Fantasy. Jede Disk ist mit 1 bis 10 bewertet, wobei 10 für sehr gut steht.

- Deklarieren Sie einen Aufzählungsdatentyp für den Typ der Disk. Deklarieren Sie einen weiteren Aufzählungsdatentyp für das Genre der Disk.
- Deklarieren Sie eine Klasse Disk, die den Typ, das Genre, die Bewertung, sowie den Titel der Disk speichert.
- Deklarieren Sie ein Array Regal, mit den Abmessungen des oben genannten Regals, das Disks enthält. Initialisieren Sie das Array als leeres Regal.
- Initialisieren Sie die Bewertung einer Disk. Schreiben sie dazu eine rekursive Methode erstelleStdBewertung, der ein Genre einer Disk übergeben wird und die die Bewertungen für alle Disks nach folgenden Regeln bezüglich des Genres vergibt:
  - Eine Disk mit Genre Musik wird mit einer 3 bewertet.
  - Komödien werden mit 2 bewertet.
  - Thriller werden mit dem zweifachen einer Komödie bewertet
  - Ein Fantasy-Film wird mit dem 1,5 fachen eines Thrillers bewertet.
  - Ein Actionfilm wird wie ein Thriller bewertet.
- Schreiben Sie eine Methode mittlereBewertung, die die mittlere Bewertung der Disks im Regal berechnet.