Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer

Kennzahl: _____

Kennwort: _____

Arbeitsplatz-Nr.: _____

Frühjahr 2015

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: **12**

Bitte wenden!

Thema Nr. 1

Teilaufgabe 1

DB.1 (Datenmodellierung: ER-Modell, SQL)

In einer *Musik-Datenbank* sollen folgende Informationen zu Interpreten und deren CDs modelliert werden:

- Zu einem Interpreten soll eine eindeutige ID, der Name, das Jahr seines Bühnenstarts, seine Geschäftsadresse sowie sein Musikgenre angegeben sein. Das Musikgenre kann mehrere Werte umfassen (mehrwertiges Attribut).

- Eine CD hat eine eindeutige ID, einen Namen (Titel), einen Interpreten, ein Erscheinungsdatum und bis zu 20 Positionen (Musikstücke). An jeder Position steht ein Musikstück. Für dieses ist der Titel und die Länge in Sekunden angegeben.

- Eine CD kann Auszeichnungen – z. B. vom Typ goldene Schallplatte oder Emmy bekommen. Ebenso kann auch ein einzelnes Musikstück Auszeichnungen bekommen.

a) Modellieren Sie das oben dargestellte Szenario möglichst vollständig in einem ER–Modell. Verwenden Sie, wann immer möglich, (binäre oder auch höherstellige) Relationships. Modellieren Sie Musikstücke in einem schwachen Entity-Typen.

b) Übertragen Sie Ihr ER–Modell – bis auf die Typen zu den Auszeichnungen – ins relationale Datenmodell. Erstellen Sie dazu Tabellen mit Hilfe von CREATE TABLE–Statements in *SQL*. Berücksichtigen Sie die Fremdschlüsselbeziehungen.

c) Es soll die Integritätsbedingung eingehalten werden, so dass die Anzahl der Positionen auf einer CD höchstens 20 ist. Schreiben Sie ein SELECT–Statement, das diese Integritätsbedingung überprüft, indem es die verletzenden CDs ausgibt.

d) Geben Sie geeignete INSERT–Statements an, die in alle beteiligten Tabellen jeweils mindestens ein Tupel einfügen, so dass alle Integritätsbedingungen erfüllt sind, nachdem alle Einfügungen ausgeführt wurden. Lediglich zu den Auszeichnungen müssen keine Tupel eingefügt werden.

Fortsetzung nächste Seite!

DB.2 (Datenbankanfragen und – änderungen in SQL)

Formulieren Sie in SQL die folgenden Anfragen, Views bzw. Datenmanipulations–Statements an Teile der *Musik–Datenbank* aus Teilaufgabe DB.1:

Interpret (Interpreten_ID, Name, Bühnenstart, Geschäftsadresse),
CD (CD_ID, Name, Interpreten_ID, Erscheinungsdatum),
Musikstück (CD_ID, Position, Titel, Länge),
Auszeichnung_CD (CD_ID, Typ),
Auszeichnung_Stück (CD_ID, Position, Typ).

a) Welche CDs hat 'Adele' herausgebracht? Geben Sie die Namen der CDs aus.

b) Geben Sie für alle Interpreten – gegeben durch die ID und den Namen – die Anzahl ihrer veröffentlichten CDs an.

c) Geben Sie die Länge des längsten Musikstücks auf der CD mit dem Namen 'Thriller' des Interpreten 'Michael Jackson' an.

d) Geben Sie die Namen aller Interpreten aus, die eine Auszeichnung für eine CD oder eines ihrer Musikstücke bekommen haben.

e) Fügen Sie ein, dass 'Adele' einen 'Emmy' für ihre CD mit dem Namen 'Adele 21' bekommen hat.

f) Ändern Sie die Geschäftsadresse von 'Genesis' auf 'Hollywood Boulevard 13, Los Angeles'.

DB.3 (Funktionale Abhängigkeiten und Zerlegungen)

Gegeben sei das Relationenschema $R=(U,F)$ mit der Attributmenge

$$U=\{A,B,C,D,E\}$$

und der folgenden Menge F von funktionalen Abhängigkeiten:

$$F=\{A\rightarrow B, ABC\rightarrow D, D\rightarrow BC\}.$$

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

b) Ist R in 3NF bzw. in BCNF? Geben Sie jeweils eine Begründung an.

c) Geben Sie eine Basis G von F an. Zerlegen Sie R mittels des Synthesealgorithmus in ein 3NF–Datenbankschema. Es genügt, die resultierenden Attributmengen anzugeben.

Fortsetzung nächste Seite!

Teilaufgabe 2

Aufgabe 1 (Klassendiagramm)

Erstellen Sie zu der folgenden Beschreibung eines automatisierten Lagers ein UML-Klassendiagramm, das Attribute, Assoziationen mit Kardinalitäten, Rollennamen und Namen sowie Methoden enthält. Setzen Sie dabei insbesondere die Konzepte der Vererbung, der abstrakten Klassen und der Kompositionsbeziehungen sinnvoll ein.

Tipp: Zeichnen Sie sie Kompositionsbeziehungen zuletzt ein.

Klassen und Assoziationen:

- Ein Lager beinhaltet ein Transportsystem, Bearbeiter, Orte und Artikel sowie Jobs, die ausgeführt werden sollen.
- Bearbeiter sind entweder Roboter oder Arbeiter.
- Orte sind entweder Ausgaben oder Lagerplätze.
- Das Transportsystem kontrolliert alle vorhandenen Roboter, kennt die Lagerplätze und verwaltet die Jobs.
- Ein Bearbeiter ist an genau einem Ort, transportiert maximal einen Artikel und ihm ist maximal ein Job zugewiesen.
- Ein Artikel ist an maximal einem Ort zu finden und hat einen Namen.
- Einem Job ist ein Artikel, der geholt werden soll, der Ort, an dem sich der Artikel befindet, und der Ort, zu dem er gebracht werden soll, bekannt.

Öffentliche Methoden:

- Jobs werden vom Transportsystem erzeugt. Zur Erzeugung eines Jobs werden dem Transportsystem der Artikel, der geholt werden soll, der Ort, an dem sich der Artikel befindet, und der Ort, zu dem er gebracht werden soll, übergeben.
- Bearbeiter können sich zu einem Ort bewegen, der ihnen genannt wird (Parameter der Methode).
- Bearbeiter können einen Job ausführen, der ihnen genannt wird.
- An der Ausgabe kann man Artikel bestellen; dazu nennt man der Ausgabe den Namen des Artikels.
- Die Ausgabe meldet wenn ein Artikel nicht verfügbar ist. Dabei wird zurückgegeben, ob der Artikel verfügbar ist oder nicht.
- Im Lagerplatz wird ein Artikel geladen; dazu muss der Artikel angegeben werden.

Fortsetzung nächste Seite!

Aufgabe 2 (Zustandsdiagramm)

Erstellen Sie ein UML-Zustandsdiagramm für einen Radiotuner. Mit dem Radiotuner können Sie ein Radioprogramm auf der Frequenz f empfangen. Beim Einschalten wird f auf 87,5 MHz gesetzt und der Tuner empfängt. Sie können nun die Frequenz in 0,5 MHz Schritten erhöhen oder senken. Bitte beachten Sie, dass das Frequenzband für den Radioempfang von 87,5 MHz bis 108 MHz reicht. Sobald f 87,5 MHz unter- bzw. 108 MHz überschreitet, soll f auf 108 MHz bzw. 87,5 MHz gesetzt werden. Weiterhin kann ein Suchmodus gestartet werden, der automatisch die Frequenz erhöht, bis ein Sender empfangen wird. Wird während des Suchmodus die Frequenz verändert oder erneut Suchen ausgeführt, dann wird die Suche beendet (und, je nach Knopf, ggf. noch die Frequenz um 0,5 MHz verändert).
Die Klasse RadioTuner besitzt folgende Methoden:

- tunerOn()
- tunerOff()
- increaseFrequency()
- decreaseFrequency()
- seek()
- hasReception()

Hinweis: Die Hilfsmethode hasReception() liefert true zurück, genau dann wenn ein Sender empfangen wird.

Aufgabe 3 (Objektsorientierte Programmierung und Entwurfsmuster)

In dieser Aufgabe implementieren Sie ein konzeptionelles Datenmodell für eine Firma, die Personendaten von Angestellten und Kunden verwalten möchte. Gegeben seien dazu folgende Aussagen:

- Eine Person hat einen Namen und ein Geschlecht (männlich oder weiblich).
- Ein Angestellter ist eine Person, zu der zusätzlich das monatliche Gehalt gespeichert wird.
- Ein Kunde ist eine Person, zu der zusätzlich eine Kundennummer hinterlegt wird.

a) Geben Sie in einer objektorientierten Programmiersprache Ihrer Wahl (geben Sie diese an) eine Implementierung des aus den obigen Aussagen resultierenden konzeptionellen Datenmodells in Form von **Klassen** und **Interfaces** an. Gehen Sie dabei wie folgt vor:

   - Schreiben Sie ein Interface *Person* sowie zwei davon erbende Interfaces *Angestellter* und *Kunde*. Die Interfaces sollen jeweils lesende Zugriffsmethoden (Getter) für die entsprechenden Attribute (Name, Geschlecht, Gehalt, Kundennummer) deklarieren.
   - Schreiben Sie eine abstrakte Klasse *PersonImpl*, die das Interface *Person* implementiert. Für jedes Attribut soll ein Objektfeld angelegt werden. Außerdem soll ein Konstruktor definiert werden, der alle Objektfelder initialisiert.

Fortsetzung nächste Seite!

- Schreiben Sie zwei Klassen *AngestellterImpl* und *KundeImpl*, die von *PersonImpl* erben und die jeweils dazugehörigen Interfaces implementieren. Es sollen wiederum Konstruktoren definiert werden, die alle Objektfelder initialisieren und dabei auf den Konstruktor der Basisklasse *PersonImpl* Bezug nehmen.

b) Verwenden Sie das Entwurfsmuster **Adapter**, um zu ermöglichen, dass vorhandene Angestellte die Rolle eines Kunden einnehmen können. Der Adapter soll eine zusätzliche Klasse sein, die das Kunden-Interface implementiert. Wenn möglich, sollen Methodenaufrufe an den adaptierten Angestellten delegiert werden. Möglicherweise müssen Sie neue Objektfelder einführen.

c) Verwenden Sie das Entwurfsmuster **Simple Factory**, um die Erzeugung von Angestellten, Kunden, sowie Adapter-Instanzen aus Aufgabe (b) zu vereinheitlichen. Die entsprechende Erzeugungs-Methode soll neben einem Typ-Diskriminator (z.B. einem Aufzählungstypen oder mehreren booleschen Werten) alle Parameter übergeben bekommen, die für den Konstruktor irgendeiner Implementierungsklasse des Interface Person notwendig sind.

Hinweis: Um eine Adapter-Instanz zu erzeugen, müssen Sie möglicherweise zwei Konstruktoren aufrufen.

Thema Nr. 2

Teilaufgabe 1

Aufgabe 1 Modellierung

Ein junges Start-Up möchte die Welt der Sozialen Netzwerke revolutionieren. Dazu entwerfen sie zunächst ein Modell für die Datenbank. Folgendes Szenario soll modelliert werden:

- Zu einer Person werden Vor- und Nachname gespeichert, außerdem das Datum der Registrierung.
- Personen können miteinander befreundet sein. Ein Attribut Status beschreibt die Freundschaftsbeziehung näher.
- Eine Person kann beliebig viele öffentliche Status-Updates erstellen, die ihr eindeutig zugeordnet werden. Ein Status-Update umfasst einen Text und den Erstellungszeitpunkt.
- Ein Status-Update kann optional genau einen Anhang besitzen. Jeder Anhang ist dabei eindeutig einem Status-Update zugeordnet. Bei jedem Anhang handelt es sich entweder um ein Bild oder um eine Online-Ressource. Zu einem Bild werden ein lokaler Pfad und die Dateigröße gespeichert. Zur Online-Ressource werden die URL und ein Titel-Text hinterlegt.
- Zu einem Bild kann angegeben werden, welche Personen darauf abgebildet sind. Zusätzlich zu dieser Information sollen keine weiteren Details gespeichert werden.
- Personen können sich gegenseitig Nachrichten schreiben. Zu jeder Nachricht soll der Sender, der Empfänger, der Text der Nachricht und ein Zeitstempel gespeichert werden.
- Verwenden Sie Attribute ID als Primärschlüssel, um die Entitäten zu identifizieren.

a) Entwerfen Sie für das beschriebene Szenario ein ER-Modell in Chen-Notation. Bestimmen Sie hierzu:

   - Die Entity-Typen, die Relationship-Typen und jeweils deren Attribute.
   - Die Funktionalitäten der Relationship-Typen. Achten Sie darauf alle Totalitäten einzutragen.

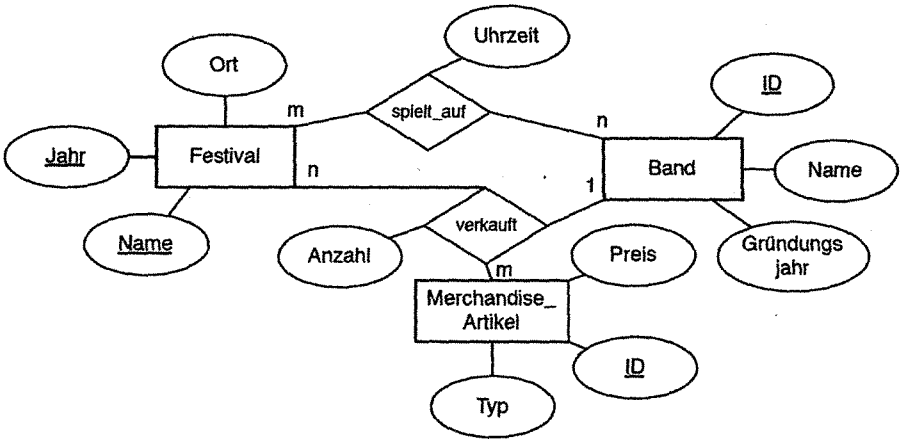
b) Konvertierung einer Generalisierung:

   - Beschreiben Sie kurz die 4 Optionen, wie eine Generalisierung im ER-Diagramm in ein relationales Schema überführt werden kann.
   - Gehen Sie dazu kurz darauf ein, unter welchen Bedingungen diese Optionen anwendbar sind.

Fortsetzung nächste Seite!

Aufgabe 2 SQL-Anfragen

Gegeben sei folgendes ER-Modell, welches eine Festival-Datenbank beschreibt. Es beinhaltet Bands, die auf Festivals spielen und dort Merchandise-Artikel verkaufen. Festivals werden durch ihren Namen, den Veranstaltungsort und das Jahr, in dem sie stattfinden, charakterisiert. Auf Festivals geben verschiedene Bands zu bestimmten Uhrzeiten Konzerte. Bands werden durch eine ID identifiziert und besitzen zusätzlich einen Namen und ihr Gründungsjahr. Die Merchandise-Artikel besitzen ebenfalls eine eindeutige ID, einen Typ (z.B. T-Shirt oder CD) und einen Preis (in Euro):



Gehen Sie von folgendem dazugehörigen relationalem Schema aus:

Festival {*[Name, Jahr, Ort]*}
Band {*[ID, Name, Gründungsjahr]*}
Merchandise_Artikel {*[ID, Preis, Typ]*}
Verkauft {*[Merchandise_Artikel_ID, Festival_Name, Festival_Jahr, Band_ID, Anzahl]*}
Spielt_Auf {*[Band_ID, Festival_Name, Festival_Jahr, Uhrzeit]*}

Formulieren Sie folgende Anfragen in SQL. Achten Sie, falls nötig, auf duplikatfreie Ausgaben:

a) Geben Sie das Gründungsjahr der Band „Metallica“ aus.

b) Formulieren Sie eine Update-Anfrage in SQL, die den Verkauf eines Merchandise-Artikels auf dem Festival mit Namen „Wacken Open Air“ im Jahr 2015 darstellt. Erhöhen Sie dazu den Wert des Attributs „Anzahl“ der Relation „Verkauft“ für den Merchandise-Artikel mit ID = 54.

Fortsetzung nächste Seite!

c) Geben Sie die Uhrzeit des Konzerts von „AC/DC“ aus, welches die Band auf dem Festival „Rock am Ring“ im Jahr 2006 gegeben hat.

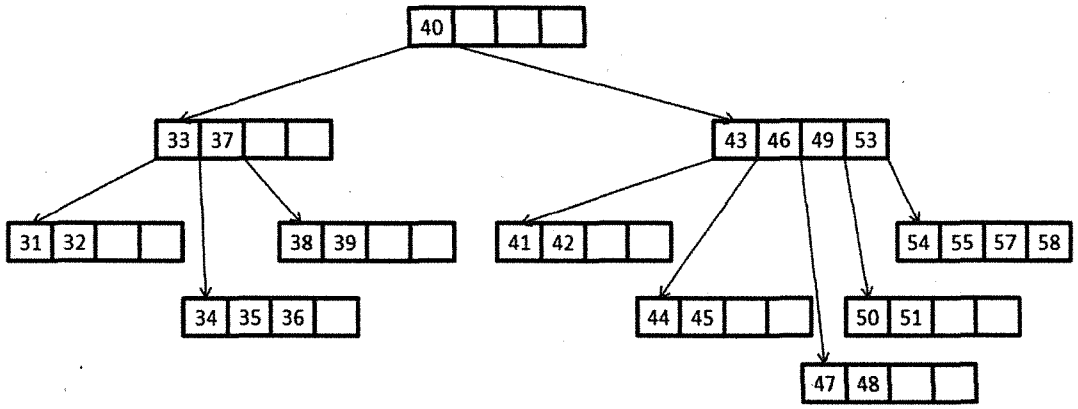
d) Formulieren Sie eine Create-Anfrage in SQL, welche eine Tabelle für die Relation „Spielt_Auf“ erzeugt.

Hinweise:
- Wählen Sie passende SQL-Datentypen.
- Achten Sie auf eventuelle Primär- und Fremdschlüsseldeklarationen.
- Achten Sie auf eventuelle NULL-Werte.

e) Geben Sie die Namen der Bands aus, die durch den Verkauf von Merchandise-Artikeln einen höheren Umsatz als 150 Euro erzielt haben.

Aufgabe 3 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 des Wertes 56

b) Löschen des Wertes 37

Fortsetzung nächste Seite!

Teilaufgabe 2

Aufgabe 1 „Floyd-Hoare-Kalkül (wp-Kalkül)"

Vereinfachen Sie die folgenden Ausdrücke ***formal*** mittels ***wp-Kalkül*** und zwar so weit wie möglich, d.h. ermitteln Sie zu jedem Ausdruck der Form *wp(<Code>, <Nachbedingung>)* die schwächste Vorbedingung des *<Code>*s bzgl. der angegebenen *<Nachbedingung>*. Geben Sie Ihre Umformungen möglichst ausführlich an, d.h. einzelne Zwischenschritte müssen erkennbar und nachvollziehbar sein.

a) $wp(\text{"a -= u-- - - - - - --d;"}, a + u + d > 0)$

b) $wp(\text{" if (a \&\&u > d) d = u+1; else if (d > u) a = !a; else u = d-1;"}, d > u)$

Aufgabe 2 „OOP/OOD - Reverse Engineering"

Leider ist das Klassendiagramm der folgenden Klassen verloren gegangen. Führen Sie ein Reverse Engineering durch und erstellen Sie aus dem Quellcode ein vollständiges UML-Klassendiagramm inklusive aller Klassen, Schnittstellen, Attribute, Methoden, Konstruktoren, Sichtbarkeiten, Assoziationen, Rollennamen, Multiplizitäten, Navigationspfeilen und evtl. Stereotypen. Der Quellcode innerhalb von Methoden und Konstruktoren soll nicht übertragen werden, wohl aber die Methodensignaturen. Assoziationsnamen und deren Leserichtung lassen sich aus dem Quellcode nur schwer erahnen und sollen deshalb ebenfalls weggelassen werden.

```
public abstract class Display implements PixelPainter {
	protected HardwareMatrix hardwareMatrix;
	protected int lastPaintedX;
	protected int lastPaintedY;

	public Display(HardwareMatrix hardwareMatrix) {
		this.hardwareMatrix = hardwareMatrix;
	}

	public int getWidth() {
		return hardwareMatrix.getWidth() / getWidthFactor();
	}

	public int getHeight() {
		return hardwareMatrix.getHeight() / getHeightFac-
tor();
	}

	public void clear() {
		// some longer code
	}

	protected abstract int getWidthFactor();

	protected abstract int getHeightFactor();
}
```

```
import java.awt.Color;

public interface PixelPainter {
	void set(int x, int y, Color color);

	int getHeight();

	int getWidth();
}
```

```
public interface HardwareMatrix {
	void set(int x, int y, int v);

	int getWidth();

	int getHeight();
}
```

Fortsetzung nächste Seite!

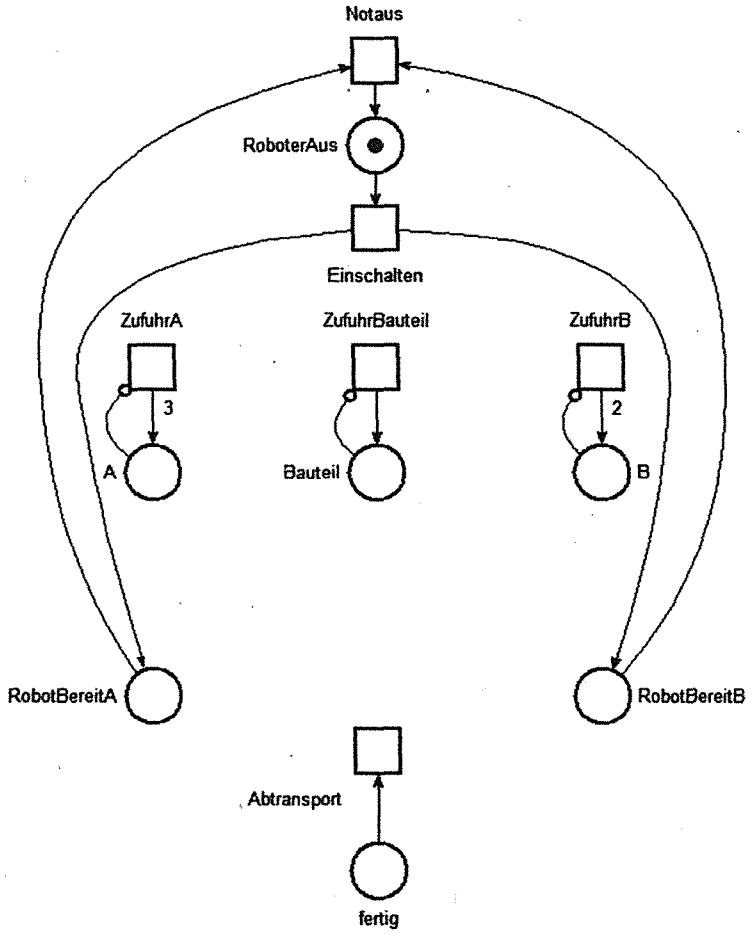
```
import java.awt.Color;

public class DisplayUnion extends RGBDisplay {
	public static final int MAX_DISPLAY_COUNT = 50;
	private int currentDisplayCount;

	private Display[] displays;

	public DisplayUnion(Display[] displays) {
		super(null);
	}

	public int getDisplayCount() {
		return 0;
	}

	protected int getWidthFactor() {
		return 1;
	}

	protected int getHeightFactor() {
		return 1;
	}

	public void set(int x, int y, Color color) {
	}
}
```

```
import java.awt.Color;

public class RGBDisplay extends Display {
	public RGBDisplay(HardwareMatrix matrix) {
		super(matrix);
	}

	public void set(int x, int y, Color color){

	}

	protected int getWidthFactor() {
		return 3;
	}

	protected int getHeightFactor() {
		return 1;
	}
}
```

Fortsetzung nächste Seite!

Aufgabe 3 „Petri-Netze“

Das folgende Grundgerüst stammt aus dem Petri-Netz-Modell einer Automatisierungsanlage mit zwei Robotern, das Sie auf Ihr Blatt übernehmen und geeignet um weitere Plätze (Stellen), Transitionen, Kapazitäten, Gewichte und Markierungen so ergänzen sollen, dass die darunter angegebenen Anforderungen eingehalten werden:



In der Anlage arbeiten zwei Roboter A und B, die über einen Schalter „Einschalten“ aktiviert und ***jederzeit*** über einen „Notaus“-Schalter deaktiviert werden können müssen. Aufgabe der Roboter ist es, jeweils abwechselnd an Bauteilen zu arbeiten, die einzeln über ein Förderband „ZufuhrBauteil“ ins System eingefahren und nach der Fertigstellung mittels „Abtransport“ aus dem System abgeführt werden. Roboter A bringt genau 3 Anbauten vom Typ „A“ an jedes Bauteil an und Roboter B macht das entsprechend mit genau 2 Anbauten vom Typ „B“. Die Anbauten werden jeweils passend über „ZufuhrA“ auf „A“ bzw. mittels „ZufuhrB“ auf „B“ bereitgestellt. Die beiden Roboter ***dürfen niemals*** gleichzeitig an einem Bauteil arbeiten – die Reihenfolge, in der sie darauf zugreifen, ist jedoch beliebig und darf von Bauteil zu Bauteil frei variieren. Sobald einer der Roboter mit der Arbeit an einem neuen Bauteil begonnen hat, darf kein weiteres Bauteil angenommen werden, ehe der jeweils andere Roboter nicht ebenfalls mit seiner Arbeit am gleichen Bauteil fertig geworden ist (und das fertige Bauteil „auf den Platz ‚fertig‘ legt“). Aus Platzgründen darf höchstens ein Bauteil auf dem Ausgangsplatz „fertig“ abgestellt werden, ehe es abtransportiert wird.