Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer

Kennzahl: _____

Kennwort: _____

Arbeitsplatz-Nr.: _____

Herbst
2022

66115

Erste Staatsprüfung für ein Lehramt an öffentlichen Schulen

— Prüfungsaufgaben —

Fach:	**Informatik (vertieft studiert)**
Einzelprüfung:	**Theoretische Informatik, Algorithmen**
Anzahl der gestellten Themen (Aufgaben):	2
Anzahl der Druckseiten dieser Vorlage:	21

Bitte wenden!

Thema Nr. 1
(Aufgabengruppe)

Es sind alle Aufgaben dieser Aufgabengruppe zu bearbeiten!

Teilaufgabe I: Algorithmen

Aufgabe 1 (Laufzeitanalyse)

[20 PUNKTE]

Gegeben sei der folgende Suchalgorithmus. Als Eingabe erhält er ein Array E mit n Einträgen $E[0]$ bis $E[n-1]$ sowie einen Schlüssel k, für den er überprüft, ob er in dem Eingabe-Array enthalten ist.

```
1  bool search(int E[], int n, int k) {
2    int left = 0, right = n - 1;
3    while (left <= right) {
4      if (E[left] == k || E[right] == k) {
5        return true;
6      }
7      left = left + 1;
8      right = right - 1;
9    }
10   return false;
11 }
```

Gehen Sie von einer nicht strikten Auswertung von booleschen Ausdrücken aus (wie es auch in Java üblich ist), d.h. es wird die rechte Seite eines oder-Ausdruckes (||) nicht mehr ausgewertet, wenn die linke Seite bereits als wahr ausgewertet wurde. Es findet somit innerhalb der if-Bedingung der rechte Vergleich nicht statt, wenn bereits der linke Vergleich zutrifft.

Im Folgenden sollen Sie die *exakte* Anzahl der Vergleiche unter verschiedenen Randbedingungen ermitteln. Zählen Sie dabei nur Vergleiche mit Einträgen im Array und nicht Vergleiche, die etwa bei der Prüfung der Schleifenbedingung stattfinden. Sie können darüber hinaus annehmen, dass n eine gerade Zahl ist.

(a) Bestimmen Sie in Abhängigkeit von n die *exakte* Anzahl der Vergleiche im schlechtesten Fall als Funktion $W(n)$. Begründen Sie Ihre Antwort.

(b) Bestimmen Sie in Abhängigkeit von n die *exakte* Anzahl der Vergleiche im besten Fall als Funktion $B(n)$. Begründen Sie Ihre Antwort.

(c) Bestimmen Sie in Abhängigkeit von n nun die *exakte* Anzahl der Vergleiche im durchschnittlichen Fall (Average Case) als Funktion $A(n)$. Gehen Sie davon aus, dass der gesuchte Schlüssel mit einer Wahrscheinlichkeit von 0,3 im Array enthalten ist und jeder Schlüssel maximal einmal im Array vorkommt. Begründen Sie Ihre Antwort.

Aufgabe 2 (Sortierverfahren) [29 PUNKTE]

In der folgenden Aufgabe soll ein Feld E von ganzen Zahlen aufsteigend sortiert werden. Das Feld habe n Elemente E[0] bis E[n−1]. Der folgende Algorithmus sei gegeben:

```
1 void sort (int E[], int left, int right) {
2   for (int k = 1; k <= right - left; k = 2*k) {
3     for (int i = left; i+k <= right; i = i+2*k) {
4       merge(E, i, i+k, min(right, i+2*k-1));
5     }
6   }
7 }
```

Der initiale Aufruf des Algorithmus lautet: sort(E,0,n−1)

Die Funktion merge(E,a,m,e) erstellt aus den beiden sortierten Teilfeldern E[a,m−1] und E[m,e] ein sortiertes Teilfeld E[a,e] durch „Verschmelzen".

(a) Sortieren Sie das folgende Feld E mittels des Algorithmus. Geben Sie nach jedem Durchlauf der *inneren* Schleife jeweils die Werte von k, i und die Belegung des Feldes in einer neuen Zeile an.

Index	0	1	2	3	4	5	6	7
Wert	5	9	3	7	4	1	6	8

(b) Beschreiben Sie in eigenen Worten die Rolle der Variable k im Rahmen des Algorithmus.

(c) Erläutern Sie, warum beim Aufruf von merge in Zeile 4 im letzten Parameter das Minimum zwischen right und i+2*k−1 gebildet wird. Warum kann man nicht einfach merge(i, i+k, i+2*k−1) aufrufen?

(d) Schätzen Sie die Anzahl der Aufrufe von merge in Abhängigkeit von der Länge des Feldes möglichst gut mit Hilfe der O-Notation ab. Begründen Sie Ihre Antwort.

(e) Der iterative Algorithmus soll in einen rekursiven Algorithmus nach dem folgenden Schema umgewandelt werden:

```
1 void sort (int E[], int left, int right) {
2   if ⟨Abbruchbedingung⟩ return;
3   int m = ⌊log2(right - left)⌋;
4   ⟨erster rekursiver Aufruf von sort⟩
5   ⟨zweiter rekursiver Aufruf von sort⟩
6   merge(E, left, m, right);
7 }
```

Geben Sie die Abbruchbedingung sowie die Parameter des ersten und zweiten rekursiven Aufrufs von sort an. Begründen Sie Ihre Antwort.

Aufgabe 3 (Streuspeicherung) [24 PUNKTE]

Gegeben seien die folgenden Schlüssel k zusammen mit ihren Streuwerten $h(k)$:

k	A	B	C	D	E	F	G	H
$h(k)$	1	1	7	5	3	1	5	5

(a) Fügen Sie die Schlüssel in der angegebenen Reihenfolge (von links nach rechts) in eine Streutabelle mit 8 Fächern ein und lösen Sie Kollisionen durch verkettete Listen auf.

Stellen Sie die Streutabelle in folgender Art und Weise dar (Streutabelle bitte auf Bearbeitungsbogen übertragen):

Fach	Schlüssel k (verkettete Liste, zuletzt eingetragener Schlüssel rechts)
0	
1	
2	
3	
4	
5	
6	
7	

(b) Fügen Sie die gleichen Schlüssel in der gleichen Reihenfolge und mit der gleichen Streufunktion in eine neue Streutabelle mit 8 Fächern ein. Lösen Sie Kollisionen diesmal aber durch lineares Sondieren mit Schrittweite +3 auf.

Geben Sie zu jedem Schlüssel an, welche Fächer Sie in welcher Reihenfolge sondiert haben und wo der Schlüssel schlussendlich gespeichert wird.

(c) Bei der doppelten Streuadressierung verwendet man eine Menge h_i von Streufunktionen. Anfangs wird Funktion h_0 verwendet. Im Falle einer Kollision wird die jeweils nächste Streufunktion verwendet. Allgemein wird nach der i-ten Kollision die Streufunktion h_i verwendet.

Die Streufunktionen h_i sind wie folgt gegeben:

k	A	B	C	D	E	F	G	H
$h_0(k)$	1	1	7	5	3	1	5	5
$h_1(k)$	3	6	4	5	7	2	6	0
$h_2(k)$	5	3	1	5	3	3	7	3
$h_3(k)$	7	0	6	5	7	4	0	6
$h_4(k)$	1	5	3	5	3	5	1	1
$h_5(k)$	3	2	0	5	7	6	2	4

Fügen Sie die Schlüssel in der angegebenen Reihenfolge (von links nach rechts) in eine Streutabelle der Größe 8 ein. Geben Sie jeweils an, welche Streufunktion Sie verwendet haben.

Aufgabe 4 (Backtracking) [33 PUNKTE]

Sie sollen in dieser Aufgabe einen Algorithmus entwickeln, der die Lösung für ein sogenanntes *magisches Quadrat* berechnet. Dieses besteht aus 3×3 Feldern, die mit den Ziffern von 1 bis 9 so belegt werden, dass

- jede Ziffer nur einmal vorkommt,
- die Summen aller Zeilen, aller Spalten und der beiden Diagonalen denselben Wert ergeben, und
- alle Felder belegt sind.

Sie sollen nun einen Algorithmus beschreiben, der *alle* Lösungen durch Backtracking findet. Zweckmäßigerweise definiert man dazu eine *Teillösung* als ein teilweise von oben links belegtes Quadrat, wie z.B.:

4	9	2
3	5	

Eine Teillösung definieren wir als nützlich, wenn:

- jede Ziffer in den belegten Feldern höchstens einmal vorkommt, und
- die Summen aller vollständig belegten Zeilen, Spalten und Diagonalen denselben Wert ergeben.

(a) Welche der drei folgend gezeigten Teillösungen (links, mitte, rechts) sind nützlich? Begründen Sie Ihre Antwort.

4	9	2
3	5	7

4	9	

4	9	2
3	7	5
8		

(b) Ihr Algorithmus soll durch rekursive Aufrufe einen Suchbaum aufbauen, der alle nützlichen Teillösungen systematisch erkundet. Welche nützliche Teilösung ist sinnvollerweise die Wurzel des Suchbaums? Welches sind die Kinder einer Teillösung im Suchbaum? Begründen Sie Ihre Antwort.

(c) Wann ist eine nützliche Teillösung eine Gesamtlösung? Begründen Sie Ihre Antwort.

(d) Das magische Quadrat sei in dem Array Q[3][3] von ganzen Zahlen gegeben. Geben Sie einen Algorithmus sumsOK(Q, f) an, der für eine Teillösung Q, die bis zum Feld f bereits gefüllt ist, berechnet, ob die Summen-Bedingung gilt oder nicht. Der Index des Feldes f soll wie folgt interpretiert werden:

0	1	2
3	4	5
6	7	8

Sie dürfen Ihren Algorithmus umgangssprachlich, in Pseudo-Code oder in einer Programmiersprache Ihrer Wahl beschreiben. Einzige Bedingung ist, dass daraus die Vollständigkeit und Korrektheit ersichtlich sein muss.

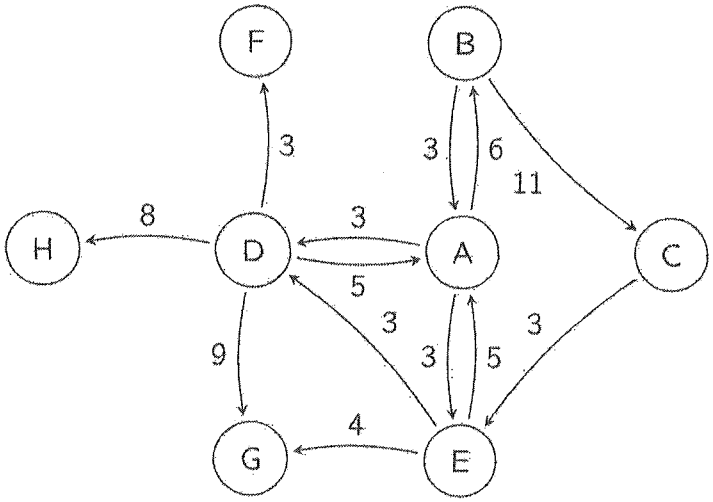
(e) Das magische Quadrat soll in dem Array Q[3][3] von ganzen Zahlen gespeichert sein. Geben Sie einen rekursiven Algorithmus magicSquare(Q, next) an, der ausgehend von einer nützlichen Teillösung Q, in der die Feldindizes des Feldes f 0 bis next−1 bereits belegt sind, mittels Backtracking alle möglichen magischen Quadrate aufzählt. Sie dürfen die Funktion sumsOK(Q,f) aus dem vorherigen Aufgabenteil als gegeben verwenden.

Sie dürfen Ihren Algorithmus umgangssprachlich, in Pseudo-Code oder in einer anderen Programmiersprache Ihrer Wahl beschreiben. Einzige Bedingung ist, dass daraus die Vollständigkeit und Korrektheit ersichtlich sein muss.

Der initiale Aufruf soll lauten: magicSquare(Q,0), wobei Q die Wurzel des Suchbaums ist (siehe Teilaufgabe b).

Aufgabe 5 (Kürzeste Wege) [14 PUNKTE]

Gegeben ist der folgende, gerichtete Graph mit Kanten- bzw. Entfernungsgewichten.



Bestimmen Sie die kürzesten Wege des Graphen vom Startknoten A aus zu allen übrigen Knoten mit Hilfe des Algorithmus von Dijkstra. Falls mehrere Knoten für die nächste Iteration zur Wahl stehen, werden die Knoten dabei in alphabetischer Reihenfolge betrachtet. Verwenden Sie dazu auf Ihrem Bearbeitungsbogen eine Tabelle der folgenden Art (die erste Spalte ist bereits ausgefüllt) und markieren Sie in jeder Zeile den jeweils als nächstes zu betrachtenden Knoten. Fassen Sie das Endergebnis für alle Knoten in geeigneter Weise zusammen.

Knoten	A							
B	6							
C	∞							
D	3							
E	3							
F	∞							
G	∞							
H	∞							

Teilaufgabe II: Theoretische Informatik

Aufgabe 1 (Reguläre und nicht-reguläre Sprachen) [46 PUNKTE]

(a) Geben Sie einen nichtdeterministischen endlichen Automaten an, der genau die Sprache

$$L_1 := \{uabv \mid u, v \in \{a, b\}^*, |v| = 2\}$$

akzeptiert und nicht mehr als 6 Zustände besitzt.

(b) Geben Sie einen regulären Ausdruck an, der das Komplement von L_1 erzeugt, d.h. die erzeugte Sprache ist

$$\overline{L_1} = \{w \mid w \in \{a, b\}^*, w \notin L_1\}.$$

Erläutern Sie, warum der von Ihnen angegebene reguläre Ausdruck die Sprache $\overline{L_1}$ erzeugt.

(c) Entscheiden Sie für jede der folgenden formalen Sprachen, ob diese regulär oder nichtregulär sind. Beweisen Sie jeweils die Korrektheit Ihrer Antwort.

(i) $L_2 := \{vuv \mid u, v \in \{a, b\}^*, |v| > 9\}$

(ii) $L_3 := \{uv \mid u \in \{a, b\}^*, v \in \{a, b, c\}^*, |v| > 0\}$

(iii) $L_4 := L(G)$ mit $G = (\{A, B\}, \{a, b\}, \{A \to c, A \to BAB, B \to a, B \to b\}, A)$

Aufgabe 2 (Typ-1- und Typ-2-Sprachen) [30 PUNKTE]

(a) Es sei

$$L_5 := \{b^i a^k b^i \mid i, k \in \mathbb{N}_0, k < i, i > 0\}.$$

Verwenden Sie das Pumpinglemma für kontextfreie Sprachen, um zu widerlegen, dass L_5 kontextfrei ist.

(b) Sei $G = (\{A, B, C, D\}, \{a, b\}, P, \{A\})$ eine kontextfreie Grammatik (in Chomsky-Normalform), wobei die Produktionen P gegeben sind durch:

$$\begin{aligned} P = \{\ & A \to DD, \\ & A \to CA, \\ & B \to AC, \\ & B \to a, \\ & C \to CD, \\ & C \to CA, \\ & C \to a, \\ & C \to b, \\ & D \to BC, \\ & D \to a\} \end{aligned}$$

Prüfen Sie mithilfe des Algorithmus von Cocke, Younger und Kasami (CYK-Algorithmus), ob die Grammatik G das Wort *ababaa* erzeugt.

Geben Sie dafür die entsprechende durch den CYK-Algorithmus berechnete Tabelle an und dokumentieren Sie deren Berechnung in geeigneter Weise.

Geben Sie an, ob das Wort von der Grammatik erzeugt wird und wie dies anhand der Tabelle ermittelt wird.

Aufgabe 3 (Entscheidbarkeit) [24 PUNKTE]

Im Folgenden bezeichne $\langle M \rangle$ die Gödelnummer der Turingmaschine M. Für ein Wort $w = a_1 \dots a_n$ bezeichne w^R das umgekehrte Wort $w^R = a_n \dots a_1$. Das Halteproblem H ist definiert als

$$H := \{\langle M \rangle \# w \mid \text{Turingmaschine } M \text{ hält bei Eingabe } w\}.$$

Das Halteproblem auf leerer Eingabe H_0 ist definiert als

$$H_0 := \{\langle M \rangle \mid \text{Turingmaschine } M \text{ hält bei leerer Eingabe}\}.$$

Es ist bekannt, dass H und H_0 unentscheidbar sind.

Beweisen Sie für jede der folgenden Sprachen, dass diese ebenfalls unentscheidbar sind, indem Sie jeweils H oder H_0 auf die Sprache reduzieren.

(a) $L_6 = \{\langle M_1 \rangle \# \langle M_2 \rangle \mid$ Turingmaschine M_1 hält auf der leeren Eingabe genau dann, wenn Turingmaschine M_2 nicht auf der leeren Eingabe hält $\}$

(b) $L_7 = \{\langle M \rangle \# w \mid$ Turingmaschine M hält bei Eingabe w mit der Ausgabe w^R $\}$

Aufgabe 4 (NP-Vollständigkeit) [20 PUNKTE]

Eine 3-CNF ist eine Menge von Klauseln, wobei jede Klausel eine Menge aus **0, 1, 2 oder 3** Literalen ist und kein Literal sowohl positiv als auch negativ in derselben Klausel vorkommt. Ein positives Literal ist eine aussagenlogische Variable x_i. Ein negatives Literal ist eine negierte aussagenlogische Variable $\neg x_i$.

Z.B. ist $F = \{\{x_1, x_2, \neg x_3\}, \{x_1, x_2\}, \{\neg x_1\}\}$ eine 3-CNF.

Eine 3-CNF, in der genau die Variablen $\{x_1, \ldots, x_n\}$ vorkommen, ist **erfüllbar**, wenn es eine Belegung $B : \{x_1, \ldots, x_n\} \to \{0, 1\}$ gibt, sodass in jeder Klausel für mindestens ein Literal L gilt: $B(L) = 1$. Dabei gilt

$$B(\neg x_i) = \begin{cases} 1, & \text{wenn } B(x_i) = 0 \\ 0, & \text{wenn } B(x_i) = 1 \end{cases}$$

Die 3-CNF F ist erfüllbar, was z.B. die Belegung B mit $B(x_1) = 0, B(x_2) = 1$ und $B(x_3) = 1$ bezeugt.

Die Entscheidungsprobleme 3-CNF-SAT und 3-CNF-SAT-VD sind in gegeben/gefragt-Notation wie folgt definiert:

3-CNF-SAT

gegeben: Eine 3-CNF F

gefragt: Gibt es eine Belegung B, die F erfüllt?

3-CNF-SAT-VD

gegeben: Eine 3-CNF F, wobei jede Variable in F mindestens zweimal vorkommt

gefragt: Gibt es eine Belegung B, die F erfüllt?

(a) Zeigen Sie, dass 3-CNF-SAT in Polynomialzeit auf 3-CNF-SAT-VD reduziert werden kann (was oft als 3-CNF-SAT $\leq_p$ 3-CNF-SAT-VD geschrieben wird).

(b) Zeigen Sie, dass 3-CNF-SAT-VD NP-vollständig ist. Dabei dürfen Sie als bekannt annehmen, dass 3-CNF-SAT NP-vollständig ist. Das Resultat aus Aufgabenteil a) darf verwendet werden.

Thema Nr. 2
(Aufgabengruppe)

Es sind alle Aufgaben dieser Aufgabengruppe zu bearbeiten!

Teilaufgabe I: Algorithmen

Aufgabe 1 (Anzahlen suchen) [20 PUNKTE]

Gegeben sei ein Feld $a[1], \ldots, a[m]$ von m Zahlen aus dem Bereich $\{1, \ldots, n\}$, wobei Zahlen mehrfach vorkommen dürfen. Ziel dieser Aufgabe ist der Entwurf von Algorithmen, welche bestimmen, wieviele Vorkommen einer Zahl $i \in \{1, \ldots, n\}$ im Feld a vorhanden sind.

(a) Geben Sie einen Algorithmus in Pseudocode an, der für das gegebene Feld a und die gegebene Zahl i berechnet, wie oft i im Feld a vorkommt. Ihr Algorithmus soll dabei mit (asymptotischer) Linearzeit in der Größe des Feldes auskommen (d.h. die Laufzeit soll in $O(m)$ liegen).

Erläutern Sie die Funktionsweise Ihres Algorithmus und analysieren Sie die Laufzeit und den Platzbedarf Ihres Algorithmus für den Worst-Case.

(b) Nehmen Sie nun an, dass das Feld a **aufsteigend sortiert** ist und geben Sie einen Algorithmus in Pseudocode für diesen Fall an. Dabei soll die Laufzeit nun echt besser als linear in m sein (d.h. die Laufzeit soll in $o(m)$ sein).

Erläutern Sie die Funktionsweise Ihres Algorithmus und analysieren Sie die Laufzeit und den Platzbedarf Ihres Algorithmus für den Worst-Case.

Aufgabe 2 (Laufzeitkomplexität) [30 PUNKTE]

(a) Bestimmen Sie für jeden der folgenden Algorithmen eine möglichst gute obere Schranke für die asymptotische Laufzeit mithilfe der O-Notation in Abhängigkeit des Parameters n. Nehmen Sie dabei an, dass arithmetische Operationen auf Zahlen wie Multiplikationen, Additionen und Vergleiche in konstanter Zeit für jede der auftretenden Zahlen ausführbar sind.

Geben Sie jeweils eine kurze Begründung für Ihre Antwort.

(i)
$i := 1;$

$j := 1;$

while $i \leq n$ **do**

while $j \leq n$ **do**

$j := j + 2;$

end-while

$i := i + 1;$

end-while

(ii)
$i := 1;$

while $i \leq n$ **do**

$j := 1;$

while $j \leq n$ **do**

$j := j + 2;$

end-while

$i := i + 1;$

end-while

(iii)
$l := 1;$

$i := 5;$

while $n \cdot l < i$ und $i < n^2$ **do**

$l := l + 1;$

$i := i + 1;$

end-while

(iv)
```
l := 0;
r := 0;
i := 1;
while i < 2^n do
    r := r + i;
    i := 2 · i;
    j := 1;
    while j < n do
        l := l + j;
        j := j + j;
    end-while
end-while
```

(b) Ordnen Sie die folgenden Funktionen in der Reihenfolge ihres asymptotischen Wachstums, so dass $f_i \in O(f_{i+1})$ gilt:

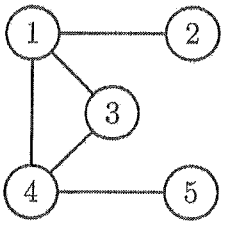
$$n^{6n}, \quad 10n^3 - 2n^2 + 10, \quad 4n^2(\log_2 n^5), \quad 6^n, \quad 2n^3(\log_3 n^3)$$

Aufgabe 3 (Graphen) [40 PUNKTE]

Ein ungerichteter Graph $G = (V, E)$ mit n Knoten $V = \{1, \ldots, n\}$ sei durch eine Adjazenzmatrix a gegeben, sodass gilt:

- $a[i][j] = 1$ wenn $i < j$ und $\{i, j\} \in E$ und
- $a[i][j] = 0$ sonst ($\{i, j\} \notin E$ oder $i \geq j$).

(a) Geben Sie die Adjazenzmatrix zu dem als Diagramm gezeigten Graph G_0 an, indem Sie die Adjazenzmatrix a auf Ihren Bearbeitungsbogen übertragen und mit Nullen und Einsen füllen.



Graph G_0

$i \backslash j$	1	2	3	4	5
1					
2					
3					
4					
5					

Adjazenzmatrix a (ungefüllt)

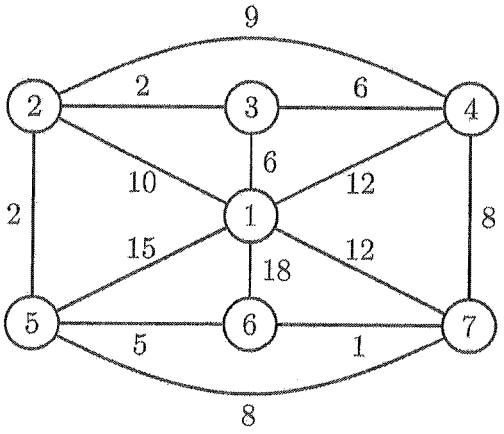
(b) Der **Komplementgraph** eines Graphen $G = (V, E)$ ist der Graph $\overline{G} = (V, \overline{E})$ mit $\overline{E} = \{\{u, v\} \mid u, v \in V, \{u, v\} \notin E\}$.

Geben Sie den Komplementgraph $\overline{G_0}$ zum Graph G_0 aus Aufgabenteil a) sowohl als Diagramm als auch in der Adjazenzmatrix-Darstellung an.

(c) Geben Sie einen Algorithmus in Pseudocode an, der für eine gegebene Adjazenzmatrix a die Adjazenzmatrix des Komplementgraphs berechnet und diese zurückgibt. Erläutern Sie die Funktionsweise Ihres Algorithmus und schätzen Sie die Laufzeit Ihres Algorithmus asymptotisch ab.

(d) Berechnen Sie für den folgenden gewichteten Graphen G_1 einen minimalen Spannbaum, indem Sie entweder den Algorithmus von Prim mit Startknoten 1 oder den Algorithmus von Kruskal verwenden. Dokumentieren Sie dabei Ihre Berechnung für den Graph G_1 schrittweise.

Erläutern Sie vorher kurz, wie der verwendete Algorithmus (von Prim oder Kruskal) vorgeht. Hier genügt eine informelle Beschreibung, wie der Algorithmus beginnt, welche Schritte er macht und wann er (mit einem minimalen Spannbaum als Resultat) terminiert.



Graph G_1

Aufgabe 4 (Dynamisches Programmieren) [30 PUNKTE]

Ein Busunternehmen habe n verschiedene Aufträge für eine Fahrt von Hamburg nach München am 1. Januar. Jeder Auftrag ist mit einer festen Personenzahl z_i $(i = 1, \ldots n)$ verknüpft, die transportiert werden müssen, und mit einem Preis p_i, welcher pauschal für den Auftrag gezahlt wird. Jeder Auftrag kann entweder ganz oder gar nicht durchgeführt werden.

Das Busunternehmen kann insgesamt m Personen transportieren (dies ist die ganzzahlige Anzahl an Fahrgastplätzen in allen Bussen zusammen).

Das algorithmische Problem ist es, jene Aufträge auszusuchen, die den Umsatz des Unternehmens maximieren.

(a) Der folgende Greedy-Algorithmus fügt die Aufträge mit dem höchsten Preis solange hinzu, bis die Plätze in den Bussen verteilt wurden:

kapazitaet := m;

umsatz := 0;

auftraege := $\{1, \ldots, n\}$;

while auftraege $\neq \emptyset$ **do**

bestimme i aus auftraege mit maximalen p_i-Wert;

auftraege := auftraege $\setminus \{i\}$;

if $z_i \leq$ kapazitaet **then**

kapazitaet := kapazitaet $-z_i$;

umsatz := umsatz $+p_i$;

end-if

end-while

Zeigen Sie, dass der Greedy-Algorithmus nicht immer ein optimales Ergebnis für den Umsatz bestimmt, indem Sie ein Gegenbeispiel angeben und erläutern, warum dieses zu einem nicht-optimalen Ergebnis führt.

(b) Stellen Sie eine Rekursionsgleichung zur exakten Berechnung des maximalen Umsatzes auf. Dafür soll die Funktion Umsatz(i,j) rekursiv formuliert werden, sodass gilt

(i) Der Basisfall Umsatz(1,j) legt den Umsatz fest, wenn nur noch über die Ausführung des Auftrages 1 entschieden werden darf und noch j Plätze in den Bussen zur Verfügung stehen.

(ii) Der allgemeine Fall Umsatz(i,j) für $i > 1$ legt den Umsatz (rekursiv!) fest, wenn die Aufträge $1, \ldots, i$ noch für die Ausführung gewählt werden können und noch j Plätze in den Bussen zur Verfügung stehen.

Erläutern Sie die Funktionsweise Ihrer Rekursionsgleichung.

(c) Verwenden Sie dynamische Programmierung, um den maximalen Umsatz effizient zu berechnen. Geben Sie den zugehörigen Algorithmus in Pseudocode an, wobei:

- Sie ein Feld $u[i][j]$ für die Aufträge $i = 1, \ldots, n$ bei $j = 0, \ldots, m$ Sitzplätzen verwenden, sodass nach Ausführung Ihres Algorithmus in jeder Zelle $u[i][j]$ der Wert von Umsatz(i, j) gespeichert ist;
- der Algorithmus zunächst jene Einträge im Feld u zuweist, deren Wert *ohne Rückgriff auf andere Einträge in* u bestimmt werden kann;
- der Algorithmus anschließend das Feld u geschickt durchläuft, sodass der jeweilige aktuell berechnete Feldeintrag durch Rückgriff auf *bereits berechnete* Werte im Feld u bestimmt werden kann.

Bestimmen Sie anschließend die asymptotische Laufzeit und Platzkomplexität im Worst-Case für Ihren Algorithmus.

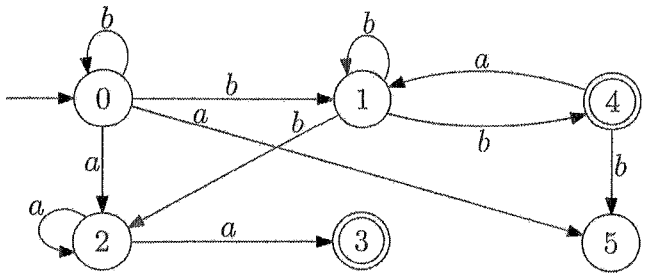
Teilaufgabe II: Theoretische Informatik

Aufgabe 1 (Reguläre Sprachen) [20 PUNKTE]

(a) Geben Sie einen deterministischen endlichen Automaten an, der die Sprache aller Wörter über dem Alphabet $\{a, b, c\}$ akzeptiert, die mindestens ein c, aber nicht das Teilwort aab enthalten. Es genügt, den Automaten durch ein Zustandsübergangsdiagramm anzugeben.

(b) Geben Sie einen kurzen regulären Ausdruck an, der alle Wörter über dem Alphabet $\{0, 1\}$ beschreibt, die das Teilwort 11 enthalten und nicht mit 10 beginnen.

(c) Wandeln Sie den gegebenen nichtdeterministischen Automaten in einen deterministischen endlichen Automaten um. Verwenden Sie dafür die Potenzmengenkonstruktion. Es genügt, den DEA in tabellarischer Form anzugeben.



Aufgabe 2 (Chomsky-Hierarchie) [30 PUNKTE]

Bestimmen Sie für die folgenden Sprachen über $\Sigma = \{a, b, c\}$ jeweils, ob sie regulär, kontextfrei und/oder kontextsensitiv sind. Beweisen Sie Ihre Antworten. Für jede zutreffende Eigenschaft genügt es, eine geeignete Beschreibung (Grammatik/regulärer Ausdruck) oder die Arbeitsweise eines geeigneten Akzeptors (Automat/Maschine) ohne Korrektheitsbeweis anzugeben. Hierbei bezeichnet der %-Operator die Modulo-Operation.

(a) $L_1 = \{a^i b^j c^k \mid i, j, k \in \mathbb{N}_0, j = 2i \text{ und } k = 3i\}$

(b) $L_2 = \{a^i b^j c^k \mid i, j, k \in \mathbb{N}_0, j = 2i \text{ oder } k = 3i\}$

(c) $L_3 = \{a^i b^j c^k \mid i, j, k \in \mathbb{N}_0, j = i\%2 \text{ und } k = i\%3\}$

Aufgabe 3 (Kontextfreie Sprachen) [17 PUNKTE]

(a) Gegeben sei die folgende Grammatik $G = (V, \Sigma, P, S)$ über dem Alphabet $\Sigma = \{a, b, c\}$, der Variablenmenge $\{S, A, B, C, D\}$, dem Startsymbol S und der folgenden Produktionsmenge P:

$$\begin{aligned} P = \{\ & S \to AB, \\ & A \to CD \mid CB, \\ & B \to DB \mid b, \\ & C \to AC \mid a, \\ & D \to b \mid c\} \end{aligned}$$

Entscheiden Sie anhand des CYK-Algorithmus, ob die von G erzeugte Sprache das Wort *abacbb* enthält.

(b) Zeigen Sie, dass die kontextfreien Sprachen nicht unter Schnitt und Komplement abgeschlossen sind.

Hinweis: Sie können ohne Beweis verwenden, dass $L = \{a^n b^n c^n \mid n \in \mathbb{N}\}$ nicht kontextfrei ist.

Aufgabe 4 (Entscheidbarkeit) [30 PUNKTE]

Bestimmen Sie für die folgenden Sprachen, ob sie entscheidbar sind und beweisen Sie jeweils Ihre Antwort. Dabei bezeichnet $\langle M \rangle$ die Gödelnummer der Turingmaschine M.

(a) $L_1 = \{\langle M \rangle \# x \mid M$ akzeptiert mindestens eine Eingabe in höchstens $|x|^3$ Schritten$\}$

(b) $L_2 = \{\langle M \rangle \# w \mid \exists n \in \mathbb{N}_0 \colon M$ akzeptiert das Wort $w^n\}$

(c) $L_3 = \{\langle M \rangle \mid$ Es gibt ein $w \neq \varepsilon$, sodass M alle Wörter w^n mit $n \in \mathbb{N}_0$ akzeptiert$\}$

Aufgabe 5 (Komplexitätstheorie) [23 PUNKTE]

Sei $G = (R \cup B, E)$ ein Graph, dessen Knoten mit den Farben rot und blau gefärbt sind. Für eine gerade Zahl k ist ein *bunter Haarball der Größe k* in G eine Menge von $k/2$ roten Knoten (aus R) und $k/2$ blauen Knoten (aus B) in G, sodass

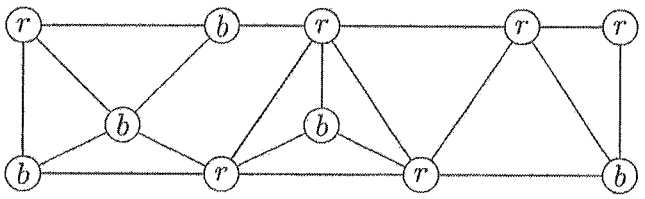
(i) die $k/2$ roten Knoten paarweise miteinander verbunden sind,

(ii) die $k/2$ blauen Knoten paarweise nicht miteinander verbunden sind,

(iii) jeder der $k/2$ roten Knoten zu genau einem der $k/2$ blauen Knoten verbunden ist, und

(iv) jeder der $k/2$ blauen Knoten zu genau einem der $k/2$ roten Knoten verbunden ist.

(a) Im nachfolgend abgebildeten Graphen G sind rote Knoten mit r und blaue Knoten mit b beschriftet. Geben Sie einen bunten Haarball der Größe 6 in G an. Übertragen Sie die unten stehende Abbildung auf Ihren Bearbeitungsbogen und markieren Sie hierzu alle Kanten, die zwei Knoten des Haarballs miteinander verbinden.



(b) Das Problem BUNTER HAARBALL ist wie folgt definiert:

Problem BUNTER HAARBALL:	
Eingabe:	Ein Graph $G = (R \cup B, E)$, mit roten und blauen Knoten und eine positive gerade Zahl k.
Frage:	Enthält G einen bunten Haarball der Größe mindestens k?

Zeigen Sie, dass das Problem BUNTER HAARBALL NP-vollständig ist.

Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer

Kennzahl: _____

Kennwort: _____

Arbeitsplatz-Nr.: _____

Herbst
2022

66116

Erste Staatsprüfung für ein Lehramt an öffentlichen Schulen

— Prüfungsaufgaben —

Fach:	**Informatik (vertieft)**
Einzelprüfung:	**Datenbanksysteme, Softwaretechnologie**
Anzahl der gestellten Themen (Aufgaben):	2
Anzahl der Druckseiten dieser Vorlage:	19

Bitte wenden!

Thema Nr. 1
(Aufgabengruppe)

Es sind alle Aufgaben dieser Aufgabengruppe zu bearbeiten!

Teilaufgabe I: Datenbanksysteme

Aufgabe 1 (ER-Modellierung) [34 PUNKTE]

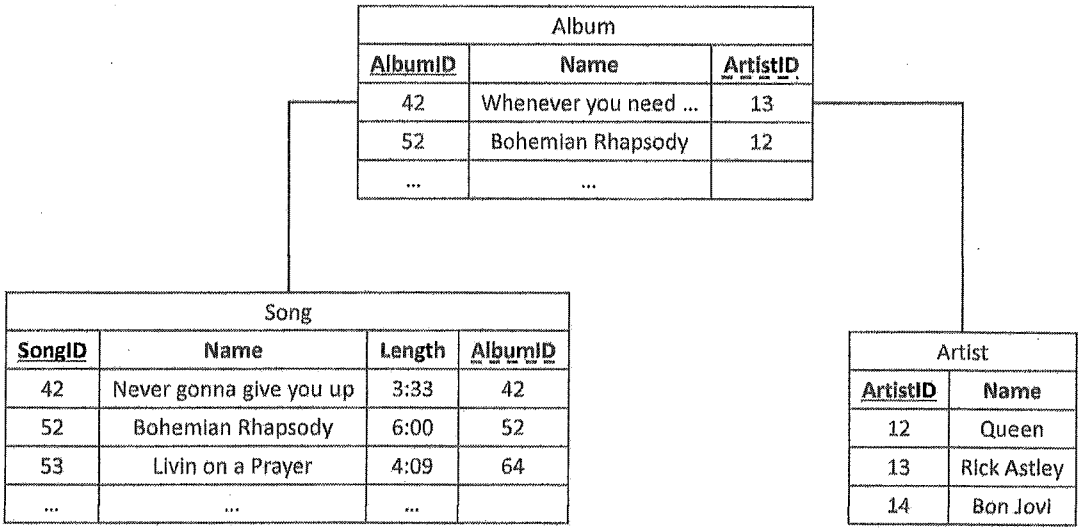
Entwerfen Sie ein Entity-Relationship Diagramm, welches das gegebene Szenario beschreibt. Dabei sind, sofern nicht explizit angegeben, alle Attribute nicht eindeutig. Wählen Sie für die Angabe der Kardinalitäten eine sinnvolle Notation aus und führen Sie jeweils einen künstlichen Schlüsselkandidaten ein, wenn nötig.

Es gibt Kinos. Diese werden mit Name und Adresse gespeichert. Zu jedem Kino gehören mindestens zwei Kinosäle. Diese werden mit Größe und einer innerhalb eines Kinos eindeutigen Nummer gespeichert. Weiterhin gibt es Mitarbeiter, deren Namen gespeichert werden. Jeder Mitarbeiter arbeitet in genau einem Kino und ist dort für mindestens zwei Kinosäle zuständig. Dabei muss jeder Kinosaal mindestens einen Mitarbeiter haben, der sich um den Saal kümmert.

Weiterhin werden Kunden mit ihren Namen gespeichert. Außerdem werden Filme mit eindeutigen Namen und dem zugehörigen Preis gespeichert. Jeder Film wird in die Kategorien Action, Comedy oder Other eingeordnet. Zu jedem Film gibt es mindestens eine Vorstellung. Diese Vorstellungen werden mit Datum und Vorstellungsnummer des jeweiligen Films gespeichert. Zuletzt soll modelliert werden, dass Kunden diese Vorstellungen in einem Kinosaal anschauen können. Dabei hat eine Vorstellung mindestens 30 Kunden und ein Kunde muss in mindestens einer Vorstellung sein.

Aufgabe 2 (SQL) [32 PUNKTE]

Gegeben sei folgendes Datenbankschema mit beispielhaften Einträgen.



Album

AlbumID	Name	ArtistID
42	Whenever you need ...	13
52	Bohemian Rhapsody	12
...	...	

Song

SongID	Name	Length	AlbumID
42	Never gonna give you up	3:33	42
52	Bohemian Rhapsody	6:00	52
53	Livin on a Prayer	4:09	64
...	...	...	

Artist

ArtistID	Name
12	Queen
13	Rick Astley
14	Bon Jovi

Hinweis: Unterstrichene Attribute sind Primärschlüssel. Unterstrichelte Attribute sind Fremdschlüssel zu gleichnamigen Primärschlüsseln anderer Tabellen.

Lösen Sie folgende Aufgaben in SQL Syntax.

1. Geben Sie alle Songs aus dem Album "Bohemian Rhapsody" aus.
2. Kann man einen Natural-Join für Teilaufgabe 1 verwenden? Erklären Sie kurz.
3. Geben Sie die Namen der Alben aus, die nicht von Rick Astley sind.
4. Gesucht sind alle Künstler (Artists), von denen insgesamt mehr als 30 Minuten Musik in der Datenbank gespeichert sind. Geben Sie die Namen dieser Künstler (Artists) und die durchschnittliche Länge ihrer Songs aus.
   Hinweis: 30 Minuten können Sie mit folgender Syntax darstellen: `TIME '00:30:00'`
5. Erstellen Sie die Tabelle „Song". Prüfen Sie dabei, dass ein Song kürzer als eine Stunde ist. Nutzen Sie sinnvolle Datentypen.
6. Geben Sie eine Tabelle aus, welche für jeden Künstler (Artist) die Anzahl an Alben und die Anzahl an Songs ausgibt. Die Ausgabe soll drei Spalten haben, mit Name des Künstlers (Artist), der Anzahl der Alben und der Anzahl der Songs, geordnet nach der Anzahl der Songs.
7. Löschen Sie alle Songs, die kürzer als 20 Sekunden sind.

Aufgabe 3 (Relationale Algebra) [12 PUNKTE]

Nutzen Sie für folgende Teilaufgaben das Datenbankschema aus Aufgabe 2.

1. Gegeben ist folgende Anfrage:

   ```
   Geben Sie die Namen aller Songs aus, die auf dem Album
   mit dem Namen "Bohemian Rhapsody" sind.
   ```

   Überführen Sie diese in einen Ausdruck der Relationalen Algebra.

2. Geben Sie die in Teilaufgabe 1 gegebene Anfrage im Tupelkalkül an.

3. Formulieren Sie folgende Anfrage in Relationaler Algebra.

   ```
   Geben Sie die Namen aller Künstler (Artists) aus, die keine Alben haben.
   ```

Aufgabe 4 (Optimierung) [12 PUNKTE]

Gegeben seien die beiden Relationen *Schüler* und *Klasse*.

Schüler{int SchuelerID, varchar Name, int zugehörig[Klasse], ...}
Klasse{int KlassenID, int Größe, ...}

mit folgendem Umfang:

Relation Schüler:	6 Attribute, 500 Datensätze
Relation Klasse:	5 Attribute, 25 Datensätze

Gegeben sei folgende SQL-Anweisung:

```
SELECT DISTINCT Name
FROM Schüler, Klasse
WHERE zugehörig=KlassenID AND KlassenID=17;
```

Geben Sie den Operatorbaum vor und nach der Optimierung an. Geben Sie zusätzlich zu jeder Operation die Anzahl der aktuellen Attribute und Datensätze an. Dabei können Sie annehmen, dass die Klasse mit ID 17 aus 20 Schülern besteht.

Aufgabe 5 (Kanonische Überdeckung und Normalform) [30 PUNKTE]

1. Gegeben sei folgendes relationales Schema $\mathcal{R} = \{\text{Name, Mail, Standort, MitarbeiterID}\}$ mit den beiden funktionalen Abhängigkeiten

$$\text{Name, Mail, Standort} \rightarrow \text{MitarbeiterID}$$
$$\text{MitarbeiterID} \rightarrow \text{Name, Mail}$$

   Ist dieses Schema in 2NF, 3NF und/oder Boyce-Codd Normalform? Begründen Sie jeweils.

2. Gegeben sei $\mathcal{R} = (A, B, C, D, E, F)$ mit folgenden funktionalen Abhängigkeiten:

$$\begin{aligned} CD &\rightarrow E \\ E &\rightarrow AC \\ BDE &\rightarrow A \\ CD &\rightarrow AB \\ AE &\rightarrow BD \end{aligned}$$

   Geben Sie die Attributhülle von E an.

3. Geben Sie alle Schlüsselkandidaten an.

4. Geben Sie die kanonische Überdeckung an.

5. Arbeiten Sie nun mit folgendem relationalen Schema $\mathcal{R} = (A, B, C, D, E, F)$ und kanonischer Überdeckung weiter:

$$\begin{aligned} AF &\rightarrow D \\ CE &\rightarrow DF \\ AD &\rightarrow EF \\ BCF &\rightarrow E \\ EF &\rightarrow BC \end{aligned}$$

   Führen Sie den Synthesealgorithmus durch, um das Schema in die 3. Normalform zu überführen. Nennen Sie dabei alle Zwischenschritte.

Teilaufgabe II: Sofwaretechnologie

Aufgabe 1 (UML-Modellierung) [30 PUNKTE]

Zeichnen Sie ein UML-Klassendiagramm zu folgendem Problem:

Ein Eisenbahnnetzbetreiber besitzt mehrere Streckenabschnitte, auf denen andere unabhängige Verkehrsunternehmen Züge fahren lassen wollen. Jeder Streckenabschnitt kann entweder elektrifiziert sein oder nicht. Außerdem ist ihm ein Maximalgewicht zugeordnet, das befahrende Züge haben dürfen. Zusätzlich wird er intern durch eine Start- und eine Endkilometermarkierung (Gleitkommazahl) definiert. Auf einem Streckenabschnitt können sich mehrere Bahnhöfe befinden, ein Bahnhof kann aber nie ohne zugehörigen Streckenabschnitt existieren. Ein Bahnhof hat dabei einen Namen und eine private Kilometermarkierung, die seine Position auf der Strecke angibt.

Die Streckenabschnitte werden von Zügen befahren. Jeder Zug befindet sich immer auf einem Streckenabschnitt, ein Streckenabschnitt muss aber nicht immer durch einen Zug befahren werden. Jeder Zug wird durch eine eindeutige fortlaufende Nummer beginnend mit eins identifiziert. Es wird zwischen Personen- und Güterzügen unterschieden. Personenzüge haben eine Sitzplatzanzahl, Güterzüge eine Beschreibung der Ladung. Nur Personenzüge können an beliebig vielen Bahnhöfen halten. Jeder Zug ist zusammengesetzt aus genau einer Lokomotive, von der er angetrieben wird, und zwischen 0 und 30 anderen Waggons. Jeder Waggon hat ein Gesamtgewicht und eine Anzahl an Achsen, auf die sich dieses Gewicht verteilt. Eine Lokomotive kann als besonderer Waggon angesehen werden, der jedoch einen Antrieb hat. Die Antriebsart kann dabei elektrisch, dieselbetrieben oder wasserstoffbetrieben sein. Lokomotiven mit anderen Antriebsarten befahren die betrachteten Streckenabschnitte nicht. Bei Rangierarbeiten können Waggons und Lokomotiven vom Zug getrennt und zu neuen Zügen zusammengesetzt werden.

Ein Verkehrsunternehmen ist Besitzer von mindestens einem Zug. Um Züge auf den Streckenabschnitten des Netzbetreibers fahren zu lassen, stellt dieser eine Methode zur Reservierung bereit. Diese Methode benötigt als Parameter einen Zug und den Streckenabschnitt, der befahren werden soll. Zurückgegeben wird eine Reservierung, über die dann der Netzbetreiber und das Verkehrsunternehmen in Verbindung stehen. Die Reservierung enthält eine Nutzungsgebühr. Außerdem wird vermerkt, auf welchem Streckenabschnitt und mit welchem Zug die Reservierung gültig ist.

Aufgabe 2 (Anforderungsanalyse) [30 PUNKTE]

Gegeben ist die folgende Beschreibung für ein Fitnessstudio:

> Ein Fitnessstudio bietet unterschiedliche Dienstleistungen für Kunden an. Hierbei wird unterschieden, ob es sich um einen Gast oder ein Mitglied handelt: Ein Kunde muss einen Check-in im Fitnessstudio durchführen. Handelt es sich um einen Gast, ist der Check-in um den Erwerb eines Tagestickets erweitert.
> Der Erwerb eines Tagestickets inkludiert das Ausstellen einer Rechnung durch einen Angestellten. Kunden können Squash spielen. Ausschließlich Mitgliedern steht die Teilnahme an Kursen zur Verfügung, wobei hier die Kurse Functional Training, Rückengymnastik und Yoga angeboten werden. Die Kurse werden von einem Fitnesstrainer durchgeführt. Ein Fitnesstrainer kann Trainingspläne für Mitglieder erstellen und ausschließlich für Gäste Beratungsgespräche anbieten. Des Weiteren übernimmt der Fitnesstrainer auch die regulären Tätigkeiten eines Angestellten, wie z.B. Geburtstagskarten schreiben. Betreuung von Kunden für Outdoor-Sportarten durch den Fitnesstrainer wird explizit nicht angeboten.

(a) Erstellen Sie ein UML Use-Case Diagramm (Anwendungsfalldiagramm), das die im beschriebenen System enthaltenen Use Cases und Akteure in Verbindung setzt.

Hinweise: Verwenden Sie Vererbung, Spezialisierung und Generalisierung, wo sinnvoll. Kennzeichnen Sie außerdem *include*- bzw. *extend*-Beziehungen entsprechend mit Bedingungen, wo notwendig.

(b) Erstellen Sie ein UML Aktivitätsdiagramm für den nachfolgenden Sachverhalt:

> Wir betrachten den zeitlichen Ablauf im Fitnessstudio dieses Mal etwas genauer. Zuerst erfolgt natürlich der Check-in. Schlägt dieser fehl, bleibt dem Kunden nichts anderes übrig, als das Fitnessstudio zu verlassen, d.h. die Aktivität ist hiermit beendet. Andernfalls kann der Kunde zwischen verschiedenen Aktivitäten wählen. Die folgenden Aktivitäten stehen zur Auswahl: Squash spielen, an einem Kurs teilnehmen, Check-out. Bei der Kursteilnahme erklärt zunächst der Fitnesstrainer den Ablauf, während sich der Teilnehmer gleichzeitig aufwärmt; erst wenn beides abgeschlossen ist, wird das Kursprogramm absolviert. Nach abgeschlossener Aktivität kann der Kunde eine neue Aktivität wählen, bis schlussendlich der Check-out erfolgt und somit die Aktivität ebenfalls endet.

Hinweise:

- Beschriften Sie ausgehende Kanten bei Verzweigungen sinnvoll.
- Ihr Modell sollte keine redundanten Aktionen bzw. Zustände beinhalten.
- Modellieren Sie nur das, was wirklich laut Sachverhalt gefordert ist.

Aufgabe 3 (Entwurfsmuster)

[30 PUNKTE]

Gegeben ist die folgende Beschreibung für eine Firma:

> Eine Firma besteht aus verschiedenen Entitäten, die ihre Mitarbeitenden beschreibt. Es gibt Entwickler, Manager sowie Abteilungen. Jede Entität bietet die Möglichkeit, Details über sich anzuzeigen. Für Abteilungen ist es ferner notwendig, dass es die Möglichkeit gibt, Mitarbeitende hinzuzufügen und auch wieder zu entfernen.
> Um es für die Firma einfacher zu machen, besteht die Möglichkeit, alle beschriebenen Entitäten zu besuchen. Konkret soll es die Möglichkeit geben, Prämien zu verteilen.

(a) Modellieren Sie den gegebenen Sachverhalt in einem UML-Klassendiagramm unter Verwendung der Entwurfsmuster *Composite* (*Kompositum*) und *Visitor* (*Besucher*).

(b) Welches Prinzip der Methodenauswahl beim Aufruf liegt dem *Visitor* Pattern zugrunde? Erklären Sie dieses Prinzip kurz.

(c) Entwurfsmuster werden in die Kategorien Erzeugungsmuster, Strukturmuster und Verhaltensmuster eingeteilt. Nennen Sie jeweils die Kategorie, denen die Pattern *Composite* und *Visitor* zugeordnet werden.

(d) Nennen Sie für jede der drei Kategorien zwei weitere, noch nicht genannte Entwurfsmuster.

(e) Nennen und beschreiben Sie die Technik, die es in Programmiersprachen wie Java erlaubt, mehrere Methoden mit dem gleichen Namen, aber unterschiedlichen Parametertypen anzubieten.

Erklären Sie zudem, was diese Technik vom Überschreiben unterscheidet.

Aufgabe 4 (Testing)

[30 PUNKTE]

(a) Erklären Sie den Unterschied zwischen Black-Box-Testing und White-Box-Testing und nennen Sie für jedes einen Vorteil gegenüber dem anderen.

(b) Gegeben sei die Funktion `isOddAndPositive`, welche als Eingabe ein Array an Integer-Zahlen nimmt und ermittelt, ob alle darin enthaltenen Zahlen positiv und ungerade sind:

```
1  boolean isOddAndPositive(int[] numbers) {
2      boolean containsNegative = false;
3      boolean containsEven = false;
4      int pos = 0;
5      while (pos < numbers.length) {
6          if (numbers[pos] < 0) {
7              containsNegative = true;
8          } else if (numbers[pos] % 2 == 0) {
9              containsEven = true;
10         }
11         pos++;
12     }
13     if (containsNegative || containsEven) {
14         return false;
15     }
16     return true;
17 }
```

Zeichnen Sie den Kontrollflussgraphen für die Funktion `isOddAndPositive`. Verwenden Sie Zeilennummern, um die Knoten des Graphen zu beschriften. Verwenden Sie einen expliziten Start- und End-Knoten, und sonst nur Knoten für Anweisungen.

(c) Bestimmen Sie die zyklomatische Komplexität (cyclomatic complexity) der Methode `isOddAndPositive`. Geben Sie hierzu die Formel sowie auch Ihre Berechnungen mit an.

(d) Gegeben sei folgender Test: `[]` (d.h. leeres Array).

- Geben Sie den Ausführungspfad (als Sequenz von Knoten im Kontrollflussgraphen) an.
- Ermitteln Sie die Anweisungsüberdeckung des Tests.
- Geben Sie dazu die allgemeine Formel zur Berechnung der Anweisungsüberdeckung an, entnehmen Sie die konkreten Werte dieser Aufrufe aus dem Kontrollflussgraphen und berechnen Sie schließlich die erreichte Überdeckung. Brüche brauchen dabei nicht gekürzt werden. Beziehen Sie Start- und End-Knoten dabei in die Berechnung ein.

(e) Geben Sie eine *minimale* Menge von Programmeingaben an, die zu 100 % Zweigüberdeckung führt.

Thema Nr. 2
(Aufgabengruppe)

Es sind alle Aufgaben dieser Aufgabengruppe zu bearbeiten!

Teilaufgabe I: Datenbanksysteme

Aufgabe 1 (Vermischte Fragen) [12 PUNKTE]

Beantworten Sie die folgenden Teilaufgaben in jeweils zwei bis drei Sätzen.

(a) Erläutern Sie, was man unter einer Integritätsbedingung (*constraint*) im Kontext eines Datenbanksystems versteht.

(b) Erklären Sie, was der Unterschied zwischen der `WHERE`- und der `HAVING`-Klausel in SQL ist.

(c) Nennen Sie die Auswirkung, die die Verwendung von `FETCH FIRST n ROWS ONLY` bei einer `SELECT`-Anfrage hat und wozu diese gut ist.

(d) Erklären Sie, warum die Projektion in der relationalen Algebra nicht immer stets das gleiche Ergebnis wie eine entsprechende `SELECT`-Anfrage in SQL liefert und wie man diesen Unterschied vermeiden kann.

(e) Erläutern Sie, was eine Sicht (*view*) ist und wofür man diese benutzt.

(f) Erläutern Sie, was man unter Atomarität bei Transaktionen versteht und wie man diese sicherstellen kann.

Aufgabe 2 (ER-Modellierung) [32 PUNKTE]

Im Folgenden finden Sie die textuelle Beschreibung eines Modells. Erstellen Sie zu dieser Beschreibung ein erweitertes ER-Diagramm. Kennzeichnen Sie die Primärschlüssel durch Unterstreichen und geben Sie die Kardinalitäten in Chen-Notation (= Funktionalitäten) an. Kennzeichnen Sie auch die totale Teilnahme von Entity-Typen an Beziehungstypen.

Es gibt Wertpapiere. Diese haben eine eindeutige, internationale Wertpapierkennnummer (ISIN), einen Namen, einen Emittenten sowie ein Land, das jedoch aus der ISIN abgeleitet werden kann. Jedes Wertpapier ist entweder eine Aktie, eine Anleihe oder ein Fonds. Aktien haben Termine, zu denen die Hauptversammlungen stattfinden. Anleihen haben zusätzlich einen Zinssatz. Fonds wiederum investieren in mindestens ein anderes Wertpapier, das eine Aktie, eine Anleihe oder erneut ein Fonds sein kann. Wertpapiere können an Börsen gehandelt werden, in diesem Fall werden pro Börse und Wertpapier für jeden Handelstag die Schlusskurse (Geldkurs und Briefkurs) gespeichert. Börsen haben einen Namen und einen eindeutigen Marktidentifikationscode (MIC). Es gibt Anleger, die beliebig viele verschiedene Wertpapiere besitzen können, wobei zu jeder Position vermerkt wird, wie viele Stücke eines Wertpapiers jemand besitzt. Es kann keine Wertpapiere geben, die niemandem gehören. Anleger können Personen oder Firmen sein. Personen haben einen Namen und eine eindeutige Steuernummer, Firmen einen eindeutigen Namen und eine Rechtsform.

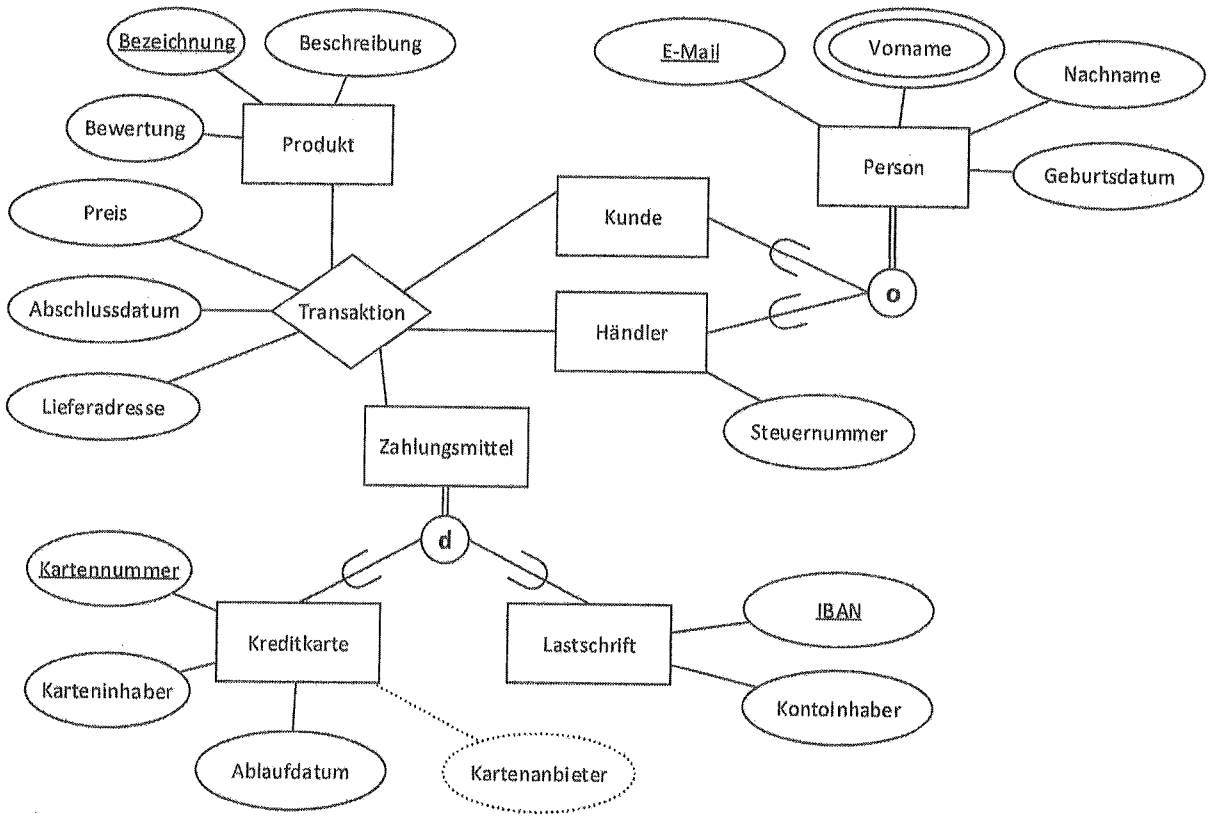
Aufgabe 3 (Relationaler Entwurf) [21 PUNKTE]

Entwerfen Sie zum untenstehenden ER-Diagramm ein Relationenschema in dritter Normalform, (3NF). Achten Sie darauf, dass Ihr Schemaentwurf zu möglichst wenigen NULL-Werten in den resultierenden Tabellen führt.

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. Wenn ein Attribut zur korrekten Abbildung des ER-Diagramms als `UNIQUE` oder `NOT NULL` ausgezeichnet werden muss, geben Sie dies an.

Beispiel:

```
Relation1 (Primärschlüssel, Attribut1, Attribut2,
 Fremdschlüsselattribut1[Relation1],
 (Fremdschlüssel2_Attribut1, Fremdschlüssel2_Attribut2)[Relation2]);
Attribut1 UNIQUE
Attribut2 NOT NULL
```



Aufgabe 4 (SQL) [33 PUNKTE]

Gegeben sind folgende Relationen:

```
Abgeordneter(Platznummer, Name, Geburtsdatum,
             Bundesland, Wahlkreis, Diäten, Mitgliedschaft[Fraktion])
Fraktion(Bezeichnung, Vorsitzender[Abgeordneter])
Abstimmung(Gesetz, Nummer[Abgeordneter], Stimme)
```

In allen Attributen sind keine NULL-Werte erlaubt.

Verwenden Sie im Folgenden nur Standard-SQL und keine produktspezifischen Erweiterungen. Sie dürfen bei Bedarf Views anlegen. Geben Sie einen Datensatz im Ergebnis nicht mehrfach aus.

(a) Schreiben Sie eine SQL-Anweisung, die die Tabelle *Abgeordneter* anlegt. Gehen Sie davon aus, dass die Tabelle *Fraktion* bereits existiert.

(b) Schreiben Sie eine SQL-Anweisung, welche die Namen aller Abgeordneten der Fraktion *Bratwurst für alle* aus dem Wahlkreis Nürnberg sowie des Fraktionsvorsitzenden ausgibt, absteigend sortiert nach der Platznummer.

(c) Schreiben Sie eine SQL-Anweisung, welche die Namen aller Abgeordneten ausgibt, die aus dem gleichen Bundesland kommen wie ihr Fraktionsvorsitzender. Geben Sie die ältesten Abgeordneten zuerst aus.

(d) Schreiben Sie eine SQL-Anweisung, welche die Diäten von allen Fraktionsvorsitzenden um 20 % erhöht.

(e) Schreiben Sie eine SQL-Anweisung, welche die Platznummern und Namen der fünf Abgeordneten ausgibt, welche am häufigsten nicht wie ihr Fraktionsvorsitzender abgestimmt haben. Standardkonforme Sprachkonstrukte, die eine Beschränkung der Ausgabe bewirken, sind erlaubt.

(f) Schreiben Sie eine SQL-Anweisung, die eine Sicht erzeugt, aus der hevorgeht, wie hoch für jedes Gesetz die Anzahl an Ja-Stimmen aus den jeweiligen Fraktionen ist. Die Namen und das Stimmverhalten der Abgeordneten sollen aber geheim bleiben.

Aufgabe 5 (Relationale Algebra) [7 PUNKTE]

Formulieren Sie basierend auf den in Aufgabe 4 gegebenen Relationen die geforderten Anfragen **in der Relationenalgebra.**

(a) Formulieren Sie eine Anfrage, welche die Gesetze ausgibt, für welche Angela Merkel mit Ja gestimmt hat.

(b) Formulieren Sie eine Anfrage, welche die Platznummern aller Abgeordneten ausgibt, die nicht Vorsitzender einer Fraktion sind.

Aufgabe 6 (Normalformen) [8 PUNKTE]

Gegeben ist die Relation

`Rechnung(KundenNr, Name, Geburtsdatum, RechnungsNr, Betrag)`

mit den Schlüsselkandidaten `(KundenNr, RechnungsNr)` und `(RechnungsNr, Name, Geburtsdatum)`. Es gelten außerdem die funktionalen Abhängigkeiten

$$\texttt{KundenNr} \rightarrow \texttt{(Name, Geburtsdatum)}$$
$$\texttt{(Name, Geburtsdatum)} \rightarrow \texttt{KundenNr}$$

Alle Attributwerte sind atomar.

Geben Sie die höchste Normalform an, die die Relation `Rechnung` erfüllt. Zeigen Sie, dass alle Bedingungen für diese Normalform erfüllt sind und dass mindestens eine Bedingung der nächsthöheren Normalform verletzt ist. Beziehen Sie sich bei der Begründung auf die gegebene Relation und nennen Sie nicht nur die allgemeinen Definitionen der Normalformen.

Aufgabe 7 (Optimierung) [7 PUNKTE]

(a) Erläutern Sie den Begriff des Sekundärindexes anhand eines Beispiels.

(b) Übertragen Sie folgendes SQL-Statement in einen *nicht optimierten* Ausdruck der relationalen Algebra.

```
SELECT Mitarbeiter.Name, Mitarbeiter.Geburtsdatum
FROM Mitarbeiter, Filiale
WHERE Mitarbeiter.Filiale = Filiale.ID
AND Filiale.Ort = 'Erlangen';
```

(c) Nennen Sie zwei Möglichkeiten, den Ausdruck aus der vorhergehenden Teilaufgabe logisch zu optimieren. Beziehen Sie sich auf konkrete Stellen und Operatoren des von Ihnen aufgestellten Ausdrucks.

Teilaufgabe II: Softwaretechnologie

Aufgabe 1 (UML-Klassendiagramm) [30 PUNKTE]

Erstellen Sie im Rahmen dieser Aufgabe ein UML-Klassendiagramm, welches ein vereinfachtes Kontoverwaltungssystem modelliert und folgende Eigenschaften erfüllen muss. Achten Sie darauf, **geeignete Sichtbarkeiten** zu verwenden, generell sollen die Attribute nach außen nicht sichtbar sein. Achten Sie bei Assoziationen auf **passende Multiplizitäten** sowie darauf, ob es sich um eine **Aggregation oder eine Komposition** handeln könnte. Verwenden Sie **passende Datentypen**.

Hinweis: Sie müssen keine Konstruktoren sowie keine getter- und setter-Methoden angeben.

Ein Aufzählungstyp definiert zwei Typen von Konten: `Girokonto` und `Sparkonto`. Ein `Konto` speichert Informationen über den Kontostand, den Typ des Kontos, sowie ob es gesperrt ist. Außerdem besitzt jedes Konto eine Kontonummer, die anhand des aktuellen Werts des statischen Attributs zaehler (Startwert 1) berechnet wird. Zusätzlich implementiert das Konto zwei Schnittstellen. Die `MitarbeiterSchnittstelle` erlaubt es, den Kontostand anzusehen, das Konto zu sperren und zu entsperren. Die `KundenSchnittstelle` erlaubt ebenfalls den Kontostand anzusehen, außerdem kann man über sie Geld einzahlen und auszahlen.

Eine `Adresse` speichert Informationen über die Postleitzahl, die Ortschaft, die Straße sowie die Hausnummer. Die abstrakte Klasse `Person` speichert Informationen über den Namen sowie die Adresse. Diese Angaben sollen in den Unterklassen `Angestellter` und `Kunde` verwendbar sein. Ein `Angestellter` verwaltet beliebig viele Konten über die `MitarbeiterSchnittstelle`. Ein `Kunde` kann beliebig viele Konten besitzen und greift auf diese über die `KundenSchnittstelle` zu.

Eine `Bank` besitzt beliebig viele Konten, wobei ein Konto genau einer Bank zugeordnet ist und außerhalb dieser Bank nicht existieren kann. Man kann ein Konto mit Hilfe der Kontonummer von einer Bank `abrufen`. Eine Bank beschäftigt beliebig viele Angestellte, die allerdings bei mehreren Banken tätig sein können. Analog gilt für Kunden: eine Bank kann mehrere Kunden haben, die Kunden können die Dienste mehrerer Banken in Anspruch nehmen.

Aufgabe 2 (Qualitätssicherung) [30 PUNKTE]

Betrachten Sie folgende **Java-Methode**, die aus einer übergebenen Zeichenkette eine neue erzeugt. Dabei werden alle Großbuchstaben durch die entsprechenden Kleinbuchstaben ersetzt und umgekehrt. Zeichen, die keine Groß- oder Kleinbuchstaben innerhalb des Zeichensatzes zwischen A und Z bzw. zwischen a und z darstellen, werden dabei ignoriert und durch Leerzeichen ersetzt.

```
1  public static String encode(String s) {
2      String result = null;
3
4      if (s != null) {
5          result = "";
6
7          int index = 0;
8          while (index < s.length()) {
9              char c = s.charAt(index);
10
11             if (c >= 'A' && c <= 'Z') {
12                 result = result + (char) (c + 32);
13             } else if (c >= 'a' && c <= 'z') {
14                 result = result + (char) (c - 32);
15             } else {
16                 result = result + " ";
17             }
18
19             ++index;
20         }
21     }
22
23     return result;
24 }
```

(a) Geben Sie für die Methode einen **Kontrollflussgraphen** an, wobei Sie die Knoten mit den jeweiligen Zeilennummern aus dem Quelltext beschriften.

(b) Geben Sie eine **minimale Testmenge** an, die das Kriterium der **Kantenüberdeckung** erfüllt.

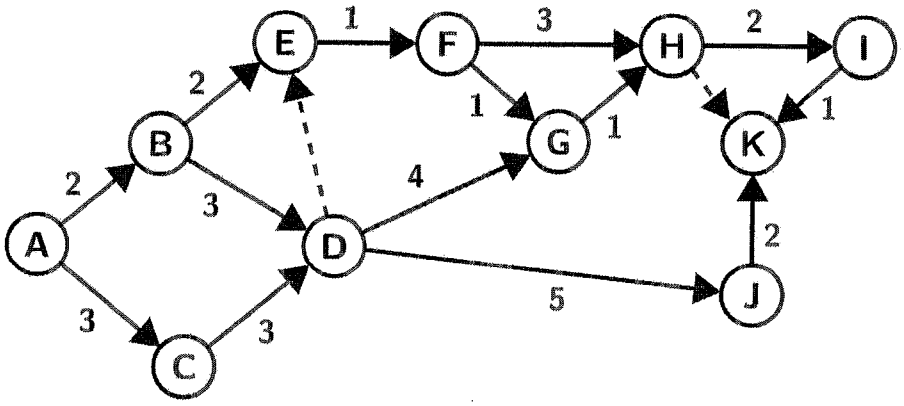
(c) Geben Sie eine **minimale Testmenge** an, die das Kriterium der **Boundary-Interior-Pfadüberdeckung** erfüllt.

(d) Beschreiben Sie den Zusammenhang zwischen **Knotenüberdeckung** und **Kantenüberdeckung** in Bezug auf das hier beschriebene Beispiel.

Hinweise: Das Kriterium *Boundary-Interior-Pfadüberdeckung* beschreibt einen Spezialfall der Pfadüberdeckung, wobei nur Pfade berücksichtigt werden, bei denen jede Schleife nicht mehr als zweimal durchlaufen wird. Eine *Testmenge* ist *minimal*, wenn es keine Testmenge mit einer kleineren Zahl von Testfällen gibt. Die Minimalität muss nicht bewiesen werden.

Aufgabe 3 (Projektmanagement) [30 PUNKTE]

Im Folgenden ist ein **CPM-Netzwerk** dargestellt, das aus elf Ereignissen besteht.

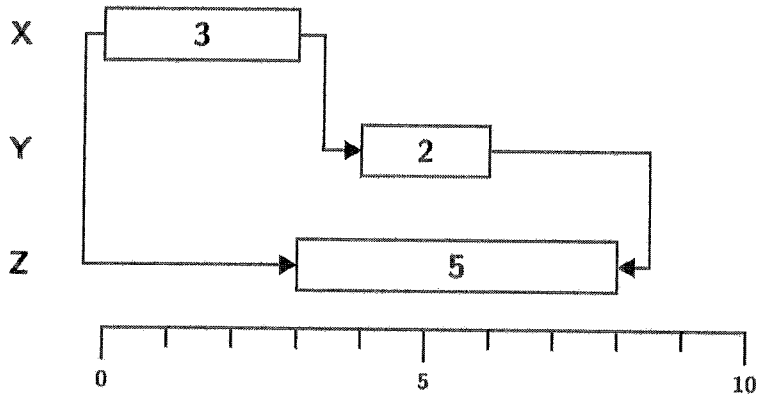


(a) Berechnen Sie die **früheste Zeit** für jedes Ereignis, wobei angenommen wird, dass das Projekt zum Zeitpunkt 0 startet.

(b) Setzen Sie anschließend beim letzten Ereignis K die späteste Zeit gleich der frühesten Zeit und berechnen Sie die **spätesten Zeiten** aller anderen Ereignisse.

(c) Berechnen Sie nun für jedes Ereignis die **Pufferzeit** und geben Sie damit den **kritischen Pfad** an.

Betrachten Sie nun folgendes **Gantt-Diagramm**, das drei Aktivitäten umfasst und mit einer Zeitachse versehen ist. Die Namen der Aktivitäten sind links dargestellt.



(d) Nennen Sie die im Diagramm abgebildeten **Arten von Abhängigkeiten**. Geben Sie weitere Möglichkeiten von Abhängigkeiten in Gantt-Diagrammen an.

(e) Konvertieren Sie das gegebene Gantt-Diagramm in ein **semantisch äquivalentes CPM-Netzwerk**. Dabei sollen Projektstart und -ende als explizite Ereignisse modelliert werden.

(f) Beschreiben Sie den konzeptionellen **Hauptunterschied** zwischen *Aktivitäten* in Gantt-Diagrammen und *Ereignissen* in CPM-Netzwerken. Was bedeutet das für die Konvertierung von Gantt-Diagrammen wie in Teilaufgabe (e)?

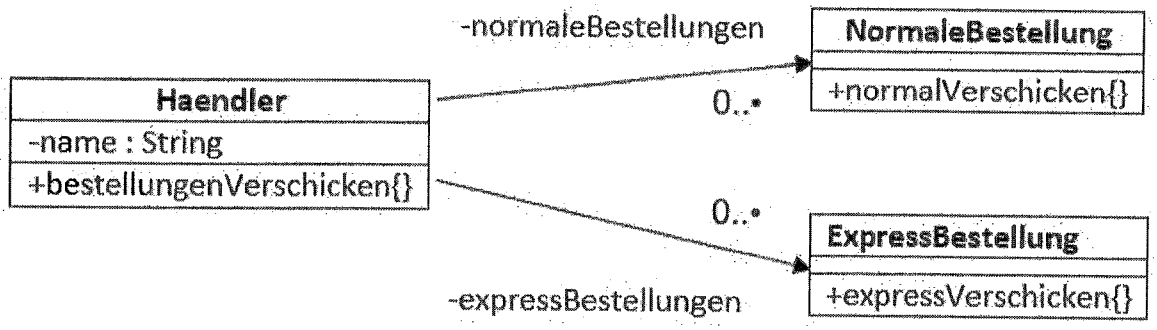
Aufgabe 4 (Entwurfsmuster) [30 PUNKTE]

(a) Diese Teilaufgabe behandelt die Erzeugung von Tickets innerhalb des Szenarios *Bahn-Navigator*. Geben Sie den vollständigen **Quelltext** in einer objektorientierten Programmiersprache Ihrer Wahl an, wobei das Entwurfsmuster **Factory** für die Erzeugung der Tickets genutzt wird. Um sicherzustellen, dass es genau eine Instanz der Fabrik gibt, kommt das Entwurfsmuster **Singleton** zum Einsatz; dabei wird die Instanz erst bei Gebrauch (nicht vor dem ersten Zugriff) erzeugt. **Orientieren** Sie sich an folgender **Beschreibung**:

Jedes Ticket ist mit dem Namen des Kunden (Zeichenkette), der Streckenkennung (Zeichenkette) sowie dem Preis (Ganzzahl) versehen. Zusätzlich ist auf jedem Ticket vermerkt, ob dieses einen flexiblen Reiseverlauf erlaubt (also keine Zugbindung besteht) und ob dieses storniert werden kann. Es werden drei Arten von Tickets unterschieden: Zunächst sind *flexible Tickets* stornierbar und es besteht keine Zugbindung; als Preis wird der Einfachheit halber *100* verwendet. Des Weiteren gibt es *Spar-Tickets*, die stornierbar sind, allerdings eine Zugbindung mit einem Preis von *70* haben. Mit *Super-Tickets* nimmt man Zugbindung und fehlende Stornierungsmöglichkeit für einen Preis von *40* in Kauf.

Die Fabrik soll zur Erzeugung der Tickets **genau eine Methode** zur Verfügung stellen; die Methode erwartet (neben weiteren benötigten Parametern) die Art des Tickets in Form eines **Aufzählungsliterals**. Geben Sie außerdem eine *main()* - Methode an, in der Sie ein *flexibles Ticket* erzeugen.

(b) Folgendes Klassendiagramm ist gegeben. Wenden Sie das **Strategy-Muster** an und erstellen Sie das passende Klassendiagramm mit dem angewendeten Muster.



(c) Geben Sie eine Implementierung in einer objektorientierten Programmiersprache Ihrer Wahl der Klassen `Haendler` und `NormaleBestellung` **unter Berücksichtigung** des Strategy-Musters an. Sie müssen keine expliziten Importe angeben.

- Achten Sie auf die Zugriffsberechtigungen. Der Name des Händlers wird **bei seiner Erzeugung festgelegt**, es soll aber möglich sein, über einen **lesenden Zugriff** diesen Namen abzufragen.
- Nach seiner Erzeugung verfügt der Händler vorerst über keine Bestellungen. Es soll möglich sein, dem Händler neue Bestellungen mitzuteilen.
- Bei der Methode `bestellungenVerschicken()` sollen alle Bestellungen des Händlers verschickt werden. Daraufhin sollte der Händler über keine eingetragenen Bestellungen mehr verfügen.
- Bei der Methode zum Verschicken einer normalen Bestellung soll lediglich eine Meldung **Bestellung verschickt!** auf der Standardausgabe ausgegeben werden.

Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer

Kennzahl: _____

Kennwort: _____

Arbeitsplatz-Nr.: _____

Herbst 2022

66118

Erste Staatsprüfung für ein Lehramt an öffentlichen Schulen
— Prüfungsaufgaben —

Fach: **Informatik (vertieft studiert)**

Einzelprüfung: **Fachdidaktik**

Anzahl der gestellten Themen (Aufgaben): **3**

Anzahl der Druckseiten dieser Vorlage: **7**

Bitte wenden!

Thema Nr. 1

Stichworte: Codierung, Verschlüsselung, Unterrichtsplanung, Hefteinträge, Wahlkurs

Der LehrplanPLUS sieht für das Fach spät beginnende Informatik in der Jahrgangsstufe 11 folgenden Lernbereich vor:

Inf11 Lernbereich 2: Codierung und Verschlüsselung (ca. 11 Std.)

Kompetenzerwartungen

Die Schülerinnen und Schüler ...

- erläutern das Prinzip und die Bedeutung der Codierung von Information anhand geeigneter Beispiele aus der Informationstechnologie (z. B. Strichcode, RGB-Farbcode, QR-Code) sowie die Möglichkeit der Validierung codierter Information durch die Verwendung von Prüfsummen, z. B. GTIN, IBAN.
- codieren natürliche Zahlen binär und hexadezimal und führen entsprechende Decodierungen durch.
- erklären die Grundprinzipien symmetrischer und asymmetrischer Verschlüsselung.
- wenden einfache symmetrische Verfahren zur Ver- und Entschlüsselung an (z. B. Caesar, Vigenère) und implementieren einen zugehörigen Algorithmus.
- bewerten Sicherheitsaspekte eines ausgewählten Verfahrens im Hinblick auf mögliche Angriffe, insbesondere durch die Brute-Force-Methode.
- erklären den Zweck und die prinzipielle Funktionsweise einer digitalen Signatur und beschreiben in diesem Zusammenhang die Bedeutung von Zertifikaten.

Inhalte zu den Kompetenzen:

- binäre und hexadezimale Codierung von Zahlen (Binärsystem, Hexadezimalsystem), Bit, Byte
- symmetrische und asymmetrische Verschlüsselung: Klartext, Geheimtext, Schlüssel
- Brute-Force-Verfahren
- Authentifizierung, Integrität, digitale Signatur, kryptografische Hashfunktion, Zertifikat

Aufgabe 1: Begründung von Unterrichtsinhalten

Einige der Inhalte aus dem obigen Lehrplanausschnitt wären auch in einem Wahlkurs, den Schülerinnen und Schüler der Jahrgangsstufe 7 besuchen, unterrichtbar. Nennen Sie diese und begründen Sie die Relevanz der Inhalte für die Schülerinnen und Schüler der Jahrgangsstufe 7!

Aufgabe 2: Differenzierung

a) Erstellen Sie **jeweils** eine Feinplanung für eine Einzelstunde zur binären Codierung natürlicher Zahlen für den Wahlkurs der **Jahrgangsstufe 7** und für die **Jahrgangsstufe 11**! Legen Sie dazu zunächst jeweils drei beobachtbare Feinziele fest! Achten Sie auf die nötige Differenzierung und darauf, dass die Feinziele Ihre jeweiligen Einzelstunden abdecken! Schildern Sie anschließend den Unterrichtsfortgang nachvollziehbar in Stichpunkten!

b) Verfassen Sie zu den Begriffen symmetrische und asymmetrische Verschlüsselung jeweils einen altersgerechten Hefteintrag für den Wahlkurs der Jahrgangsstufe 7 bzw. für die Jahrgangsstufe 11!

c) Begründen Sie aus fachdidaktischer Sicht Ihre in a) und b) vorgenommenen Differenzierungen!

Thema Nr. 2

Stichworte: Algorithmus, Alltagsbezug, Auswahl von Werkzeugen, Unterrichtsentwurf, Strukturelement „Wiederholung"

Im LehrplanPLUS der Jahrgangsstufe 7 des bayerischen Gymnasiums wird im Schwerpunkt Informatik des Faches Natur und Technik die Beschreibung von Abläufen durch Algorithmen thematisiert. Dabei werden folgende Kompetenzen angeführt:

Kompetenzerwartungen

Die Schülerinnen und Schüler ...

- analysieren und strukturieren geeignete Problemstellungen u. a. aus ihrer Erfahrungswelt (z. B. Bedienung eines Geräts), entwickeln Algorithmen zu deren Lösung und beschreiben diese unter effizienter Verwendung von Kontrollstrukturen.
- setzen unter sinnvoller Nutzung algorithmischer Bausteine einfache Algorithmen mithilfe geeigneter Programmierwerkzeuge um.

Inhalte zu den Kompetenzen:

- Algorithmus: Definition des Begriffs, Strukturelemente (Anweisung, Sequenz, ein- und zweiseitig bedingte Anweisung, Wiederholung mit fester Anzahl, Wiederholung mit Bedingung)
- Fachbegriffe: Algorithmus, Anweisung, Sequenz, ein- und zweiseitig bedingte Anweisung, Wiederholung mit fester Anzahl, Wiederholung mit Bedingung

Aufgabe 1

Sie möchten das Thema „Beschreibung von Abläufen durch Algorithmen" einführen.

a) Geben Sie zwei Phänomene (maximal ein Beispiel zur Bedienung eines Geräts) aus der Lebenswelt der Schülerinnen und Schüler an, die sich für die Einführung des Begriffs „Algorithmus" eignen und erläutern Sie diese kurz!

b) Diskutieren Sie die Bedeutung von Alltagsbezug für den Informatikunterricht! (ca. 1 Seite)

Aufgabe 2

a) Geben Sie fünf geeignete Kriterien zur Auswahl von Werkzeugen für die Einführung in die Programmierung an und erklären Sie diese Kriterien!

b) Nennen Sie zwei konkrete Werkzeuge zur Einführung in die Programmierung! Diskutieren Sie die Vor- und Nachteile und entscheiden Sie sich auf dieser Basis für ein Werkzeug!

Aufgabe 3

Im Folgenden erstellen Sie einen Unterrichtsentwurf für eine Doppelstunde zum Thema „Wiederholung"! Gehen Sie hierfür davon aus, dass die Strukturelemente Anweisung und Sequenz bereits zuvor behandelt wurden!

a) Geben Sie ein Grobziel und drei beobachtbare Feinziele der Unterrichtsstunde an! Achten Sie darauf, dass die Feinziele die Doppelstunde abdecken!

b) Erstellen Sie einen Unterrichtsentwurf für die Doppelstunde unter Berücksichtigung Ihrer in a) definierten Lernziele! Beschreiben Sie dazu in textueller Form den geplanten Ablauf und skizzieren Sie Tafelbilder, Hefteinträge und Arbeitsblätter grob! Gliedern Sie Ihren Entwurf nach Unterrichtsphasen und markieren Sie die Stellen, an denen Sie davon ausgehen, dass die Lernziele erreicht wurden!

c) Nennen Sie Ihre drei wichtigsten fachdidaktischen Entscheidungen und begründen Sie diese!

Aufgabe 4

Auf einer Website zum Thema „Programmieren lernen" finden Sie zur Einführung der „Wiederholung" folgenden Text:

Es gibt vier Arten von Schleifen: die „for-Schleife", die „while-Schleife", die „do-while-Schleife" und die „for-each-Schleife".

Nehmen Sie aus fachdidaktischer Sicht zu dieser Aussage Stellung!

Thema Nr. 3

Stichworte: Objektorientierte Modellierung und Programmierung, Programmiersprache, Konstruktor, Unterrichtsplanung, Fehler

Der LehrplanPLUS legt für das Fach Informatik in der Jahrgangsstufe 9 unter anderem folgende Kompetenzen und Inhalte fest:

Inf9 Lernbereich 3: Grundlagen der objektorientierten Modellierung und Programmierung (ca. 26 Std.)

Kompetenzerwartungen

Die Schülerinnen und Schüler ...

- analysieren Objekte aus ihrer Erfahrungswelt (z. B. Fahrzeuge, Personen) hinsichtlich ihrer Eigenschaften (Attribute) und Fähigkeiten (Methoden) und abstrahieren sie zu Klassen. Sie stellen Objekte und Klassen als Grundlage einer möglichen Implementierung grafisch dar.
- deklarieren eine Klasse sowie die zugehörigen Attribute und Methoden in einer objektorientierten Programmiersprache.
- verwenden bei der Implementierung Wertzuweisungen, um Attributwerte zu ändern, und interpretieren diese als Zustandsänderung des zugehörigen Objekts.
- formulieren unter Verwendung der Kontrollstrukturen Algorithmen zu geeigneten Problemstellungen, u. a. durch grafische Darstellungen.
- implementieren Methoden auf der Grundlage gegebener Algorithmen objektorientiert, wobei sie sich des Unterschiedes zwischen Methodendefinition und Methodenaufruf bewusst sind. Dabei nutzen sie ggf. auch Methoden anderer Klassen.
- analysieren, interpretieren und modifizieren Algorithmen, wodurch sie die Fähigkeit erlangen, fremde Programme flexibel einzusetzen und kritisch zu bewerten.
- modellieren durch Klassendiagramme einfache Generalisierungshierarchien zu geeigneten Strukturen aus ihrer Erfahrungswelt.
- implementieren mithilfe einer objektorientierten Sprache einfache Generalisierungshierarchien; dabei nutzen sie das Konzept der Vererbung sowie die Möglichkeit, Methoden zu überschreiben.

Inhalte zu den Kompetenzen:

- objektorientierte Konzepte, u. a. Objekt, Klasse, Attribut, Attributwert, Methode
- Variablenkonzept; Arten von Variablen: Parameter, lokale Variable und Attribute; Übergabewert
- Wertzuweisung zur Änderung von Variablenwerten
- Methoden: Methodenkopf, Methodenrumpf, Methodendefinition, Methodenaufruf, Übergabewert, Rückgabewert; Konstruktor als spezielle Methode; Standardmethoden zum Geben und Setzen von Attributwerten
- Algorithmus: Strukturelemente, grafische Darstellung, Pseudocode
- Datentypen: ganze Zahlen, Gleitkommazahlen, Wahrheitswerte, Zeichen, Zeichenketten
- Generalisierung und Spezialisierung: Ober- und Unterklasse, Vererbung von Attributen und Methoden an Unterklassen, Überschreiben von Methoden
- Fachbegriffe: Parameter, Übergabewert, Rückgabewert, lokale Variable, Wertzuweisung, Konstruktor, Methodenkopf, Methodenrumpf, Vererbung, Generalisierung, Spezialisierung, Oberklasse, Unterklasse

Aufgabe 1: Wahl einer geeigneten Entwicklungsumgebung

Für die Umsetzung des Lernbereichs 3 gibt es unterschiedliche Möglichkeiten, die Lerninhalte anzuordnen. Ein Einflussfaktor kann auch die Wahl der Entwicklungsumgebung sein.
Betrachten Sie im Folgenden die zwei Möglichkeiten, eine Entwicklungsumgebung zu verwenden,

- die sowohl das imperative als auch das objektorientierte Paradigma gleichberechtigt ermöglicht, bzw.
- die von Beginn an auf das objektorientierte Paradigma abzielt!

Erläutern Sie auf etwa eineinhalb Seiten, welche Auswirkungen die beiden Alternativen für die Grobplanung der Unterrichtssequenz haben! Treffen Sie eine begründete Wahl für eine der beiden Alternativen, indem Sie Vor- und Nachteile abwägen!

Aufgabe 2: Konstruktor als „spezielle Methode"

Bei den Lehrplaninhalten zur Methode ist der „Konstruktor als spezielle Methode" erwähnt.

a) Erläutern Sie die Rolle des Konstruktors für eine Klasse altersgerecht! Gehen Sie dabei auch darauf ein, warum er als „spezielle Methode" bezeichnet werden kann!

b) Erklären Sie anhand zweier Beispiele, inwiefern der Vergleich zwischen Konstruktor und Methode problematisch werden kann! Ziehen Sie zur Illustration eine konkrete, für die Jahrgangsstufe 9 geeignete Programmiersprache heran!

Im Folgenden planen Sie eine Doppelstunde zum Thema „Einführung des Konstruktors"! Sie können dabei auf Ihre Ergebnisse aus den Teilaufgaben a) und b) zurückgreifen.

c) Schildern Sie die Lernvoraussetzungen für Ihre Doppelstunde!

d) Geben Sie drei beobachtbare Feinziele für Ihre Doppelstunde zur Einführung des Konstruktors an! Achten Sie darauf, dass die Feinziele die Doppelstunde abdecken!

e) Beschreiben Sie den Unterrichtsverlauf der Doppelstunde nachvollziehbar in textueller Form! Gliedern Sie Ihren Entwurf nach Unterrichtsphasen und markieren Sie die Stellen, an denen Sie davon ausgehen, dass die Feinziele aus d) erreicht wurden!

f) Nennen Sie Ihre drei wichtigsten fachdidaktischen Entscheidungen und begründen Sie diese!

Aufgabe 3: Fehler

Sie verwenden im Unterricht eine Klasse „Veranstaltungsticket" zur Einübung der Konzepte von Methoden, Konstruktoren und Abläufen in Methoden. Die Schülerinnen und Schüler arbeiten dabei eigenständig an einer Reihe von Aufgaben. Im Folgenden finden Sie zwei Aufgaben und je eine Lösung der Schülerinnen und Schüler (in Java).
Beschreiben Sie, welche Fehler in den Lösungen vorhanden sind und wie sich diese auf den Ablauf des Codes auswirken! Formulieren Sie individuelle Hinweise, die Sie den Schülerinnen und Schülern zur Behebung der Fehler geben können!

Aufgabe: Schreibe einen neuen Konstruktor, der die Parameterwerte entgegennimmt und die Werte der Attribute `preis` und `veranstaltungsname` passend setzt.

```
public Veranstaltungsticket Veranstaltungsticket(double preis, String
veranstaltungsname){
    preis = preis;
    veranstaltungsname = veranstaltungsname;
}
```

Aufgabe: Implementiere eine Methode, die die Mehrwertsteuer berechnet und zurückgibt. Beim Aufruf der Methode soll der Mehrwertsteuersatz übergeben werden.

```
double mehrwertSteuerGeben(int mwstSatz, double mehrwertsteuer){
    mehrwertsteuer = preis * mwstSatz / (100 + mwstSatz);
}
```

Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer

Kennzahl: _____

Kennwort: _____

Arbeitsplatz-Nr.: _____

Herbst 2022

46115

Erste Staatsprüfung für ein Lehramt an öffentlichen Schulen

— Prüfungsaufgaben —

Fach: **Informatik (Unterrichtsfach)**

Einzelprüfung: **Theoretische Informatik/Algorithmen/Datenstrukturen**

Anzahl der gestellten Themen (Aufgaben): 2

Anzahl der Druckseiten dieser Vorlage: 15

Bitte wenden!

Thema Nr. 1
(Aufgabengruppe)

Es sind alle Aufgaben dieser Aufgabengruppe zu bearbeiten!

Teilaufgabe I: Algorithmen

Aufgabe 1 (Algorithmenanalyse) [24 PUNKTE]

Gegeben sei ein Feld A von ganzen Zahlen. Das Feld habe n Elemente A[1] bis A[n]. Die Problemgröße ist der Wert n. Gezählt werden sollen Aufrufe der Funktion print im schlechtesten Fall.

(a) Bestimmen Sie die Komplexitätsklasse in der O-Notation für die folgenden Algorithmen. Begründen Sie jeweils Ihre Antwort.

i.
```
1  function magic (int A[]) {
2    for (x = 1; x <= n; x++) {
3      for (y = 1; y <= 10; y++) {
4        for (z = 1; z <= n; z++) {
5          if (A[x] == A[z]) {
6            print "Magic";
7          }
8        }
9      }
10   }
11 }
```

ii.
```
1  function more_magic (int A[], int n) {
2    for (x = 1; x <= n; x++) {
3      for (y = 1; y <= n; y++) {
4        for (z = 1; z <= n; z++) {
5          if (A[x] == A[z]) {
6            print "More Magic";
7          }
8        }
9      }
10   }
11   a = n;
12   for (j = 1; j < a; j++) {
13     if (A[j] == A[a - j]) {
14       print "More Magic";
15     }
16   }
17 }
```

iii.
```
1  function even_more_magic (int A[], int n) {
2    for (x = 0; x <= 100000000; x++) {
3      y = n * n;
4      for (z = 0; z <= y; z = z+z) {
5        if (A[x] == A[z]) {
6          print "Even More Magic";
7        }
8      }
9    }
10 }
```

(b) Bestimmen Sie die Komplexitätsklasse in der O-Notation für den folgenden rekursiven Algorithmus. Stellen Sie zunächst eine Rekursionsgleichung auf. Begründen Sie Ihre Antwort.

```
1  function recursive_magic (int A[], int n) {
2    if (n > 1) {
3      for (x = 0; x < 27; x++) {
4        recursive_magic(A, n/3);
5      }
6      for (y = 0; y < n*n; y++) {
7        for (z = 0; z <= n; z = z+z) {
8          if (A[z] > 0) {
9            print "Recursive Magic";
10         }
11       }
12     }
13   }
14 }
```

Aufgabe 2 (Sortierverfahren) [26 PUNKTE]

In der folgenden Aufgabe soll ein Feld E von ganzen Zahlen aufsteigend sortiert werden. Das Feld habe n Elemente E[0] bis E[n−1]. Der folgende Algorithmus sei gegeben:

```
1   void sort (int E[], int n) {
2     int i, j, m;
3     for (i = 0; i < n ; i++) {
4       m = i;
5       for (j = i + 1; j < n ; j++) {
6         if (E[j] <= E[m]) {
7           m = j;
8         }
9       }
10      if (true) {
11        swap(E, i, m);
12      }
13    }
14  }
```

Die Funktion swap(E,x,y) vertauscht im Feld E die Elemente an den Stellen x und y.

(a) Sortieren Sie das folgende Feld E mittels des Algorithmus. Geben Sie die Belegung des Feldes nach jedem Durchlauf der äußeren Schleife in einer neuen Zeile an.

Index	0	1	2	3	4	5	6	7	8	9
Wert	1	3	2	7	0	4	8	5	7	6

(b) Geben Sie die *genaue* Anzahl der in Zeile 6 durchgeführten *Vergleichsoperationen* zwischen Feldelementen als Funktion f der Eingabegröße n an. Begründen Sie Ihre Antwort. (Beachten Sie, dass hier die *genaue* Anzahl gefragt ist und *nicht* die asymptotische Komplexität.)

(c) Wie muss das Feld beschaffen sein, damit möglichst selten die Zuweisung in Zeile 7 ausgeführt wird? Begründen Sie Ihre Antwort.

(d) Welche Auswirkung auf das Ergebnis des Algorithmus hätte es, wenn die Bedingung in Zeile 6 wie folgt lauten würde?

if (E[j] > E[m])

Begründen Sie Ihre Antwort.

(e) Geben Sie die *genaue* Anzahl der in Zeile 11 durchgeführten Aufrufe der Funktion swap als Funktion g der Eingabegröße n an. Begründen Sie Ihre Antwort. (Beachten Sie, dass hier die *genaue* Anzahl gefragt ist und *nicht* die asymptotische Komplexität.)

(f) Der Algorithmus führt auch dann Vertauschungen durch, wenn diese für die Herstellung der Sortierung nicht notwendig sind. Geben Sie ein Feld mit möglichst wenig Feldelementen als Beispiel an, bei dem dieser Fall eintritt. Begründen Sie Ihre Antwort.

(g) Wie muss die Bedingung in der if-Anweisung in Zeile 10 angepasst werden, damit nur dann Einträge im Feld getauscht werden, wenn es wirklich nötig ist? Begründen Sie Ihre Antwort.

Aufgabe 3 (Binäre Suche) [18 PUNKTE]

Gegeben sei ein aufsteigend sortiertes Array A der Länge n, in dem Werte auch mehrfach vorkommen können. Ziel ist es, die Anzahl der Vorkommen eines beliebigen Wertes k in A mit einer logarithmischen Laufzeit von $O(\log n)$ zu bestimmen.

(a) Ergänzen Sie die zwei Bedingungen in der *rekursiven* Methode first: Sie verwendet binäre Suche, um die Position des ersten Vorkommens von k in A innerhalb des geschlossenen Intervalls [low; high] zu bestimmen und gibt -1 zurück, falls k nicht in A vorkommt.

```
1   int first(int A[], int k, int low, int high) {
2     if (high < low) {
3       return -1;
4     }
5     int mid = low + (high - low) / 2;
6     if ⟨erste Bedingung⟩ {
7       return mid;
8     }
9     if ⟨zweite Bedingung⟩ {
10      return first(A, k, mid+1, high);
11    } else {
12      return first(A, k, low, mid-1);
13    }
14  }
```

(b) Angenommen, Sie haben die folgenden zwei Hilfsmethoden first und last zur Verfügung, die binäre Suche verwenden, um die Position des ersten bzw. letzten Vorkommens von k in A innerhalb des geschlossenen Intervalls [low; high] zu bestimmen. Falls k nicht in A vorkommt, geben die Methoden -1 zurück. Die Signaturen dieser Methoden lauten jeweils:

`int first(int A[], int k, int low, int high)`

`int last(int A[], int k, int low, int high)`

Ergänzen Sie die Methode count: Sie verwendet die beiden Hilfsmethoden, um die Anzahl der Vorkommen von k in A zu bestimmen.

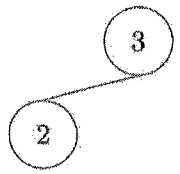
```
1 int count(long A[], long k) {
2   int n = A.length; // berechnet Laenge von A
3   ⟨hier Ihre Loesung⟩
4 }
```

Aufgabe 4 (Bäume und Graphen) [22 PUNKTE]

(a) Fügen Sie für jede Teilaufgabe den angegebenen Wert in den dazu angegebenen AVL-Baum mit aufsteigender Sortierung ein. Führen Sie danach bei Bedarf die erforderliche(n) Rotation(en) aus und zeichnen Sie das Ergebnis.

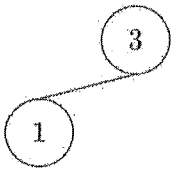
i.)

1 *einfügen:*



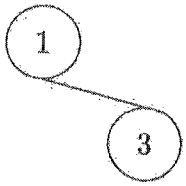
ii.)

2 *einfügen:*



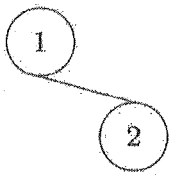
iii.)

2 *einfügen:*

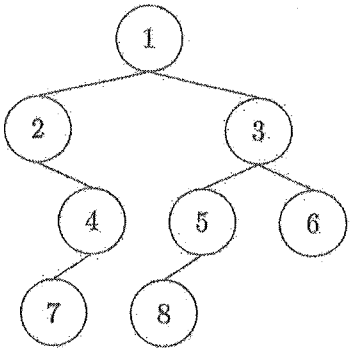


iv.)

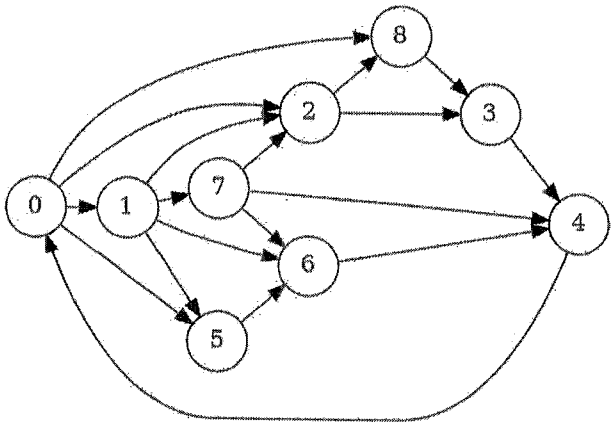
3 *einfügen:*



(b) Traversieren Sie den folgenden Baum beginnend bei der Wurzel 1 mittels Tiefensuche und geben Sie die besuchten Knoten jeweils in der Reihenfolge *preorder*, *inorder* und *postorder* an. Falls mehrere Nachfolger möglich sind, besuchen Sie diese in aufsteigender Reihenfolge.



(c) Traversieren Sie den folgenden gerichteten Graphen beginnend bei Knoten 0 mittels Tiefensuche und mittels Breitensuche und geben Sie jeweils die verarbeiteten Knoten in der Besuchsreihenfolge (preorder) aus. Falls mehrere Nachfolger möglich sind, besuchen Sie diese in aufsteigender Reihenfolge.



Teilaufgabe II: Theoretische Informatik

Aufgabe 1 (Reguläre und kontextfreie Sprachen) [52 PUNKTE]

Bergpanoramen in ASCII-Darstellung sind beliebige Wörter über dem Alphabet $\{/, \backslash, \_\}$. Die drei Symbole meinen dabei: Mit jedem /-Zeichen erhöht sich die Höhe um 1, mit jedem \-Zeichen erniedrigt sich die Höhe um 1 und _-Zeichen lassen die Höhe unverändert. Z.B. verändert das Panorama $P_1 :=$ _//_\/_ insgesamt die Höhe um 2, es kann mit Höhenänderung gezeichnet werden als:

```
  _ _
 / \/
_/
```

(a) Beweisen Sie: Die Sprache aller Bergpanoramen ist regulär.

(b) Beweisen Sie: Die Sprache aller Bergpanoramen ist kontextfrei.

(c) Geben Sie einen deterministischen endlichen Automaten M an, der genau jene Bergpanoramen akzeptiert, die mit _/ beginnen und mit _\ enden, d.h. die akzeptierte Sprache des Automaten ist

$$L(M) := \{\_/u\_\backslash \mid u \in \{/, \backslash, \_\}^*\}$$

(d) Sei $L = \{/u\backslash \mid u \in \{/, \backslash, \_\}^+\}$. Geben Sie einen regulären Ausdruck α an, der genau das Komplement von L erzeugt, d.h. die von α erzeugte Sprache ist

$$L(\alpha) := \{u \in \{/, \backslash, \_\}^* \mid u \notin L\}.$$

Erläutern Sie, warum Ihre Lösung genau $L(\alpha)$ erzeugt.

(e) Ein Bergpanorama ist *höheninvariant*, wenn es auf derselben Höhe endet, auf der es beginnt (O.B.d.A. kann angenommen werden, dass es auf Höhe 0 beginnt und dann auf Höhe 0 enden muss.). Das Panorama P_1 ist nicht höheninvariant, aber z. B. ist das Panorama $P_2 :=$//_\/__\\ ein höheninvariantes Bergpanorama, es lässt sich mit Höhenänderung zeichnen als:

```
  _ __
 / \/  \
/      \
```

Ebenso ist $P_3 :=$__/___/ ein höheninvariantes Bergpanorama, mit Höhenänderung gezeichnet als:

```
  _
__/ \
    \__/
```

Die Sprache aller höheninvarianten Bergpanoramen kann definiert werden durch

$$L_{hi} := \{w \mid w \in \{/, \backslash, \_\}^*, \#_/(w) = \#_\backslash(w)\},$$

wobei $\#_s(w)$ die Anzahl an Vorkommen des Symbols s im Wort w bezeichnet.

Beweisen Sie, dass L_{hi} nicht regulär ist.

(f) Beweisen Sie: Die Sprache L_{hi} ist kontextfrei, indem Sie eine kontextfreie Grammatik G angeben, welche genau alle höheninvarianten Bergpanoramen erzeugt (d. h. $L(G) = L_{hi}$). Erläutern Sie, warum Ihre Grammatik L_{hi} erzeugt.

(g) Ein Bergpanorama heißt *ausgewogen*, wenn die Anzahlen der Symbole /, \ und _ im Panorama identisch sind (z. B. ist Panorama P_2 ausgewogen, während P_1 und P_3 nicht ausgewogen sind).

Die Sprache aller ausgewogenen Bergpanoramen kann geschrieben werden als

$$L_{ag} := \{w \mid w \in \{/, \backslash, \_\}^*, \#_/(w) = \#_\backslash(w) = \#_\_(w)\},$$

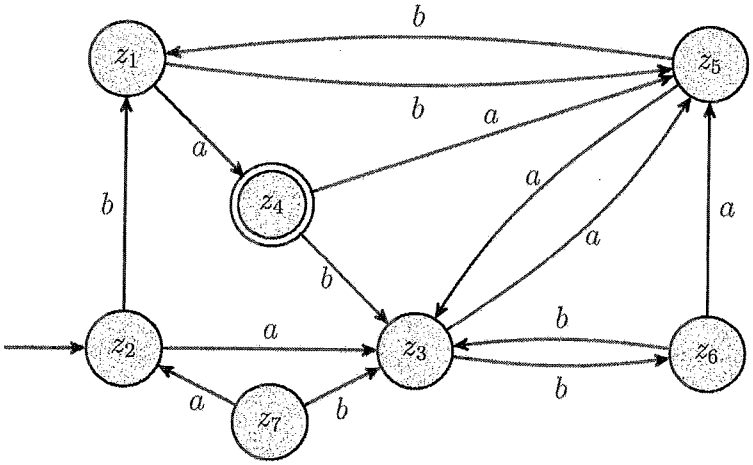
wobei $\#_s(w)$ die Anzahl an Vorkommen des Symbols s im Wort w bezeichnet.

Verwenden Sie das Pumpinglemma für kontextfreie Sprachen, um zu widerlegen, dass L_{ag} kontextfrei ist.

Aufgabe 2 (Minimierung von deterministischen endlichen Automaten) [18 PUNKTE]

Berechnen Sie für den folgenden deterministischen endlichen Automaten (der Worte über dem Alphabet $\Sigma = \{a, b\}$ verarbeitet) den **Minimalautomaten**, d. h. einen deterministischen endlichen Automaten, der die gleiche Sprache akzeptiert und eine minimale Anzahl an Zuständen benutzt.

Erläutern Sie Ihre Berechnung, indem Sie z. B. eine Minimierungstabelle angeben, und geben Sie den Minimalautomaten als Zustandsgraph an.



Aufgabe 3 (Berechenbarkeit) [20 PUNKTE]

Mit $\langle M \rangle$ bezeichnen wir die Gödelnummer der Turingmaschine M.

Prüfen Sie für jede der folgenden formalen Sprachen, ob diese entscheidbar oder unentscheidbar sind und beweisen Sie Ihre Antwort.

1. $L_1 = \{\langle M \rangle \mid$ Turingmaschine M macht bei Eingabe 100 mindestens 100 Schritte und M hält bei Eingabe 101 nach höchstens 200 Schritten$\}$
2. $L_2 = \{\langle M \rangle \mid$ Turingmaschine M hält bei Eingaben 41 und 42 und M hält weder bei Eingabe 43 noch bei Eingabe 44$\}$
3. $L_3 = \{\langle M \rangle \mid$ Turingmaschine M entscheidet das Halteproblem$\}$

Thema Nr. 2
(Aufgabengruppe)

Es sind alle Aufgaben dieser Aufgabengruppe zu bearbeiten!

Teilaufgabe I: Algorithmen

Aufgabe 1 (Suchen) [25 PUNKTE]

(a) Gegeben sei das Feld $a[1], \ldots, a[15]$, welches die folgenden Zahlen in aufsteigend sortierter Reihenfolge enthält:

a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]	a[11]	a[12]	a[13]	a[14]	a[15]
3	9	11	12	15	22	23	40	49	51	53	54	58	61	62

Erläutern Sie, wie die **binäre Suche** nach der Zahl 58 auf dem Feld a schrittweise vorgeht. Dokumentieren Sie insbesondere, welche Vergleiche stattfinden und welches Intervall des Feldes a noch betrachtet werden muss.

(b) Gegeben sei ein Feld $b[1], \ldots, b[m]$ von m Zahlen und eine Zahl n (mit $m \geq n$).

Geben Sie einen Algorithmus in Pseudocode an, der prüft, ob das Feld b **alle Quadratzahlen** von 1 bis n^2 enthält (d.h. die Zahlen $1^2, 2^2, 3^2, \ldots, n^2$).

Sie dürfen annehmen, dass Multiplikationen und Wurzelberechnungen in konstanter Laufzeit für jede Zahl im Feld b durchgeführt werden können.

Geben Sie einen möglichst laufzeit- und platzeffizienten Algorithmus an, der obiges Problem löst. Versuchen Sie im Worst-Case mit einer Laufzeit von $O(m)$ und einem Platzbedarf von $O(n)$ zusätzlichem Platz auszukommen.

Erläutern Sie die Funktionsweise Ihres Algorithmus und analysieren Sie die Laufzeit und den Platzbedarf Ihres Algorithmus für den Worst-Case.

Aufgabe 2 (Komplexität, O-Notation) [25 PUNKTE]

(a) Beweisen Sie die folgenden Aussagen:

i. Sei $f(n) = n^2$. Dann gilt **nicht** $f \in O(n)$.

ii. Sei $f(n) = 20 \cdot n^3 - 3 \cdot n + 40$. Dann gilt $f \in O(n^3)$.

iii. Sei $f : \mathbb{N} \to \mathbb{N}$ definiert durch die folgende Rekursionsgleichung

$$f(n) = \begin{cases} 1, & \text{für } n \leq 1 \\ 2 \cdot f(n-1) - 1, & \text{für } n > 1 \end{cases}$$

Dann gilt $f \in O(2^n)$.

(b) Ordnen Sie die folgenden Funktionen in der Reihenfolge ihres asymptotischen Wachstums, so dass $f_i \in O(f_{i+1})$ gilt.

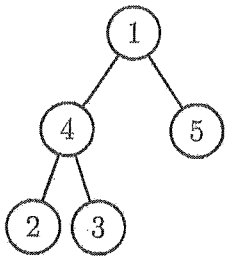
$$2^{(3^n)}, \quad n\log_2(\log_2(5n)), \quad 6n\log_2(3n), \quad 3n\log\sqrt{9n}, \quad 5^n$$

Aufgabe 3 (Binärbäume) [40 PUNKTE]

Jeder Knoten n eines Binärbaums mit Ganzzahlen als Schlüssel habe die Attribute

- $n.key \in \mathbb{Z}$ für den Schlüsselwert des Knotens,
- $n.left$ für das linke Kind von n, wobei $n.left = null$, wenn das linke Kind nicht existiert,
- $n.right$ für das rechte Kind von n, wobei $n.right = null$, wenn das rechte Kind nicht existiert.

Ein Beispielbaum ist:



(a) Der Konstruktor *create* erzeugt Binärbäume: Durch Aufrufe der Form $create(k, l, r)$ wird ein Baum mit einer neuen Wurzel n erzeugt, sodass $n.key = k$, $n.left = l$ und $n.right = r$.

Geben Sie ein Programm in Pseudocode an, welches Aufrufe des Konstruktors *create* verwendet, um den oben gezeigten Beispielbaum zu erzeugen.

(b) Die Schlüssel an den Knoten eines Binärbaums können in verschiedenen Reihenfolgen ausgegeben werden. Neben der Preorder-, Postorder- und Inorder-Reihenfolge ist auch die Levelorder-Reihenfolge möglich. Bei dieser werden alle Knoten der Tiefe 0, dann der Tiefe 1, dann der Tiefe 2 usw. ausgegeben.

Geben Sie je einen **vollständigen** binären Baum an, für den die Ausgabe der Schlüssel jeweils 5, 15, 10, 25, 20, 30, 35 ist, bei Durchlaufen der Knoten in

i. Preorder-Reihenfolge
ii. Postorder-Reihenfolge
iii. Inorder-Reihenfolge
iv. Levelorder-Reihenfolge

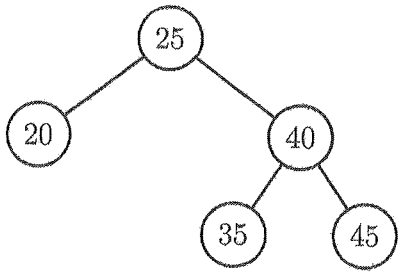
(c) Geben Sie vier Algorithmen in Pseudocode an, die jeweils den Wurzelknoten n eines Binärbaums als Eingabeparameter erhalten und alle Schlüssel des Binärbaums ausgeben in

i. Preorder-Reihenfolge
ii. Postorder-Reihenfolge
iii. Inorder-Reihenfolge
iv. Levelorder-Reihenfolge

Erläutern Sie jeweils die Funktionsweise Ihrer Algorithmen.

(d) Ein binärer Baum ist ein (unbalancierter) **Suchbaum**, wenn für jeden Knoten n gilt: Alle Schlüssel im linken Teilbaum unter n sind kleiner als $n.key$ und alle Schlüssel im rechten Teilbaum unter n sind größer als $n.key$.

Fügen Sie die Schlüssel 15, 5, 10, 40, 30 in dieser Reihenfolge in den Suchbaum



ein und zeichnen sie den entstehenden Baum auf Ihren Bearbeitungsbogen.

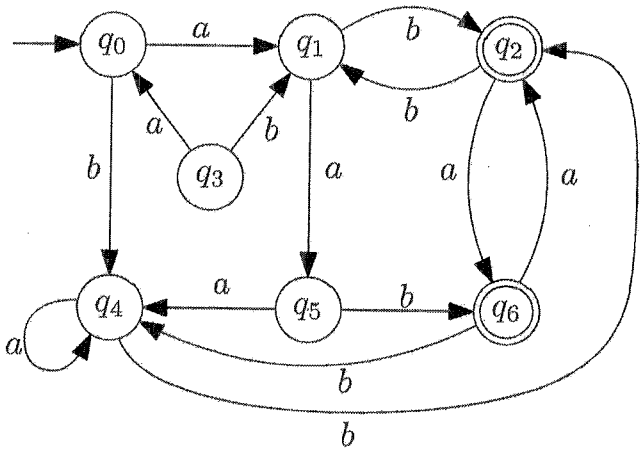
Teilaufgabe II: Theoretische Informatik

Aufgabe 1 (Automaten) [25 PUNKTE]

(a) Eine natürliche Zahl ist bekanntlich genau dann ganzzahlig durch 4 teilbar, wenn die Zahl gebildet aus ihren letzten beiden Ziffern durch 4 teilbar ist.

Sei $\Sigma = \{1, 2, 3, 4\}$ ein Alphabet. Geben Sie einen deterministischen endlichen Automaten an, der ein Eingabewort über dem Alphabet Σ (interpretiert als natürliche Zahl im Dezimalsystem) genau dann akzeptiert, wenn die eingelesene Zahl ganzzahlig durch 4 teilbar ist. Es genügt, den Automaten durch ein Zustandsübergangsdiagramm anzugeben.

(b) Konstruieren Sie für folgenden deterministischen endlichen Automaten, der Worte über dem Alphabet $\{a, b\}$ verarbeitet, den Minimalautomaten, das heißt, einen deterministischen endlichen Automaten, der die gleiche Sprache akzeptiert und eine minimale Anzahl an Zuständen benutzt. Erläutern Sie Ihre Vorgehensweise, indem Sie zum Beispiel eine Minimierungstabelle angeben, und geben Sie das Zustandsdiagramm des Minimalautomaten an.



(c) Zeigen Sie, dass die regulären Sprachen unter Komplement abgeschlossen sind.

Aufgabe 2 (Chomsky-Hierarchie) [20 PUNKTE]

Bestimmen Sie für die folgenden Sprachen über $\Sigma = \{a, b, c\}$ jeweils, ob sie regulär sind und ob sie kontextfrei sind. Beweisen Sie Ihre Antworten. Für „Ja-Antworten" genügt es, eine geeignete Beschreibung (Grammatik/regulärer Ausdruck) oder die Arbeitsweise eines geeigneten Akzeptors (Automat/Maschine) ohne Korrektheitsbeweis anzugeben. Hierbei bezeichnet der %-Operator die Modulo-Operation.

(a) $L_1 = \{a^n b^l c^m \mid l, m, n \in \mathbb{N}_0, l \leq n\}$

(b) $L_2 = \{a^n b^m c^l \mid l, m, n \in \mathbb{N}_0, l \leq n\%2\}$

Aufgabe 3 (Turingmaschinen) [30 PUNKTE]

Gesucht ist eine 1-Band Turingmaschine mit Eingabealphabet $\Sigma = \{0, 1\}$, die bei einer Eingabe ungerader Länge das „mittlere Bit flippt", also eine 0 in der Mitte durch eine 1 ersetzt und andersherum. Bei einer Eingabe gerader Länge sollen beide mittleren Zeichen „geflippt" werden. Beispielsweise wird aus 111 die Ausgabe 101 und aus 101010 wird 100110. Eine leere Eingabe bleibt unverändert.

(a) Beschreiben Sie auf konzeptioneller Ebene, wie Ihre Turingmaschine funktioniert.

(b) Geben Sie eine vollständige Implementierung Ihrer Turingmaschine z.B. als gezeichnetes Zustandsübergangsdiagramm oder TM-Programm an.

(c) Dokumentieren Sie, wie die Zustände aus (b) mit der Beschreibung aus (a) zusammenhängen. Geben Sie hierzu *kurz* die Rolle jedes einzelnen Zustands bzw. seiner Übergänge an.

Aufgabe 4 (Entscheidbarkeit) [15 PUNKTE]

Bestimmen Sie für die folgenden Sprachen, ob sie entscheidbar sind und beweisen Sie jeweils Ihre Antwort. Dabei bezeichnet $\langle M \rangle$ die Gödelnummer der Turingmaschine M.

(a) $L_1 = \{w \in \{0,1\}^* \mid \text{Es gibt eine Turingmaschine } M, \text{ die } w \text{ akzeptiert}\}$

(b) $L_2 = \{\langle M \rangle \mid \text{Es gibt ein Wort } w, \text{ das von } M \text{ akzeptiert wird}\}$

Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer

Kennzahl: _____

Kennwort: _____

Arbeitsplatz-Nr.: _____

Herbst 2022

46116

Erste Staatsprüfung für ein Lehramt an öffentlichen Schulen

— Prüfungsaufgaben —

Fach: **Informatik (Unterrichtsfach)**

Einzelprüfung: **Softwaretechnologie/Datenbanksysteme**

Anzahl der gestellten Themen (Aufgaben): 2

Anzahl der Druckseiten dieser Vorlage: 18

Bitte wenden!

Thema Nr. 1
(Aufgabengruppe)

Es sind alle Aufgaben dieser Aufgabengruppe zu bearbeiten!

Teilaufgabe I: Datenbanksysteme

Aufgabe 1 (ER-Modellierung) [28 PUNKTE]

Entwerfen Sie ein Entity-Relationship Diagramm. Dabei sind, sofern nicht explizit angegeben, alle Attribute nicht eindeutig. Wählen Sie für die Angabe der Kardinalitäten eine sinnvolle Notation aus und führen Sie einen Schlüsselkandidaten ein, wenn nötig.

Es gibt Schwimmbäder mit eindeutigen Namen. In jedem Schwimmbad gibt es mindestens fünf Bahnen, die pro Schwimmbad eine eindeutige Nummer haben. Dazu wird die Länge der Bahn mitgespeichert. Außerdem gibt es pro Schwimmbad genau eine Adresse, welche sich aus PLZ, Stadt und Straße zusammensetzt.

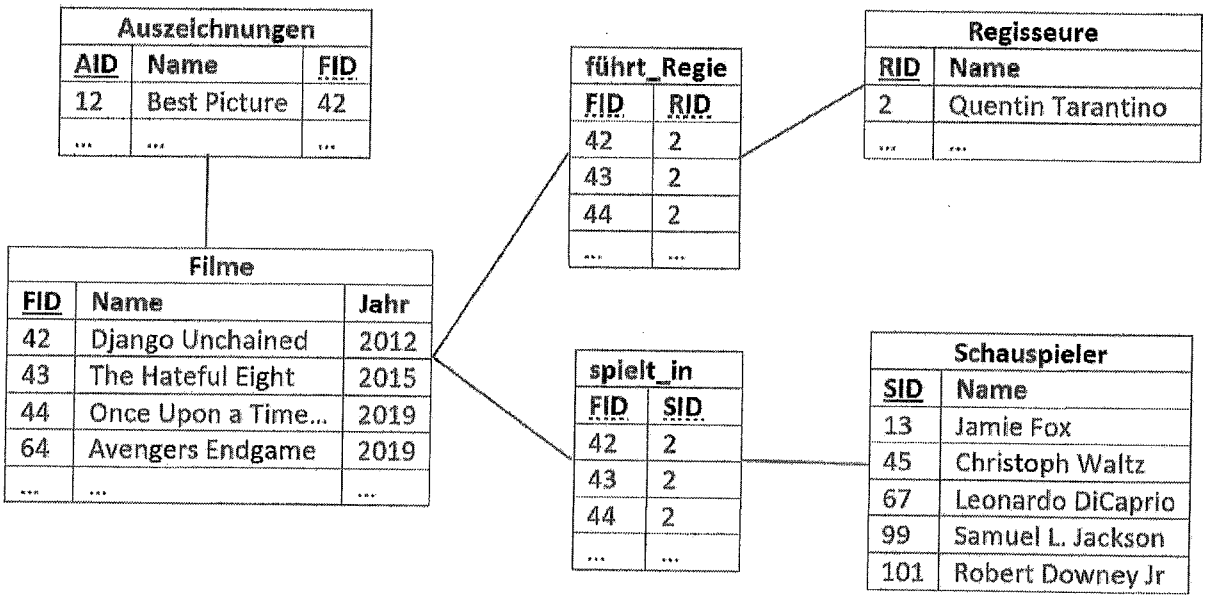
Weiterhin gibt es Kunden, welche mit Namen und E-Mail-Adresse gespeichert werden. Dazu wird jeder Kunde entweder unter der Kategorie Leistungssportler oder Breitensportler eingeordnet. Für Leistungssportler wird zusätzlich das Alter mitgespeichert. Außerdem können Kunden eine Adresse mit den gleichen Eigenschaften wie oben angeben, müssen aber nicht.

Zuletzt können Kunden Reservierungen tätigen. Eine Reservierung ist eindeutig durch den Kunden, das Datum und die Uhrzeit identifizierbar. Weiterhin wird mit der Reservierung mindestens eine Bahn reserviert.

Aufgabe 2 (SQL)

[34 PUNKTE]

Gegeben ist folgendes Datenbankschema mit beispielhaften Einträgen.



Auszeichnungen

AID	Name	FID
12	Best Picture	42
...	...	...

Filme

FID	Name	Jahr
42	Django Unchained	2012
43	The Hateful Eight	2015
44	Once Upon a Time...	2019
64	Avengers Endgame	2019
...	...	...

führt_Regie

FID	RID
42	2
43	2
44	2
...	...

Regisseure

RID	Name
2	Quentin Tarantino
...	...

spielt_in

FID	SID
42	2
43	2
44	2
...	...

Schauspieler

SID	Name
13	Jamie Fox
45	Christoph Waltz
67	Leonardo DiCaprio
99	Samuel L. Jackson
101	Robert Downey Jr

Hinweis: Unterstrichene Attribute sind Primärschlüssel. Gepunktete Linien identifizieren Fremdschlüssel, welche auf gleichnamige Attribute in den anderen Tabellen zeigen.

Formulieren Sie folgende Anfragen in SQL-Syntax.

(a) Geben Sie die Namen aller Filme aus dem Jahr 2012 aus.

(b) Geben Sie die Namen und Jahre aller Filme aus, die unter der Regie von „Quentin Tarantino“ entstanden.

(c) Kann Teilaufgabe (b) nur unter Verwendung von „natural joins“ gelöst werden? Begründen Sie.

(d) Geben Sie pro Schauspieler an, in wie vielen Filmen er mitgespielt hat. Geben Sie die SID, den Namen des jeweiligen Schauspielers und die Anzahl aus.

(e) Geben Sie alle Regisseure aus, die für alle ihre Filme mehr als fünf Auszeichnungen gewonnen haben. Die Ausgabe soll die RID, den Namen des jeweiligen Regisseurs und die Anzahl umfassen.

(f) Erstellen Sie eine Tabelle „Schauspielerauszeichnungen“, um Auszeichnungen der Schauspieler zu verwalten. Darin soll der Name der Auszeichnung, eine SAID und der zugehörige Schauspieler (SID) gespeichert werden.

(g) Geben Sie die Namen aller Filme aus, in denen „Samuel L. Jackson“ NICHT mitgespielt hat.

(h) Geben Sie Paare von Schauspielern aus, die in mindestens zwei Filmen zusammen mitgespielt haben. Die Ausgabe soll die Namen beider Schauspieler beinhalten und die Anzahl an Ko-Auftritten.
Beachten Sie, dass die Ausgabe keine Ergebnisse enthält, bei denen Schauspieler mit sich selbst gespielt haben.

(i) Löschen Sie alle Filme ohne Auszeichnungen.

Aufgabe 3 (Normalformen) [20 PUNKTE]

Gegeben sei $\mathcal{R} = (A, B, C, D, E, F)$ mit folgenden funktionalen Abhängigkeiten.

$$\begin{aligned} BF &\rightarrow ACD \\ B &\rightarrow C \\ ABF &\rightarrow C \\ ABD &\rightarrow EF \\ ABC &\rightarrow D \end{aligned}$$

(a) Geben Sie die Schlüsselkandidaten an. Begründen Sie ihre Antwort kurz.

(b) Geben Sie die kanonische Überdeckung an. Nennen Sie dabei die einzelnen Schritte und führen Sie diese jeweils aus.

(c) Führen Sie nun den Synthesealgorithmus durch. Geben Sie auch hier alle Zwischenschritte an.

Aufgabe 4 (Transaktionen)

[8 PUNKTE]

Gegeben ist folgende Transaktionshistorie H.

Schritt	T_1	T_2	T_3	T_4
1	BOT			
2	w(A)			
3				BOT
4				r(B)
5		BOT		
6		w(A)		
7				w(B)
8	commit			
9				commit
10		w(B)		
11		commit		
12			BOT	
13			r(B)	
14			w(C)	
15			commit	

Beantworten Sie folgende Fragen und begründen Sie Ihre Antwort kurz.

(a) Ist H serialisierbar (SR)?

(b) Ist H rücksetzbar (RC)?

(c) Vermeidet H kaskadierendes Zurücksetzen (ACA)?

(d) Ist H strikt (ST)?

Teilaufgabe II: Softwaretechnologie

Aufgabe 1 (Projektmanagement) [8 PUNKTE]

Gegeben sei folgender Projektplan:

Aktivität	Abhängig von	Dauer in Wochen
A1	–	2
A2	A1	4
A3	A1	3
A4	A2	5
A5	A3	3
A6	A5	2

(a) Zeichnen Sie ein Gantt-Diagramm für den Projektplan.

(b) Bestimmen Sie die Länge des kritischen Pfades und geben Sie an, welche Arbeitspakete an ihm beteiligt sind.

Aufgabe 2 (Agile Softwareentwicklung) [11 PUNKTE]

Beschreiben Sie den Ablauf der Softwareentwicklung mit Scrum.

(a) Nennen und beschreiben Sie kurz die drei Rollen in Scrum.

(b) Nennen und beschreiben Sie kurz die drei Artefakte in Scrum.

(c) Nennen und beschreiben Sie kurz die fünf Ereignisse in Scrum.

Aufgabe 3 (Aktivitätsdiagramm) [14 PUNKTE]

Erstellen Sie ein UML-Aktivitätsdiagramm, das die Benutzerauthentifizierung für ein Online-System beschreibt, deren Ablauf wie folgt aussieht:

(1) Der Benutzer gibt zur Identifizierung seine E-Mail-Adresse an.

(2) Ist der Benutzer im System bekannt, wird das Passwort abgefragt.

(3) Wird das Passwort korrekt eingegeben, so ist der Benutzer angemeldet und wird zur Startseite weitergeleitet.

(4) Wird das Passwort falsch eingegeben, so muss der Benutzer es noch einmal eingeben, solange bis es korrekt ist.

(5) Ist der Benutzer im System noch nicht bekannt, so erfolgt die Registrierung, indem ein neues Passwort abgefragt wird.

(6) Um vor Tippfehlern zu schützen, wird das Passwort danach ein zweites Mal abgefragt.

(7) Wenn das zweimal eingegebene Passwort ungültig ist, so müssen beide Eingaben des Passworts wiederholt werden.

(8) Wenn das zweimal eingegebene Passwort gültig ist, so wird eine E-Mail mit einem Validierungslink an den Benutzer verschickt. Anschließend kann der Benutzer durch Auswahl des Links die Gültigkeit der E-Mail-Adresse bestätigen. Gleichzeitig zum E-Mail-Versand und zur E-Mail-Validierung kann der Benutzer das Profil vervollständigen.

(9) Sobald die E-Mail-Adresse validiert und das Profil vervollständigt ist, ist der Benutzer angemeldet und wird zu einer Einführung in das System geleitet.

(10) Von dort kommt der Benutzer dann weiter zur Startseite.

Aufgabe 4 (UML-Modellierung) [15 PUNKTE]

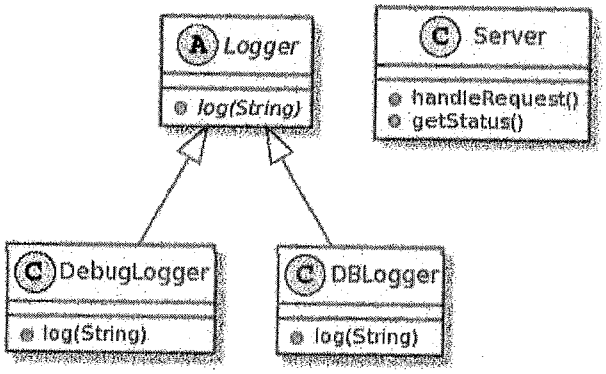
Zeichnen Sie ein UML-Klassendiagramm zu folgendem Problem:

> Ein Eisenbahnnetzbetreiber besitzt mehrere Streckenabschnitte, auf denen andere unabhängige Verkehrsunternehmen Züge fahren lassen wollen. Jeder Streckenabschnitt kann entweder elektrifiziert sein oder nicht. Außerdem ist ihm ein Maximalgewicht zugeordnet, das ihn befahrende Züge haben dürfen. Zusätzlich wird er intern durch eine Start- und eine Endkilometermarkierung (Gleitkommazahl) definiert.
>
> Die Streckenabschnitte werden von Zügen befahren. Jeder Zug befindet sich immer auf einem Streckenabschnitt, ein Streckenabschnitt muss aber nicht immer durch einen Zug befahren werden. Jeder Zug wird durch eine eindeutige fortlaufende Nummer beginnend mit eins identifiziert. Es wird zwischen Personen- und Güterzügen unterschieden. Personenzüge haben eine Sitzplatzanzahl, Güterzüge eine Beschreibung der Ladung. Jeder Zug ist zusammengesetzt aus 0 bis 30 Waggons. Jeder Waggon hat ein Gesamtgewicht. Bei Rangierarbeiten können Waggons vom Zug getrennt und zu neuen Zügen zusammengesetzt werden. Ein Verkehrsunternehmen ist Besitzer von mindestens einem Zug.
>
> Der Netzbetreiber stellt eine Methode bereit, mithilfe derer geprüft werden kann, ob ein Zug einen Streckenabschnitt befahren kann. Diese Methode benötigt als Parameter einen Zug und den Streckenabschnitt, der befahren werden soll. Zurückgegeben wird ein Wahrheitswert, der aussagt, ob der Zug geeignet ist. Der Netzbetreiber nutzt diese Methode, wenn er eigene Züge auf bestimmten Streckenabschnitten fahren lassen möchte.

Aufgabe 5 (Design)

[24 PUNKTE]

Gegeben sind die folgenden Klassen:



- Die Klasse `Server` besitzt eine Methode `handleRequest`, die Benutzereingaben verarbeitet, und eine Methode `getStatus`, mit der der Status des Servers abgefragt werden kann.
- Um Aktionen des Servers zu protokollieren, gibt es eine abstrakte Klasse `Logger`, welche über eine Methode `log` verfügt. Ein `DebugLogger` implementiert diese Methode so, dass eine Ausgabe auf der Konsole passiert; ein `DBLogger` protokolliert Informationen in einer Datenbank.

(a) Wenden Sie das Entwurfsmuster *Observer* (*Beobachter*) an, damit ein `Server` beliebig viele `Logger` über Benutzeranfragen informieren kann, so dass diese zu jeder Anfrage Informationen über den Status loggen können. Übertragen Sie dazu das UML-Klassendiagramm auf Ihren Bearbeitungsbogen und erweitern Sie es entsprechend.

(b) Erstellen Sie ein UML-Sequenzdiagramm, das den folgenden Sachverhalt darstellt: Initial existiert ein `Main` Objekt und ein `Server`. Das `Main` Objekt erstellt einen `DebugLogger` und registriert diesen beim `Server`. Danach stellt der Benutzer eine Anfrage (`handleRequest`), welche mit Hilfe des Observer-Musters den `DebugLogger` informieren soll. Dieser fragt zuerst den aktuellen Status des Servers über dessen `getStatus` Methode ab und protokolliert dann einen Eintrag mit der Methode `log` (die Ausgabe selbst muss nicht mehr modelliert werden).

Aufgabe 6 (Testing)

[18 PUNKTE]

(a) Gegeben sei die Funktion `isEvenAndNegative`, welche als Eingabe ein Array an Integer-Zahlen nimmt, und ermittelt, ob alle darin enthaltenen Zahlen negativ und gerade sind:

```
1  public boolean isEvenAndNegative(int[] numbers) {
2      int pos = 0;
3      while (pos < numbers.length) {
4          int num = numbers[pos];
5          if (num > 0) {
6              return false;
7          } else if (num % 2 == 1) {
8              return false;
9          }
10         pos++;
11     }
12     return true;
13 }
```

Zeichnen Sie den Kontrollflussgraphen für die Funktion `isEvenAndNegative`. Verwenden Sie Zeilennummern, um die Knoten des Graphen zu beschriften. Verwenden Sie einen expliziten Start- und Ende-Knoten und sonst nur Knoten für Anweisungen.

(b) Gegeben sei folgender Test: `[-2, -4]`.

- Geben sie den Ausführungspfad an.
- Ermitteln Sie die Anweisungsüberdeckung des Tests.
- Ermitteln Sie die Zweigüberdeckung des Tests.

Entnehmen Sie die konkreten Werte dieser Aufrufe aus dem Kontrollflussgraphen und berechnen Sie schließlich die erreichte Überdeckung als Anteil überdeckter Elemente. Brüche brauchen dabei nicht gekürzt werden. Beziehen Sie *Start* und *Ende* Knoten dabei in die Berechnung ein.

Thema Nr. 2
(Aufgabengruppe)

Es sind alle Aufgaben dieser Aufgabengruppe zu bearbeiten!

Teilaufgabe I: Datenbanksysteme

Aufgabe 1 (Vermischte Fragen) [16 PUNKTE]

Beantworten Sie die folgenden Teilaufgaben in jeweils ein bis zwei Sätzen.

(a) Erläutern Sie, was man unter Datenunabhängigkeit versteht.

(b) Wozu wird `ON DELETE CASCADE` im Kontext einer `CREATE TABLE` Anweisung gebraucht? Erklären Sie Zweck und Funktionsweise.

(c) Geben Sie an, warum bei der Relationenalgebra die Operation zur Umbenennung unverzichtbar ist. Nennen Sie ein Beispiel für eine Frage, die ohne Umbenennung nicht beantwortet werden kann.

(d) Erläutern Sie, warum bei einer SQL-Anfrage eine Aggregation (wie z.B. `AVG` oder `MAX`) in der `WHERE`-Klausel keinen Sinn macht.

(e) Erläutern Sie anhand eines Beispiels, dass eine Tabelle in 3NF sein kann und trotzdem die BCNF nicht erfüllt.

(f) Gegeben sei eine Tabelle $R(A, B, C)$ mit dem Primärschlüssel AB. Wenn jedes Attribut aus dem Wertebereich $\{0..9\}$ ist, wie viele Tupel kann die Tabelle dann höchstens enthalten? Begründen Sie ihre Antwort.

(g) Was versteht man im Zusammenhang mit Transaktionen unter einem *lost update*? Geben Sie ein Beispiel an.

Aufgabe 2 (ER-Modellierung) [19 PUNKTE]

Im Folgenden finden Sie die textuelle Beschreibung eines Modells. Erstellen Sie zu dieser Beschreibung ein ER-Diagramm. Kennzeichnen Sie die Primärschlüssel durch passendes Unterstreichen und geben Sie die Kardinalitäten in Chen-Notation (= Funktionalitäten) an. Kennzeichnen Sie auch die totale Teilnahme von Entity-Typen an Beziehungstypen.

Das ER-Digramm soll die Abgeordneten in einem Parlament beschreiben. Jeder Abgeordnete hat eine eindeutige Platznummer, einen Namen und ein Geburtsdatum. Ein Abgeordneter kann höchstens zu einer Fraktion gehören. Es kann aber auch fraktionslose Abgeordnete geben. Jede Fraktion hat eine eindeutige Bezeichnung. Es soll außerdem die Anzahl der Abgeordneten jeder Fraktion aus der Datenbank entnehmbar sein. Jede Fraktion hat genau einen Vorsitzenden. Abgeordnete können in mehreren Ausschüssen mitwirken. Es gibt keine Ausschüsse, an denen keine Abgeordneten mitwirken. Innerhalb eines Ausschusses nimmt ein Abgeordneter eine bestimmte Rolle ein (z.B. Vorsitzender oder Schriftführer). Darüber hinaus benennt jede Fraktion für jeden Ausschuss genau einen Fraktionsberichterstatter. Jeder Ausschuss hat eine eindeutige Bezeichnung, ein Anfangsdatum und ein Enddatum.

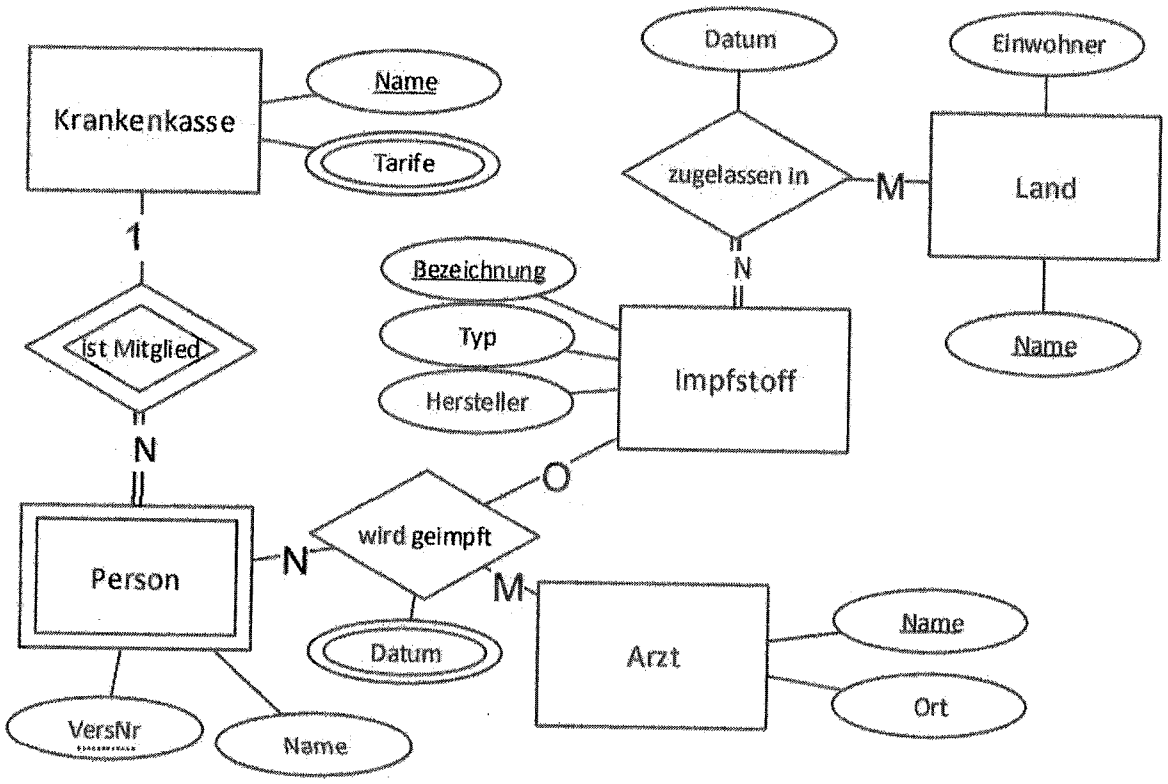
Aufgabe 3 (Relationaler Entwurf) [18 PUNKTE]

Entwerfen Sie zum untenstehenden ER-Diagramm ein Relationenschema in dritter Normalform (3NF) 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. Wenn ein Attribut zur korrekten Abbildung des ER-Diagramms als `UNIQUE` oder `NOT NULL` ausgezeichnet werden muss, geben Sie dies an.

Beispiel:

```
Relation1 (Primärschlüssel, Attribut1, Attribut2,
 Fremdschlüsselattribut1[Relation1],
 (Fremdschlüssel2_Attribut1, Fremdschlüssel2_Attribut2)[Relation2]);
Attribut1 UNIQUE
Attribut2 NOT NULL
```



Aufgabe 4 (SQL)

[27 PUNKTE]

Gegeben ist das folgende Relationendatenbankschema zur Verwaltung einer Autovermietung:

```
Kunde(Führerscheinnummer, Vorname, Nachname, Anschrift, Kreditkartennummer)
Fahrzeug(Kennzeichen, Klasse, Hersteller, Modell, Kraftstoff, Getriebe, Ort[Station])
Vertrag(Führerscheinnummer[Kunde], Kennzeichen[Fahrzeug], Datum, Dauer, Preis)
Station(ID, Straße, Hausnummer, Stadt)
```

In allen Attributen sind keine NULL-Werte erlaubt.

Verwenden Sie im Folgenden nur Standard-SQL und keine produktspezifischen Erweiterungen. Sie dürfen bei Bedarf Views anlegen. Geben Sie einen Datensatz nicht mehrfach aus.

(a) Schreiben Sie eine SQL-Anweisung, die die Tabelle *Fahrzeug* anlegt. Gehen Sie davon aus, dass alle anderen Tabellen bereits existieren.

(b) Schreiben Sie eine SQL-Anweisung, die Vorname und Nachname aller Kunden ausgibt, die eine Kreditkarte von Visa benutzen. Die Nummern solcher Karten beginnen stets mit einer „4". Geben Sie die Kunden absteigend sortiert nach Nachname und bei gleichen Nachnamen aufsteigend nach Vorname aus.

(c) Schreiben Sie eine SQL-Anweisung, die für jeden Kunden neben dem Vor- und Nachnamen den durch den Kunden bisher erzielten Umsatz der Autovermietung ermittelt.

(d) Schreiben Sie eine SQL-Anweisung, welche alle Kunden löscht, die seit dem 01. Januar 2021 kein Auto mehr gemietet haben.

(e) Schreiben Sie eine SQL-Anweisung, welche die zehn Fahrzeuge ausgibt, welche am häufigsten angemietet wurden. Geben Sie zu jedem Fahrzeug die Attribute Kennzeichen, Hersteller und Modell an sowie die Anzahl der Anmietungen. Standardkonforme Sprachkonstrukte, die eine Beschränkung der Ausgabe bewirken, sind erlaubt.

(f) Schreiben Sie eine SQL-Anweisung, welche die Kennzeichen, den Hersteller und das Modell aller Dieselfahrzeuge ausgibt, die zu weniger als 70 % des Durchschnittspreises aller Fahrzeuge in allen Standorten angemietet wurden. Achten Sie darauf, dass jedes Fahrzeug nur einmal ausgegeben wird.

Aufgabe 5 (Normalformen) [10 PUNKTE]

(a) Gegeben ist die Relation $R(A, B, C, D, E)$. Sie besitzt den Schlüsselkandidaten AB und die funktionale Abhängigkeit $C \rightarrow E$.
Begründen Sie in ein bis zwei Sätzen, warum R sich nicht in *dritter Normalform* befindet. Geben Sie eine Zerlegung von R an, sodass die Ergebnisrelationen die zweite Normalform erfüllen. Erzeugen Sie dabei so wenige Relationen wie möglich.

(b) Gegeben ist die Relation $R(A, B, C, D)$. Sie besitzt die Schlüsselkandidaten AB und AC. Weiterhin gilt die funktionale Abhängigkeit $B \rightarrow C$.
Begründen Sie in ein bis zwei Sätzen, warum R sich nicht in der *Boyce-Codd-Normalform* befindet. Geben Sie eine Zerlegung von R an, sodass die Ergebnisrelationen die Boyce-Codd-Normalform erfüllen. Erzeugen Sie dabei so wenige Relationen wie möglich.

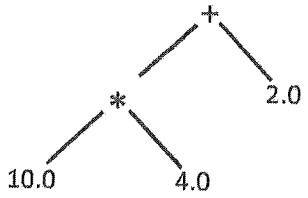
Teilaufgabe II: Softwaretechnologie

Aufgabe 1

[25 PUNKTE]

Gegenstand dieser Aufgabe sind Auswertungsbäume für einfache arithmetische Ausdrücke. Ein Auswertungsbaum besteht aus Operatoren (zweistellige Addition, Subtraktion, Multiplikation und Division) und Operanden (rationale Zahlen). Operatoren lassen sich schachteln, d. h. der Operand eines Operators kann ein Operator sein.

Beispiel:



Die Auswertung dieses Baums liefert den Wert 42.0.

(a) Geben Sie ein UML-Klassendiagramm für Auswertungsbäume an. Verzichten Sie auf Sichtbarkeiten. Geben Sie für Attribute jeweils den Namen und den Typ an. Geben Sie für Assoziationen jeweils die Namen und Multiplizitäten der Assoziationsenden an. Machen Sie ggf. von Komposition und Vererbung Gebrauch. Definieren Sie die Operation `auswerten` in geeigneter Weise.

(b) Geben Sie ein Objektdiagramm für den Auswertungsbaum im obigen Beispiel an.

(c) Nennen Sie das Entwurfsmuster, das in Ihrer Lösung von Teilaufgabe (a) verwendet wird.

(d) Implementieren Sie das Klassendiagramm aus Teilaufgabe (a) in einer objektorientierten Programmiersprache Ihrer Wahl. Verwenden Sie stets öffentliche Sichtbarkeit.

Aufgabe 2

[20 PUNKTE]

Gegeben sei folgende Methode zum Vergleich von Arrays ganzer Zahlen:

```
1  public class Vergleich
2  {
3          public static int gleich(int[] a, int[] b) {
4                  if (a == null || b == null)
5                          return -1; // Fehlercode für undefinierte Argumente
6                  else if (a.length != b.length)
7                          return 0; // Ergebnis falsch
8                  else {
9                          int i = 0;
10                         while (i < a.length) {
11                                 if (a[i] != b[i])
12                                         return 0; // Ergebnis falsch
13                                 i = i + 1;
14                         }
15                         return 1; // Ergebnis wahr
16                 }
17         }
18 }
```

(a) Konstruieren Sie einen Kontrollflussgraphen für die Methode `gleich`. Identifizieren Sie dabei Prädikate und Anweisungen jeweils durch Zeilennummern.

(b) Erläutern Sie die Begriffe Anweisungsüberdeckung und Kantenüberdeckung. Begründen Sie, welches dieser Kriterien stärker ist.

(c) Geben Sie eine minimale Testmenge an, die das Kriterium der Kantenüberdeckung erfüllt. Geben Sie für jeden Fall den durchlaufenen Pfad und das erwartete Ergebnis an.
Hinweis: Eine Testmenge ist minimal, wenn die Zahl der Testfälle minimal ist. Die Minimalität braucht nicht bewiesen zu werden.

Aufgabe 3

[20 PUNKTE]

Eine Fußball-Liga hat einen Namen und besteht aus einer Menge von Mannschaften, die in Spielen gegeneinander antreten. Jede Mannschaft hat einen Namen und eine Position in der Tabelle. Außerdem werden für jede Mannschaft die Zahl der absolvierten Spiele, die Zahl der geschossenen und die Zahl der kassierten Tore sowie die Zahl der Punkte verwaltet. Jede Mannschaft hat genau einen Trainer und mehrere Spieler. Jeder Trainer und jeder Spieler hat einen Namen und ein Gehalt. Jeder Trainer und jeder Spieler ist genau einer Mannschaft zugeordnet. Jedes Spiel findet an einem bestimmten Spieltag statt; die Spieltage werden durchnummeriert. An jedem Spiel nehmen eine Heim-Mannschaft und eine Gast-Mannschaft teil; jede Mannschaft erzielt eine bestimmte Anzahl von Toren.

Erstellen Sie ein UML-Klassendiagramm auf der Basis der oben angegebenen textuellen Beschreibung. Verzichten Sie auf Operationen und Sichtbarkeiten. Wählen Sie für Attribute geeignete Datentypen. Verwenden Sie Vererbung, abstrakte Klassen und Komposition, wo dies möglich und sinnvoll ist. Verzichten Sie bei Assoziationen auf den Assoziationsnamen, benennen Sie jedoch die Assoziationsenden und ordnen Sie ihnen Multiplizitäten zu. Lassen Sie Navigationsrichtungen unspezifiziert.

Aufgabe 4

[25 PUNKTE]

Der Rezensions- und Publikationsprozess für einen Aufsatz in einer wissenschaftlichen Zeitschrift läuft folgendermaßen ab: Der Autor reicht den Aufsatz ein. Der Editor prüft, ob er für die Zeitschrift passend ist. Ist dies nicht der Fall, wird der Aufsatz abgelehnt. Andernfalls wird der Aufsatz von Rezensenten begutachtet, die vom Editor ausgewählt werden. Der Editor entscheidet, ob der Aufsatz abgelehnt, angenommen oder überarbeitet wird, und benachrichtigt den Autor. Im Falle einer geforderten Überarbeitung bricht der Autor den Prozess entweder ab oder reicht eine neue Version ein, die erneut begutachtet wird (Überarbeitungen sind prinzipiell beliebig oft möglich, bis eine finale Entscheidung getroffen wird). Ist der Aufsatz angenommen, reicht der Autor die finale Version ein. Der Redakteur des Verlags überführt sie in das Publikationsformat. Anschließend publiziert der Verlag den Aufsatz.

(a) Erstellen Sie ein UML-Anwendungsfalldiagramm für den oben beschriebenen Anwendungsfall.

(b) Erstellen Sie ein UML-Aktivitätsdiagramm, das den Ablauf beschreibt. Ordnen Sie dafür jede Aktion einem Akteur zu.

Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer

Kennzahl: _____

Kennwort: _____

Arbeitsplatz-Nr.: _____

Herbst 2022

46119

Erste Staatsprüfung für ein Lehramt an öffentlichen Schulen

— Prüfungsaufgaben —

Fach: **Informatik (Unterrichtsfach)**

Einzelprüfung: **Fachdidaktik - Realschulen**

Anzahl der gestellten Themen (Aufgaben): **3**

Anzahl der Druckseiten dieser Vorlage: **8**

Bitte wenden!

Thema Nr. 1

Stichworte: Algorithmik, Programmierung, Wiederholung, Variablen, Unterrichtsplanung

In der Neufassung des Moduls IT 1.2 legt der LehrplanPLUS für das Fach Informationstechnologie der Realschule Folgendes fest:

IT 1.2: Einführung in die Programmierung (ca. 14 Std.)

Die Schülerinnen und Schüler arbeiten mit bildungsorientierten Programmiersprachen bzw. Entwicklungsumgebungen, um mit grundlegenden algorithmischen Kenntnissen Programmcode für einfache Abläufe zu erstellen und sich mit fundamentalen Aspekten bei der Entwicklung von Software vertraut zu machen.

Kompetenzerwartungen

Die Schülerinnen und Schüler ...

- analysieren, interpretieren und formulieren Handlungsvorschriften zur Lösung von beschreibbaren Abläufen.
- analysieren Eingabe- und Ausgabeereignisse einfacher Programme.
- erstellen einfache Programme unter Verwendung von algorithmischen Grundbausteinen, Operatoren und Variablen.

Inhalte zu den Kompetenzen:

- Begriffe: Software, Programmiersprache, Entwicklungsumgebung
- Handlungsvorschriften zu Beispielen aus der Erfahrungswelt der Schülerinnen und Schüler, z. B. zur intelligenten Steuerung eines Roboters in Haushalt oder Garten
- Verwendung von Eingabeereignissen, z. B. Benutzereingaben, Sensorwerte
- Einbindung von Ausgabeereignissen, z. B. optische oder akustische Ausgaben
- Algorithmische Grundbausteine: Anweisung, Sequenz, einseitige und zweiseitige Auswahlstrukturen, Wiederholungsstrukturen mit fester Anzahl und Anfangsbedingung
- Operatoren: Rechenoperatoren, Vergleichsoperatoren
- Variablen: Bezeichner, Wertzuweisung

Aufgabe 1: Begründung der Kompetenzen und Inhalte
Durch die Neufassung des Lehrplans wird die Rolle von Algorithmik und Programmierung im Anfangsunterricht gestärkt. Begründen Sie die Notwendigkeit der aufgeführten Kompetenzen und Inhalte für den Anfangsunterricht im Fach Informationstechnologie! Denken Sie bei Ihrer Antwort daran, dass das Modul Pflicht für alle Ausbildungsrichtungen der Realschule ist (ca. 1½ Seiten)!

Aufgabe 2: Bildungsorientierte Programmiersprachen bzw. Entwicklungsumgebungen
Stellen Sie kurz je eine für den Einsatz in Modul IT 1.2 geeignete block- bzw. textorientierte Programmiersprache bzw. Entwicklungsumgebung vor! Wählen Sie anhand sinnvoller Kriterien eine der beiden für die Durchführung von Modul IT 1.2 aus! Begründen Sie Ihre Auswahl!

Aufgabe 3: Einführung der Wiederholung mit fester Anzahl
Zwei verschiedene Schulbücher verwenden eine textorientierte Programmierumgebung, in der eine Schildkröte gesteuert werden soll, die ihre abgelaufenen Wege markiert (sog. Turtle-Grafik). In den Büchern finden Sie zwei unterschiedliche Beispiele (mit Kommentaren) zur Motivation der Wiederholung mit fester Anzahl:

Buch 1: Ausgangssituation

```
vorwärts()          # 50 Pixel vorwärts
nachLinksDrehen()   # 90° nach links
vorwärts()
nachLinksDrehen()
vorwärts()
nachLinksDrehen()
vorwärts()
nachLinksDrehen()
```

Buch 2: Ausgangssituation

```
vorwärts(50)          # 50 Pixel vorwärts
nachLinksDrehen(90)   # 90° nach links
vorwärts(50)
nachLinksDrehen(90)
vorwärts(50)
nachLinksDrehen(90)
vorwärts(50)
nachLinksDrehen(90)
```

Buch 1: Wiederholung

```
wiederhole 4 mal:
    vorwärts()
    nachLinksDrehen()
```

Buch 2: Wiederholung

```
wiederhole 4 mal:
    vorwärts(50)
    nachLinksDrehen(90)
```

Nennen Sie kurz den Unterschied der beiden Schulbücher! Beurteilen Sie den Unterschied aus fachdidaktischer Sicht! Gehen Sie davon aus, dass die Schülerinnen und Schüler bereits alle in den beiden Ausgangssituationen vorkommenden Anweisungen anwenden können!

Aufgabe 4: Doppelstunde zur Einführung von Variablen
Im Folgenden planen Sie eine Doppelstunde zur Einführung von Variablen im Modul IT 1.2.

a. Nennen Sie die notwendigen inhaltlichen Lernvoraussetzungen für die von Ihnen geplante Doppelstunde!

b. Geben Sie drei beobachtbare Feinziele für die Doppelstunde an! Achten Sie darauf, dass Ihre Feinziele die Doppelstunde abdecken!

c. Beschreiben Sie den Unterrichtsverlauf der Doppelstunde nachvollziehbar in textueller Form! Begründen Sie die von Ihnen getroffenen Entscheidungen aus fachdidaktischer Sicht! Gliedern Sie den Text in Unterrichtsphasen und markieren Sie die Stellen, an denen Sie Ihre Feinziele aus b. erreicht haben!

Aufgabe 5: Einsatz von Robotiksystemen
Im Lehrplan werden in Beispielen mit Robotern und Sensorwerten zwei Aspekte aus dem Bereich der Robotik erwähnt. Denkbar wäre hier die Verwendung eines fahrbaren Roboters mit Abstands- und Linienfolgesensor. Gehen Sie davon aus, dass Sie eine ausreichende Anzahl solcher Systeme für Ihren Unterricht zur Verfügung haben!

Diskutieren Sie auf ca. einer Seite, welche Vor- und Nachteile die Verwendung eines solchen Systems im Rahmen des Moduls IT 1.2 hätte und treffen Sie anschließend eine begründete Entscheidung für bzw. gegen die Nutzung der Roboter!

Thema Nr. 2

Stichworte: Einführung in die Programmierung, visuelle vs. textbasierte Programmierung, Unplugged Aktivitäten, Unterrichtsplanung

Der LehrplanPLUS für das Fach Informationstechnologie an der Realschule legt im Lernbereich 1 des Anfangsunterrichts für das Modul IT 1.2 Folgendes fest:

IT 1.2: Einführung in die Programmierung (ca. 14 Std.)

Die Schülerinnen und Schüler arbeiten mit bildungsorientierten Programmiersprachen bzw. Entwicklungsumgebungen, um mit grundlegenden algorithmischen Kenntnissen Programmcode für einfache Abläufe zu erstellen und sich mit fundamentalen Aspekten bei der Entwicklung von Software vertraut zu machen.

Kompetenzerwartungen

Die Schülerinnen und Schüler ...

- analysieren, interpretieren und formulieren Handlungsvorschriften zur Lösung von beschreibbaren Abläufen.
- analysieren Eingabe- und Ausgabeereignisse einfacher Programme.
- erstellen einfache Programme unter Verwendung von algorithmischen Grundbausteinen, Operatoren und Variablen.

Inhalte zu den Kompetenzen:

- Begriffe: Software, Programmiersprache, Entwicklungsumgebung
- Handlungsvorschriften zu Beispielen aus der Erfahrungswelt der Schülerinnen und Schüler, z. B. zur intelligenten Steuerung eines Roboters in Haushalt oder Garten
- Verwendung von Eingabeereignissen, z. B. Benutzereingaben, Sensorwerte
- Einbindung von Ausgabeereignissen, z. B. optische oder akustische Ausgaben
- Algorithmische Grundbausteine: Anweisung, Sequenz, einseitige und zweiseitige Auswahlstrukturen, Wiederholungsstrukturen mit fester Anzahl und Anfangsbedingung
- Operatoren: Rechenoperatoren, Vergleichsoperatoren
- Variablen: Bezeichner, Wertzuweisung

Die Themen Nr. 1 und Nr. 2 enthalten beabsichtigt den gleichen Lehrplanauszug.

Aufgabe 1: Programmierung Unplugged

Zur Einführung in die Programmierung bietet es sich an, Unplugged-Aktivitäten im Unterricht einzusetzen. Wählen Sie einen der im Lehrplanausschnitt für das Modul IT 1.2 genannten Inhalte aus und schildern Sie eine Unplugged-Aktivität, die Sie zur Vermittlung des ausgewählten Inhalts im Unterricht durchführen könnten! Begründen Sie die gewählte Aktivität didaktisch und gehen Sie auf mögliche Hindernisse bei der Umsetzung ein!

Aufgabe 2: Grobplanung des Moduls
Entwerfen Sie eine Grobplanung des Moduls 1.2 in sieben Doppelstunden! Geben Sie dazu für jede Doppelstunde mindestens ein Grobziel an! Beschreiben Sie den Verlauf der jeweiligen Doppelstunden in ca. drei Sätzen! Arbeiten Sie Ihre Ergebnisse aus Aufgabe 1 in die Grobplanung ein und sehen Sie eine Doppelstunde zur Einführung von einseitigen und zweiseitigen Auswahlstrukturen vor!

Aufgabe 3: Unterrichtsplanung
Sie planen eine Doppelstunde zur Einführung von einseitigen und zweiseitigen Auswahlstrukturen. Berücksichtigen Sie bei der Planung Ihre Ergebnisse aus den vorangehenden Aufgaben!

a. Geben Sie drei beobachtbare Feinziele für Ihre Unterrichtseinheit an! Achten Sie darauf, dass Ihre Feinziele die Doppelstunde abdecken!

b. Beschreiben Sie den Verlauf der Doppelstunde in textueller Form! Gliedern Sie Ihren Unterrichtsverlauf nach Unterrichtsphasen, gehen Sie auf Sozialformen und Medien ein! Gliedern Sie den Text in Unterrichtsphasen und markieren Sie die Stellen, an denen Sie Ihre Feinziele aus a. erreicht haben!

c. Nennen Sie Ihre drei wichtigsten fachdidaktischen Entscheidungen und begründen Sie diese!

Thema Nr. 3

Stichworte: Unterrichtssequenz Digitale Medien, Unterrichtsstunde Datenbanken

Der LehrplanPLUS für das Fach IT enthält das Modul 1.9 in folgender, **hier gekürzter** Fassung:

IT 1.9 Digitale Medien

Die Schülerinnen und Schüler nutzen und analysieren digitale Geräte und Dienste, um diese [...] zu bewerten [...].

Kompetenzerwartungen

Die Schülerinnen und Schüler ...

- [...]
- ergreifen Maßnahmen zum Schutz von persönlichen Daten, Gesundheit und Umwelt, um problematischen Aspekten im Umgang mit digitalen Umgebungen vorzubeugen.
- entscheiden situativ, welche Daten sie bei der Nutzung digitaler Angebote von sich preisgeben und reflektieren Motivationen zur Auswertung sowie Methoden für die Zuordnung von Daten durch digitale Dienstleister.

Inhalte zu den Kompetenzen:

- [...]
- Maßnahmen zum Schutz der Privatsphäre (z. B. durch entsprechende Einstellungen bei Geräten, Betriebssystemen, Anwendungen und Diensten)
- problematische Aspekte im Umgang mit digitalen Umgebungen (z. B. Gewalt, Betrug, Belästigung, Sucht, Manipulation, Desinformation, Ressourcenverschwendung) und geeignete Maßnahmen zum Schutz davor
- Preisgegebene Daten (z. B. Konsumgewohnheiten, Standortdaten) und Maßnahmen zur Datensparsamkeit
- Motivationen digitaler Dienstleister zur Auswertung und Verknüpfung von Daten (z. B. für optimierte Verkehrsführung aber auch gezielte Einflussnahme auf Kaufverhalten oder politische Meinung)
- Methoden digitaler Dienstleister zur Zuordnung von Daten zu Benutzern (z. B. Auswertung von Seitenbesuchen, Setzen von Cookies, Browser-Fingerprinting)

Aufgabe 1: Informatische Bildung
Das Modul IT 1.9 gehört zum Anfangsunterricht und ist für alle Schülerinnen und Schüler verpflichtend. Begründen Sie, warum Sie dieses Modul aus Perspektive der informatischen Bildung für ein selbstbestimmtes Leben der Schülerinnen und Schüler für nötig halten (ca. 1–1½ Seiten)! Die medienpädagogische Sinnhaftigkeit muss in diesem Kontext nicht erläutert werden.

Aufgabe 2: Grobplanung von Modul IT 1.9
Die zitierten Kompetenzerwartungen und Inhalte des Moduls lesen sich sehr vielschichtig und könnten sicherlich auch mehrere Module füllen. Im Modul IT 1.9 stehen jedoch maximal **fünf Doppelstunden** zur Verfügung (die restlichen beiden Doppelstunden sind für den hier gekürzten Teil nötig).

a. Als Lehrkraft stehen Sie hier daher vor dem Dilemma, den Unterricht entweder so zu konzipieren, dass die Schülerinnen und Schüler einen guten Überblick bekommen oder einzelne Themen exemplarisch herauszugreifen. Treffen Sie eine Entscheidung und begründen Sie diese kurz! Bewertet wird die Begründung.

b. Erstellen Sie eine zu Ihrer Antwort aus a. passende Grobplanung der fünf Doppelstunden, indem Sie zu jeder Doppelstunde ein Grobziel angeben! Skizzieren Sie grob den Ablauf der fünf Doppelstunden
(je ca. 2–3 Sätze oder aussagekräftige Satzfragmente)!

Aufgabe 3: Bezüge zwischen Digitalen Medien und Datenbanksystemen
Viele Schülerinnen und Schüler bearbeiten im Aufbauunterricht auch das Thema Datenbanksysteme. Im Modul IT 2.3.1 stehen Anwendungsgebiete von Datenbanken, einfache Datenbestände und erste Abfragen im Fokus. Das Modul IT 2.3.2 erweitert das Szenario auf mehrere Tabellen. Die Kompetenzerwartungen der beiden Module finden Sie am Ende der Aufgabe.

a. Welche Inhalte aus Modul IT 1.9 erscheinen Ihnen geeignet, um motivierende Beispiele für einen Datenbestand mit einer Tabelle (Modul IT 2.3.1) bzw. einen Datenbestand mit verknüpften Tabellen (Modul IT 2.3.2) darzustellen? Nennen Sie für jedes Modul ein passendes Beispiel und erläutern Sie es kurz!

Nachfolgend wählen Sie ein Beispiel aus a. als Ausgangspunkt einer konkreten Planung einer Doppelstunde im Bereich Datenbanken aus!

b. Formulieren Sie ein Grobziel und fünf beobachtbare Feinziele, die Sie im Rahmen Ihrer Doppelstunde erreichen möchten! Achten Sie darauf, dass Ihre Feinziele die Doppelstunde abdecken! Zwei der Feinziele sollten einen erkennbaren Bezug zu den Kompetenzerwartungen aus Modul IT 1.9 haben.

c. Beschreiben Sie danach die komplette Doppelstunde nachvollziehbar in Textform! Gliedern Sie dabei Ihre Ausführungen durch Überschriften in Unterrichtsphasen und markieren Sie die Stellen, an denen Sie die Feinziele als erreicht ansehen!

d. Nennen Sie drei wichtige fachdidaktische Entscheidungen Ihrer Doppelstunde und begründen Sie diese jeweils kurz!

Aufgabe 4: Bezüge zwischen Modulen allgemein
Verknüpfungen zwischen Modulen wie in Aufgabe 3 sind im Lehrplan nicht explizit vorgesehen. Nennen Sie für solche Verbindungen mögliche Vor- und Nachteile! Welche Argumente überwiegen aus Ihrer Sicht? Bewertet wird eine schlüssige Begründung.

Als Information zu Aufgabe 3 sind nachfolgend die Kompetenzerwartungen der Module IT 2.3.1 und IT 2.3.2 genannt:

IT 2.3.1 Datenbanksysteme I (ca. 14 Std.)

Die Schülerinnen und Schüler analysieren die Struktur eines einfachen Datenbestandes, um sie mit einem Datenbanksystem umzusetzen und mithilfe von Abfragen auszuwerten.

Kompetenzerwartungen

Die Schülerinnen und Schüler ...

- analysieren Anwendungsgebiete für Datenbanksysteme (z. B. Onlineshop), um einen Zusammenhang zwischen Dateneingabe und Datennutzung herzustellen.
- modellieren die Struktur eines einfachen Datenbestandes (z. B. Adressdaten) und implementieren diesen mit einem Datenbanksystem.
- erstellen Abfragen, um gewünschte Teilmengen eines Datenbestandes zu ermitteln und in Form von Berichten darzustellen.
- nutzen Schnittstellen, um mit externen Programmen auf einen Datenbestand zuzugreifen.

IT 2.3.2 Datenbanksysteme II (ca. 14 Std.)

Die Schülerinnen und Schüler modellieren einen umfangreichen Datenbestand (z. B. zur Abwicklung einer Buchausleihe), um ihn in einem relationalen Datenbanksystem umzusetzen und mit Abfragen über mehrere Tabellen auszuwerten.

Kompetenzerwartungen

Die Schülerinnen und Schüler ...

- modellieren ein Szenario mithilfe mehrerer Tabellen (z. B. Kunden, Produkte und Bestellungen eines Online-Kaufhauses), um Daten und ihre Beziehungen darzustellen und korrekt zu speichern.
- verwenden Abbildungsregeln, um das erstellte Modell in einem relationalen Datenbanksystem umzusetzen.
- erstellen Abfragen über mehrere Tabellen, um verknüpfte Informationen aus dem Datenbestand zu erhalten.
- generieren Formulare, um eingegebene Daten zu validieren und in der Datenbank abzulegen.