Baden-Württemberg

Leistungsfach Informatik

Schriftliche Abiturprüfung
ab 2023

Informationen

Beispielaufgabe

Aufgabenfundus

Ab der Abiturprüfung 2023 ändert sich das Aufgabenformat in der schriftlichen Abiturprüfung im Leistungsfach Informatik. Im Folgenden werden die wesentlichen Änderungen dargestellt.
Beispielaufgabe und Aufgabenfundus enthalten exemplarisch, d. h. in keiner Weise auf Vollständigkeit angelegt, Aufgabenstellungen zu neuen Inhalten nach dem Bildungsplan 2016, die 2023 erstmals Gegenstand der Abiturprüfung sein können.
Insbesondere verdeutlicht die Beispielaufgabe den neuen Zuschnitt der Themenverteilung.

Wenn im folgenden auf ZPG-Materialien verwiesen wird, sind die aktuell laufenden Fortbildungen „ZPG: Informatik in der Kursstufe (Teile 1 bis 4)“ gemeint. Die Fortbildungen sind unter https://lfbo.kultus-bw.de/lfb/suche recherchierbar. Die zugehörigen Materialien werden demnächst auf dem Lehrerfortbildungsserver freigeschaltet:
https://lehrerfortbildung-bw.de/u_matnatech/informatik/gym/bp2016/fb2/

Ministerium für Kultus, Jugend und Sport
Thouretstraße 6 (Postquartier)
70173 Stuttgart

Version 2.3 vom 20.03.2022

Inhalt

Struktur eines Aufgabensatzes

bisher 2017 – 2022 Bearbeitungszeit: 240 min (270 min in den Jahren 2021 und 2022)

Pflichtaufgabe A*	**Wahlaufgabe B1**	**Wahlaufgabe B2**	**Wahlaufgabe B3**
Objektorientierte Modellierung und Programmierung	Datenbanken	Automaten und formale Sprachen	Abstrakte Datentypen
(20 VP)	(20 VP)	(20 VP)	(20 VP)

← *Lehrerwahl* →

* In den Jahren 2021 und 2022 wurden der Lehrkraft zwei Pflichtaufgaben A1 und A2 vorgelegt.

Die Lehrkraft wählt (eine Aufgabe A und) zwei Aufgaben aus den Schwerpunktgebieten B.
Die Schülerin / der Schüler erhält drei Aufgabenteile (eine Aufgabe A und zwei Aufgaben B_i) und bearbeitet diese drei Aufgabenteile vollständig.

ab 2023 Bearbeitungszeit: 270 min

Pflicht-aufgabe A	**Wahlaufgabe B1**	**Wahlaufgabe B2**	**Wahlaufgabe B3**	**Wahlaufgabe B4**
Pflichtteil aus 5 – 7 Basisaufgaben à 2 – 3 VP	Entwurf / Analyse von Programmen und Programmiertechniken	Algorithmen und Datenstrukturen	Automaten / Formale Sprachen / technische Informatik	Datenbanken / Kryptologie / Datenschutz
(15 VP)	(15 VP)	(15 VP)	(15 VP)	(15 VP)

← *Schülerwahl* →

Die Schülerin / der Schüler erhält alle fünf Aufgabenteile. Er bearbeitet die Pflichtaufgabe A und drei Aufgaben aus den Schwerpunktgebieten B nach seiner Wahl, insgesamt vier Aufgabenteile.

29. Informatik

Der Abiturjahrgang 2023 wird – mit Ausnahme der am Schulversuch Informatik teilnehmenden Schulen – nach dem Bildungsplan 2004 unterrichtet.

- **Schulen, die nicht am Schulversuch Informatik teilnehmen**, können nur das bisherige Wahlfach nach dem Bildungsplan 2004 anbieten, wozu Informationen unter 29.4 zu finden sind.
- **Schulversuchsschulen, die am Schulversuch nur mit Modul A teilnehmen** (bestehend aus Brückenkurs, Basisfach und dem neuen Wahlfach), finden die zugehörigen Informationen unter 29.2 und 29.3.
- **Schulversuchsschulen, die gemäß Erlass auch das Leistungsfach anbieten dürfen** (Module A und B), finden die zugehörigen Informationen unter 29.1, 29.2 und 29.3.

29.1 Leistungsfach (Schulversuch)

29.1.1 Verbindliche Inhalte

Dem Unterricht und der Prüfung liegen die im Bildungsplan 2016 für das Leistungsfach Informatik ausgewiesenen Inhalte und Kompetenzen für den Schulversuch Informatik zugrunde: http://www.bildungsplaene-bw.de/,Lde/LS/BP2016BW/ALLG/GYM/INF **(3.1 Brückenkurs und 3.3 Leistungsfach)**.

Die folgenden Themen des Bildungsplans sind **nicht** Gegenstand der schriftlichen Prüfung:

3.3.3. Rechner und Netze, inhaltsbezogene Kompetenzen (8) bis (13) (von-Neumann-Rechner, Rechnernetze)

29.1.2 Leistungsmessung

In der Qualifikationsphase sind in den ersten drei Schulhalbjahren jeweils mindestens zwei Klausuren und im vierten Schulhalbjahr mindestens eine Klausur anzufertigen.

Die Klausuren sind so zu stellen, dass jeweils Leistungen aus allen drei Anforderungsbereichen eingefordert werden. Der Schwerpunkt der zu erbringenden Prüfungsleistung liegt im Anforderungsbereich II. Der Anforderungsbereich I ist gegenüber dem Anforderungsbereich III stärker zu akzentuieren.

Der zeitliche Umfang einer Klausur beträgt in der Regel zwei Unterrichtsstunden.

Im Übrigen gelten die Regelungen der Notenbildungsverordnung auch in den beiden Jahrgangsstufen (vgl. § 11 Absatz 2 Notenbildungsverordnung).

29.1.3 Schriftliche Prüfung

Bearbeitungszeit: **270** Minuten einschließlich Auswahlzeit

Hilfsmittel:

- Nachschlagewerke zur deutschen Rechtschreibung
- Der im jeweiligen Kurs eingeführte wissenschaftliche Taschenrechner (WTR) mit dem mitgelieferten Handbuch.
  Hierzu sind die Ausführungen in der Anlage des Erlasses des Kultusministeriums vom 26.02.2014 (Az.: 36/45-6624.03-P/234) zu beachten.
  Vor Prüfungsbeginn ist sicherzustellen, dass alle Speicherinhalte auf den wissenschaftlichen Taschenrechnern der Schülerinnen und Schüler gelöscht sind.

Der Fachlehrerin, dem Fachlehrer werden die Aufgabenstellungen

Pflichtteil A:
eine Sammlung von Basisaufgaben aus allen Themengebieten des Bildungsplanes

sowie vier Wahlaufgaben **B1**, **B2, B3** und **B4** mit verschiedenen Schwerpunkten aus den folgenden Themengebieten vorgelegt:

B1: eine Aufgabe mit dem Schwerpunkt **Entwurf / Analyse von Programmen und Programmiertechniken**
B2: eine Aufgabe mit dem Schwerpunkt **Algorithmen und Datenstrukturen**
B3: eine Aufgabe mit dem Schwerpunkt **Automaten / Formale Sprachen / technische Informatik**
B4: eine Aufgabe mit dem Schwerpunkt **Datenbanken / Kryptologie / Datenschutz**

Die Schülerin, der Schüler

- erhält den **Pflichtteil A sowie alle vier Wahlaufgaben aus Gruppe B** und wählt **drei** der vier Wahlaufgaben aus Gruppe B zur Bearbeitung aus;
- ist verpflichtet, die Vollständigkeit der vorgelegten Aufgaben vor Bearbeitungsbeginn zu überprüfen (Anzahl der Blätter, Anlagen usw.);
- bearbeitet die **Aufgabe A** und die **drei** ausgewählten **Aufgaben der Gruppe B;**
- vermerkt auf der Reinschrift, welche Aufgaben sie/er bearbeitet hat.

Die Implementierung von Programmen oder Programmteilen erfolgt in der im jeweiligen Kurs eingeführten Programmiersprache.
Bei der Programmierung ist davon auszugehen, dass sämtliche erforderlichen Daten in einem vernünftigen Rahmen eingegeben werden. An eine Fehlerbehandlung von Falscheingaben ist nicht gedacht.

...

Aufgabenzuschnitt

Zielsetzung:
Die bisherige starke Fokussierung auf OOP/OOM und ADTs sollte zurückgefahren werden.
Trotzdem soll das Abitur einen nennenswerten Anteil an Programmieraufgaben enthalten, damit dem großen Anteil der Programmierung im Unterricht Rechnung getragen wird.
Die Bereiche Datenbanken/Kryptologie/Datenschutz und Formale Sprachen/Automaten müssen vielfältiger werden, da die zur Verfügung stehenden Aufgabenmöglichkeiten zu begrenzt waren.
Alle Schüler sollen die grundlegenden Techniken aus allen Bereichen beherrschen.
Eine Wahlmöglichkeit für die Schüler soll erhalten bleiben.
Auch Themen aus der Schnittmenge des Brückenkurses und des Informatikanteils von IMP können in der Abiturprüfung behandelt werden.

Neues Aufgabenformat ab dem Abitur 2023:
Das neue Abitur enthält fünf Aufgabenteile: einen Pflichtteil und vier Wahlteile.
Nicht mehr die Lehrkraft, sondern die Schülerin / der Schüler wählt aus: sie / er bearbeitet die **Pflichtaufgabe A** und die **drei** ausgewählten **Aufgaben der Gruppe B.**
Jeder dieser Aufgabenteile umfasst 15 Verrechnungspunkte.

Pflichtteil A:

Der Pflichtteil überprüft an kleinen unzusammenhängenden Aufgaben grundlegende Programmiertechniken, Standardverfahren und Definitionen aus allen Bildungsplanbereichen.
Es werden eine Reihe von Aufgabentypen definiert, die aber nicht alle jedes Jahr vorkommen müssen. Jede Aufgabe wird in der Regel mit 2 – 3 VP bewertet, somit ergeben sich ca. fünf bis sieben Aufgaben.

Aufgabentypen:
- Standardalgorithmen (kürzeste Wege, LZW, Huffman usw.) erläutern, anwenden usw.
- Array oder Zeichenketten-Implementationen (z. B. zweitgrößtes Element bestimmen, Mittelwert, Palindromtest)
- Analyse von Laufzeitverhalten (Quelltexte oder Nassi-Shneiderman analysieren), Laufzeiten von Sortieralgorithmen
- Beziehungen bei Datenbanken und OOM (z. B. Kardinalitäten bestimmen, Assoziationen/Vererbung begründen, Umwandlung in ein relationales Datenbankschema)
- Rekursion (einfache Funktionen implementieren, Rekursion erläutern, rekursive Aufrufe nachvollziehen)
- SQL-Anfragen (SQL-Anfragen formulieren und interpretieren)
- Begriffsklärungen (z. B. Primärschlüssel / Vererbung / Endzustand / öffentlicher Schlüssel / Variablentyp)
- Logische Schaltungen (Halbaddierer / Volladdierer angeben, Wahrheitstafel zu einer Schaltung bestimmen)
- Formale Sprachen / Automaten (gehört Wort zu einer Sprache? …)
- Codierung (Zweierkomplement / Einerkomplement / Festkommazahlen, Rechnen mit diesen Zahlen usw.)
- Analysieren und Bewerten (von Code, Verfahren, ...)
- Datentypen: Liste, Queue, Stack, Baum, Graph
- und weitere...

Wahlbereiche B1 bis B4:

Es gibt vier Wahlbereiche, deren thematische Schwerpunkte festgelegt sind. Trotzdem können insbesondere bei Aufgaben aus dem Anforderungsbereich III in jeder Wahlaufgabe auch Inhalte aus anderen Bereichen vorkommen.

Bereich 1: Entwurf/Analyse von Programmen und Programmiertechniken
- OOP / OOM
- Begründung der Entwurfsentscheidung
- Rekursion (Backtracking, Divide & Conquer usw.)
- Aufwandsanalysen
- Sortierverfahren / Suchverfahren (auf Arrays)
- Codeanalyse
- ...

Bereich 2: Algorithmen und Datenstrukturen
- rekursive Datenstrukturen (Binärbaum, verkettete Listen und Algorithmen darauf)
- Suchen und Sortieren
- ADTs (Schlange, Stapel unter Verwendung generischer Typen)
- Graphen (auch Algorithmen auf Graphen)
- Codeanalyse
- ...

Bereich 3: Automaten / Formale Sprachen / technische Informatik
- Grammatiken
- Syntaxdiagramme
- Automaten (auch Mealy-Automaten)
- Schaltungen
- ...

Bereich 4: Datenbanken / Kryptologie / Datenschutz
- ERD und UML für Datenbanken (entwerfen, modellieren)
- SQL-Anfragen
- symmetrische & asymmetrische Kryptologie, Zertifikate usw.
- Anwendung kryptographischer Hashfunktionen
- ggfs. Sortierverfahren auch hier
- ...

Bei Wahlbereichen, die mehrere Themengebiete umfassen, müssen Aufgaben einzelner Jahre nicht alle Themengebiete abdecken, sondern können sich von Jahr zu Jahr unterscheiden (z. B. eine reine Kryptologieaufgabe aus dem Bereich 4 oder eine Aufgabe aus der technischen Informatik in Bereich 3).

Operatoren

In den Standards für inhaltsbezogene Kompetenzen werden Operatoren (handlungsleitende Verben) verwendet. Standards legen fest, welche Anforderungen die Schülerinnen und Schüler in der Regel erfüllen. Zusammen mit der Zuordnung zu einem der drei Anforderungsbereiche (AFB) dienen Operatoren einer Präzisierung. Dies sichert das Erreichen des vorgesehenen Niveaus und die angemessene Interpretation der Standards.

Beschreibung der drei Anforderungsbereiche

- **Anforderungsbereich I** umfasst das Wiedergeben von Sachverhalten und Kenntnissen sowie das Anwenden und Beschreiben geübter Arbeitstechniken und Verfahren.
- **Anforderungsbereich II** umfasst das selbstständige Verarbeiten und Darstellen bekannter Sachverhalte in einem durch Übung bekannten Zusammenhang und das selbstständige Übertragen des Gelernten auf vergleichbare, neue Sachverhalte.
- **Anforderungsbereich III** umfasst das Verarbeiten komplexer Sachverhalte mit selbstständiger Auswahl geeigneter Arbeitstechniken mit dem Ziel, zu selbstständigen Lösungen, Gestaltungen oder Deutungen, Folgerungen, Verallgemeinerungen, Begründungen und Wertungen zu gelangen und das eigene Vorgehen zu reflektieren.

Zuordnung zu Anforderungsbereichen

Die Zuordnung eines Operators ist im Einzelfall auch vom Kontext der Aufgabenstellungen und ihrer unterrichtlichen Einordnung abhängig. Im Folgenden werden die Operatoren dem überwiegend in Betracht kommenden Anforderungsbereich zugeordnet.

Operatoren	Beschreibung	AFB
analysieren	eine konkrete Materialgrundlage unter einer gegebenen Fragestellung auf wichtige Bestandteile, Eigenschaften oder Zusammenhänge untersuchen	III
angeben	Ergebnisse numerisch oder verbal formulieren, ohne Darstellung des Lösungsweges und ohne Begründungen	I
anwenden, nutzen, umgehen mit, verwenden	Fachbegriffe, Regeln, mathematische Sätze, Zusammenhänge oder Verfahren auf einen (anderen) Sachverhalt beziehen	II
begründen	eine Aussage oder einen Sachverhalt durch Berechnungen, nach gültigen Schlussregeln, durch Herleitungen oder inhaltliche Argumentation verifizieren oder falsifizieren	III
berechnen	Ergebnisse von einem Ansatz oder einer Formel ausgehend durch Rechenoperationen gewinnen	I
beschreiben	Strukturen, Sachverhalte, Verfahren, Prozesse und Eigenschaften von Objekten in der Regel unter Verwendung der Fachsprache in vollständigen Sätzen wiedergeben (hier sind auch Einschränkungen möglich: „Beschreiben Sie in Stichworten“) beziehungsweise in einer vorgeschriebenen Form darstellen (zum Beispiel: „Beschreiben Sie als Term“)	II
bestimmen, ermitteln, erschließen	Lösungen, Lösungswege beziehungsweise Zusammenhänge auf der Basis von Vorkenntnissen oder Verfahren rechnerisch, grafisch oder experimentell finden und darstellen	II
bewerten	einen Sachverhalt nach fachwissenschaftlichen oder fachmethodischen Kriterien, persönlichem oder gesellschaftlichem Wertebezug begründet einschätzen und ein selbstständiges Urteil formulieren	III

darstellen	Zusammenhänge, Sachverhalte oder Arbeitsverfahren in strukturierter oder formal definierter Form (zum Beispiel grafisch) wiedergeben	II
durchführen	nach bekannten Regeln oder Anweisungen von einer Aufgabenstellung zu einem definierten Ziel gelangen	II
entwerfen	nach vorgegebenen Bedingungen ein sinnvolles Konzept selbstständig planen/erarbeiten	III
erklären	Sachverhalte, Strukturen, Prozesse und Zusammenhänge erfassen sowie auf Vorkenntnisse oder allgemeine Aussagen und Gesetze unter Verwendung der Fachsprache zurückführen	II
erläutern	Sachverhalte, Strukturen, Prozesse und Zusammenhänge erfassen sowie auf Vorkenntnisse oder allgemeine Aussagen und Gesetze unter Verwendung der Fachsprache zurückführen und durch zusätzliche Informationen oder Beispiele verständlich machen	II
erstellen	Herstellen und Gestalten eines Systems unter vorgegebener Zielsetzung	II
ergänzen, erweitern	weitere Bestandteile zu einem gegebenen Sachverhalt hinzufügen	II
identifizieren	Objekte, Muster oder Strukturen und die zugehörigen Fachbegriffe begründet miteinander verbinden	I
implementieren	Datenstrukturen oder Algorithmen in einer Programmiersprache umsetzen	II
interpretieren	Sachverhalte und Zusammenhänge im Hinblick auf Erklärungsmöglichkeiten untersuchen und abwägend herausstellen	III
kommentieren	einen gegebenen Sachverhalt oder einen gegebenen Algorithmus mit erläuternden Hinweisen versehen	I
modellieren	zu einem Ausschnitt der Realität ein informatisches Modell erstellen	II
nennen	Elemente, Sachverhalte, Begriffe, Daten, Fakten ohne Erläuterung wiedergeben	I
überführen	eine Darstellungsform in eine andere Darstellungsform bringen überprüfen	II
überprüfen	durch Anwendung fachlicher Regeln oder Kenntnisse in einer ergebnisoffenen Situation einen vorgegebenen Sachverhalt verifizieren oder falsifizieren	III
untersuchen	Objekte, Sachverhalte und Fragestellungen nach fachlichen Kriterien zielorientiert erkunden und Zusammenhänge herausarbeiten	II
vergleichen	Gemeinsamkeiten, Ähnlichkeiten und Unterschiede herausarbeiten zuordnen	II
zuordnen	einen begründeten Zusammenhang zwischen Objekten, Strukturen und Darstellungen herstellen	II

Quelle: BP2016BW-ALLG-GYM-INF (Seiten 49/50)

A Pflichtteil

A1

(a) Überführen Sie die Dezimalzahl -13_{10} in 8-Bit-Zweierkomplementdarstellung.

(b) Berechnen Sie schriftlich in 8-Bit-Zweierkomplementdarstellung die Summe aus $0001\ 1000_2$ und -13_{10}.

(2 VP)

Meta 1: Es wird die Schreibweise 101_{10} für das Dezimalsystem, 0101_{16} für das Hexadezimalsystem, $0000\ 0101_2$ für das Binärsystem verwendet. Zweierkomplementdarstellungen werden im Aufgabentext explizit benannt.

Meta 2: Zahlen außerhalb des Zehnersystems werden mit führenden Nullen byteweise aufgefüllt, d. h. bei Binärzahlen mindestens ein Oktett.

Meta 3: Bei schriftlichen Berechnungen ist der Rechenweg mit Überträgen darzustellen.

A2 Gegeben ist die folgende Adjazenzmatrix:

		nach				
		A	**B**	**C**	**D**	**E**
von	**A**		X	X		X
	B	X			X	
	C				X	
	D		X			X
	E				X	

(a) Überführen Sie die Adjazenzmatrix in ein Diagramm des gerichteten Graphen.

(b) Erläutern Sie, wie man der Adjazenzmatrix ansehen kann, dass der Graph gerichtet ist.

(3 VP)

A3 Gegeben ist ein optimiertes relationales Datenbankschema:
Schueler (<u>SNr</u>, Vorname, Nachname)
Kurs (<u>KNr</u>, Name)
besucht (<u>SNr</u> ↑, <u>KNr</u> ↑)

Überführen Sie das Datenbankschema in ein ER- oder UML-Diagramm unter Angabe der Beziehungskardinalitäten.

(1,5 VP)

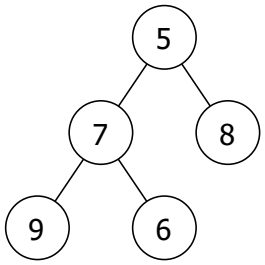
A4 Implementieren Sie eine Methode `filterGerade`, die eine Liste `e` von natürlichen Zahlen als Parameter bekommt und eine neue Liste zurückgibt, die nur noch die geraden Zahlen aus `e` enthält.

(2 VP)

Meta 1: Es ist in solchen Aufgabe nicht ausdrücklich verlangt, Collection-Befehle und Lambda-Ausdrücke zu kennen. Es können aber Aufgaben gestellt werden, die damit deutlich einfacher werden.

Meta 2: In der Formulierung der Aufgabe wurde die Art der Liste bewusst offen gelassen, um sprachunabhängig zu bleiben.

A5 Gegeben ist ein Binärbaum:



(a) Geben Sie die Reihenfolge der Knoten bei einer Preorder-Traversierung und einer Postorder-Traversierung an.

(b) Geben Sie einen anderen Binärbaum an, dessen Preorder-Traversierung die gleiche Knotenabfolge liefert.

(2 VP)

A6

(a) Bestimmen Sie eine weitestmöglich vereinfachte Form des folgenden booleschen Terms: $A \vee \neg(\neg B \wedge C) \wedge C$

(b) Überführen Sie den vereinfachten booleschen Term in ein Schaltbild.

(2,5 VP)

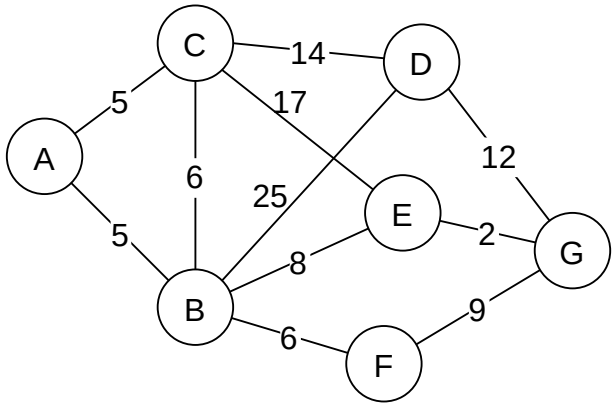
Meta 1: Der hier dargestellte Term stellt die obere Grenze des zu erwartenden Komplexitätsgrads dar.

Meta 2: Es wird die Schreibweise der Aussagenlogik (mit Haken für „NICHT“ und Symbolen für UND/ODER-Operatoren wie oben) verwendet. Weitere Symbole werden nicht verwendet. Andere Schreibweisen in Schülerlösungen werden akzeptiert.

Meta 3: Ein Term ist weitestmöglich vereinfacht, wenn die Anzahl der Operatoren minimal ist.

A7

(a) Bestimmen Sie den minimalen spannenden Baum durch Kanten-Markierung innerhalb des untenstehenden Graphen:



(b) Beschreiben Sie ein Anwendungsproblem, für das ein minimaler spannender Baum eine Lösung liefert.

(2 VP)

B1 Entwurf / Analyse von Programmen und Programmiertechniken

B1.1 Schätzungen zufolge werden 25-30% der weltweiten Rechenzeit für das Sortieren aufgewendet. Eines der einfachsten Sortierverfahren ist Bubblesort.

(a) Implementieren Sie den Algorithmus Bubblesort in der Methode `bubbleSort(daten: int[])`.

Der Sortieralgorithmus Shakersort (auch als „Cocktailsort“ bekannt) ist eine Variante von Bubblesort. Während Bubblesort das Array immer von links nach rechts bearbeitet, wechselt Shakersort nach jedem Durchgang die Richtung. Für den Rest dieser Aufgabe wird angenommen, dass sowohl Bubblesort als auch Shakersort abbrechen, sobald ein Durchgang ohne Vertauschungen abgeschlossen wurde.

(b) Stellen Sie den Inhalt des Arrays `{ 45, 12, 99, 37, 2 }` nach jeder Vertauschungsoperation des Shakersort-Algorithmus dar. Dabei erfolgt der erste Durchlauf von links nach rechts.

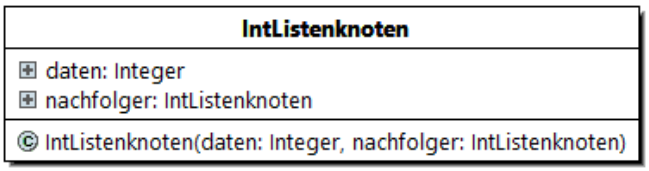
(c) Erläutern Sie den Begriff „worst case“ im Zusammenhang mit Sortieralgorithmen.

(d) Begründen Sie, warum das Array `{ 2, 3, 4, 5, …, 99, 100, 1 }` für Bubblesort einen worst case in Bezug auf die Anzahl der Vergleiche darstellt, für Shakersort aber nicht.

(8 VP)

B1.2 Ein effizienteres Verfahren zum Sortieren von unsortierten Daten ist Mergesort.

(a) Beschreiben Sie, wie der Algorithmus Mergesort arbeitet.

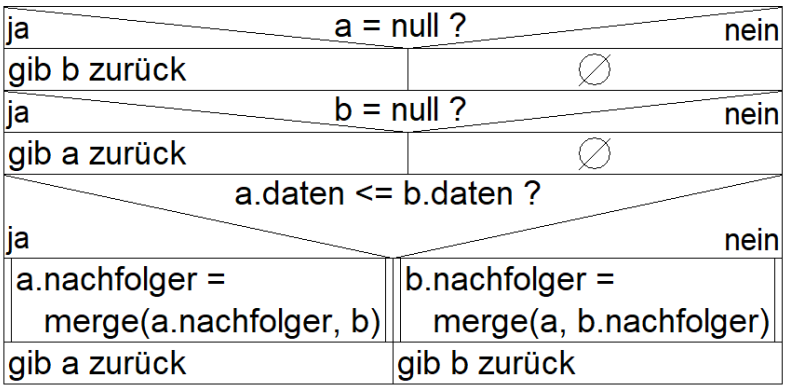


Mergesort kann auch auf verketteten Listen effizient ausgeführt werden. Dabei wird eine Hilfsmethode
`merge(a: IntListenknoten, b: IntListenknoten): IntListenknoten`
verwendet, die zwei sortierte Listen übergeben bekommt und diese zu einer einzigen sortierten Liste zusammenfügt und diese zurückgibt. Die Listen werden jeweils durch den Verweis auf ihren ersten Knoten repräsentiert.

(b) Erläutern Sie anhand des gegebenen Nassi-Shneiderman-Diagramms, wie die `merge`-Methode arbeitet.

Die Liste `a` habe die Länge n, die Liste `b` habe die Länge m. Es wird ein Aufruf `merge(a, b)` ausgeführt.

merge(a: IntListenknoten, b: IntListenknoten): IntListenknoten



(c) Analysieren Sie die Methode `merge` in Bezug auf die Anzahl der rekursiven Aufrufe in Abhängigkeit von n und m. Geben Sie für die Anzahl eine Unter- und eine Obergrenze an.

(7 VP)

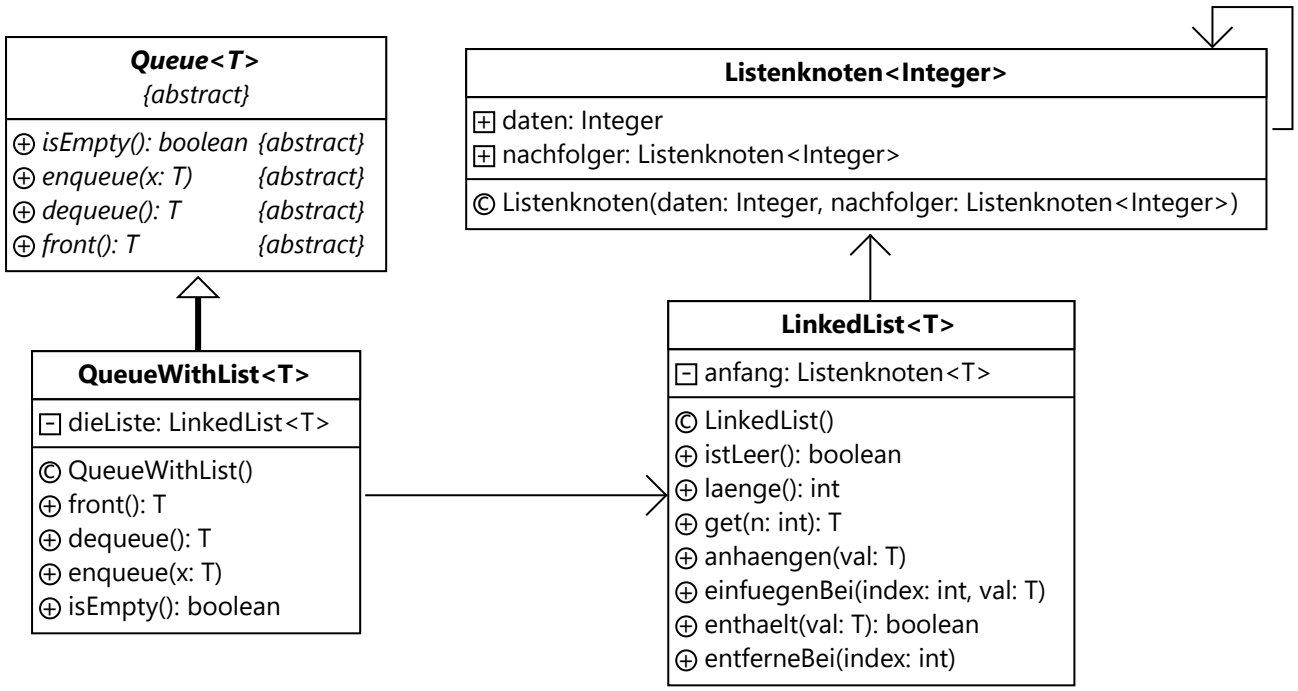
Meta 1: Erläuterungen zu Nassi-Shneiderman-Diagrammen oder Quelltexten (Pseudocode) können gefragt werden. Es ist nicht gedacht, zeilenweise zu übersetzen, was da steht, sondern die Wirkungsweise im Zusammenhang und den Aufbau des Codes zu erklären.

Meta 2: Es wird nicht verlangt, selbst Nassi-Shneiderman-Diagramme oder Pseudocode zu erstellen.

Meta 3: Dies ist ein Beispiel für das Mischen von Themengebieten.

Meta 4: Es kann Aufgaben geben, in denen keine OOP vorkommt. Es kann aber auch sein, dass wie in den früheren Jahren reine OOP-Aufgaben gestellt werden.

B2 Algorithmen und Datenstrukturen



B2.1 Gegeben ist das obenstehende UML-Diagramm der Klassen für Liste und Queue (Schlange).

(a) Implementieren Sie die Klasse `Listenknoten<T>`.

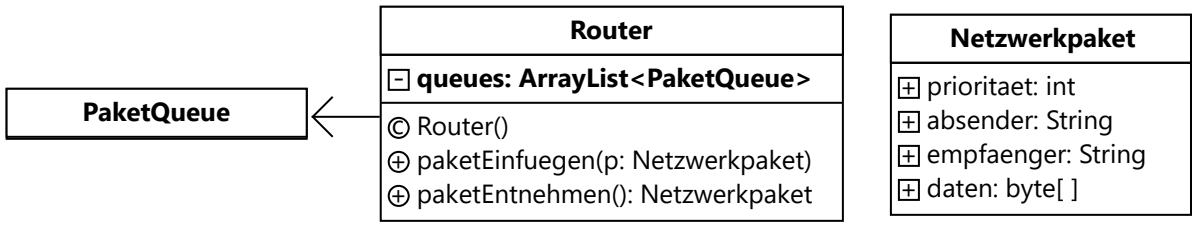
(b) Implementieren Sie die Methoden `enqueue(x: T)` und `dequeue(): T` in der Klasse `QueueWithList<T>`. Sie können dabei alle Methoden der Klasse `LinkedList<T>` verwenden.

(c) Implementieren Sie die Methode `anhaengen(val: T)` der Klasse `LinkedList<T>`. (2 VP)

(5 VP)

B2.2 Bei einem Router kommen Datenpakete an und werden weitergeleitet. Pakete haben eine Priorität, zum Beispiel erhalten Pakete für Videostreaming und -telefonie höhere Prioritäten als E-Mails. Dafür hat der Router mehrere Queues.

Jedes mit der Methode `paketEinfuegen(p: Netzwerkpaket)` empfangene `Netzwerkpaket` wird in diejenige `PaketQueue` eingefügt, die seiner Priorität entspricht. Falls es noch keine `PaketQueue` zur Priorität gibt, wird eine neue angelegt.
Sobald die ausgehende Leitung frei ist, wird die Methode `paketEntnehmen(): Netzwerkpaket` aufgerufen, um das nächste abzuschickende Paket zu bestimmen. Dabei wird die `PaketQueue` mit der höchsten Priorität gesucht und das nächste Paket daraus entfernt.



(a) Modellieren Sie die Klasse `PaketQueue` mit Hilfe eines UML-Diagramms. Dieses soll alle Attribute und Methoden enthalten. Sie können die Klassen, die im Aufgabenteil B2.1 dargestellt sind, verwenden.

(b) Implementieren Sie die Methode `paketEinfuegen(p: Netzwerkpaket)`.

(c) Implementieren Sie die Methode `paketEntnehmen(): Netzwerkpaket`. Sie gibt `null` zurück, falls keine der Queues ein weiterzuleitendes Paket enthält.

(7 VP)

B2.3 Ein großer Provider, der Privatkunden den Internetzugang anbietet, möchte auch mit Streamingdiensten Verträge abschließen: Wer für den „Premium-Service“ bezahlt, dessen Pakete erhalten automatisch eine höhere Priorität.
Dieser Service entspricht nicht dem Prinzip der Netzneutralität:
„Netzneutralität bezeichnet die Gleichbehandlung von Daten bei der Übertragung im Internet und den diskriminierungsfreien Zugang bei der Nutzung von Datennetzen. Netzneutrale Internetdienstanbieter behandeln alle Datenpakete bei der Übertragung gleich, unabhängig von Sender und Empfänger [und] dem Inhalt der Pakete [...]“
Wikipedia, Artikel „Netzneutralität“, Abruf vom 30.6.2021

(a) Bewerten Sie das Vorhaben „Premium-Service“ aus den verschiedenen Perspektiven der Beteiligten.

(3 VP)

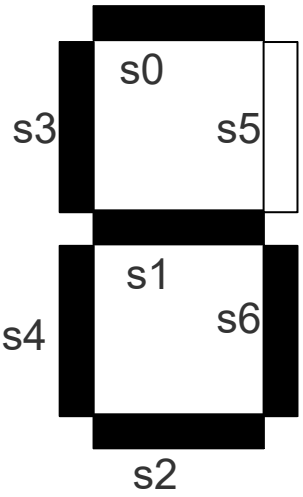
Meta 1: Die Betrachtung von gesellschaftlichen Auswirkungen der Informatik (vgl. pbK 2.4.8) kann in vergleichsweise offenen Fragestellungen wie dieser vorkommen. Dabei kann auch eine Stellungnahme der SuS verlangt werden (pbK 2.4.9).

Meta 2: Die Struktur der ADTs mit generischen Typen, abstrakten Oberklassen und Assoziation von Queue bzw. Stack auf die Datenstruktur (hier: Liste) wird in Zukunft als Standard angesehen. Die Attribute der Klasse `Listenknoten` sind `public`, weil eine Kapselung in diesem Fall keinen Mehrwert liefert (vgl. ZPG-Materialien).

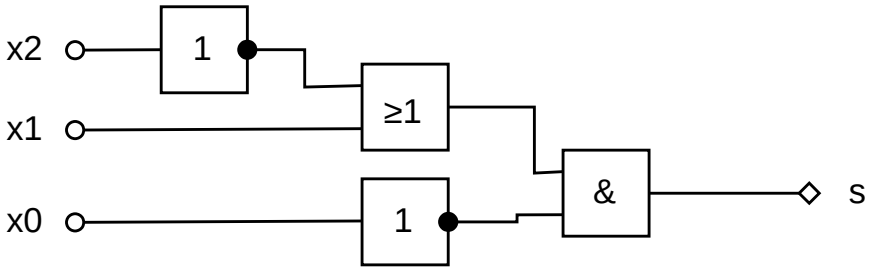
Meta 3: Einfache Einstiegsaufgaben wie „Beschreiben Sie die Arbeitsweise...“ kommen verstärkt im Pflichtteil und entfallen dafür in den Wahlaufgaben.

B3 Automaten / Formale Sprachen / technische Informatik

B3.1 Eine Sieben-Segment-Anzeige wie im Bild rechts wird immer noch häufig zur Anzeige von Ziffern verwendet. Im Beispiel wird die Ziffer 6 dargestellt. Eine Schaltung zur Ansteuerung der Sieben-Segment-Anzeige hat drei Eingangsleitungen x0, x1 und x2, mit denen die anzuzeigende Ziffer von 0 bis 7 binär codiert wird. x0 ist dabei das niederwertigste Bit.



Die Abbildung zeigt eine Schaltung, die eines der sieben Segmente steuert.



(a) Bestimmen Sie die Wahrheitstafel für die abgebildete Schaltung.

(b) Geben Sie das Segment an, das von der Schaltung gesteuert wird.

(c) Geben Sie ein KV-Diagramm für die boolesche Funktion an, die das Verhalten von Segment s0 beschreibt.

(d) Bestimmen Sie die minimale disjunktive Normalform der booleschen Funktion für das Segment s0.

(e) Überführen Sie die boolesche Funktion aus Aufgabe (d) in ein Schaltbild.

(8 VP)

B3.2 Wir betrachten die Sprache L_1 = { x, axb, aaxbb, aaaxbbb, … }, deren Wörter mit beliebig vielen a beginnen und nach genau einem x die gleiche Anzahl b enthalten.

(a) Geben Sie eine Grammatik für L_1 an.

(b) Begründen Sie, warum es keinen regulären Ausdruck geben kann, der L_1 beschreibt.

(c) Beschreiben Sie, wie ein Kellerautomat arbeitet, der die Sprache L_1 erkennt.

Eine Variante des Kellerautomaten verwendet zwei Keller (Stacks) statt einem.

(d) Begründen Sie, dass es Sprachen gibt, die der Zwei-Keller-Automat erkennen kann, aber der Ein-Keller-Automat nicht.

(7 VP)

Meta 1: Es wird in den zukünftigen Abituraufgaben keine formale Beschreibung und keine Konstruktion eines Kellerautomaten verlangt. Eine grundsätzliche Beschreibung der Arbeitsweise ist ausreichend. (vgl. ZPG-Unterlagen)

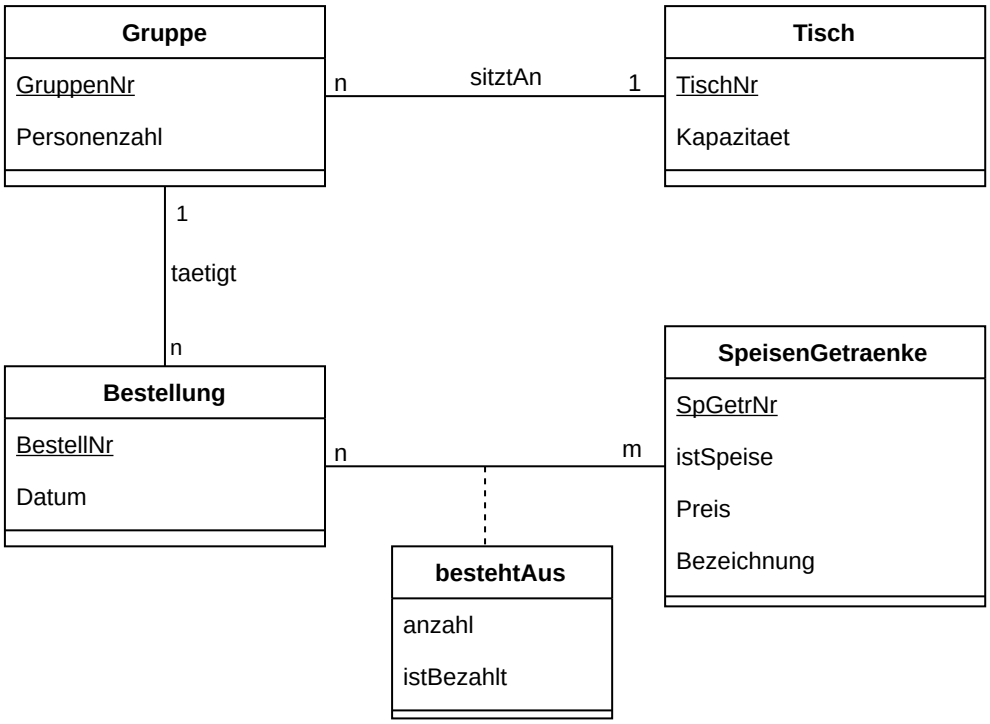
Meta 2: Wahrheitstafeln werden mit den Werten 0 und 1 gefüllt.

Meta 3: AND- und OR-Gatter dürfen mehr als 2 Eingänge haben, auch wenn sie hier nicht verwendet werden.

Meta 4: Für die Negation wird ein vorgeschaltetes NOT-Gatter verwendet. In Schülerlösungen können Negationspunkte am Eingang akzeptiert werden.

B4 Datenbanken / Kryptologie / Datenschutz

B4.1 In einem Restaurant werden die Bestellungen mit Hilfe einer Datenbank verwaltet. Es ergibt sich folgendes UML-Diagramm:



(a) Überführen Sie das UML-Diagramm in ein optimiertes relationales Datenbankschema.

(b) Geben Sie jeweils die passende SQL-Abfrage an:

- Wie viele Speisen stehen auf der Speisekarte?
- Was wurde zum jeweiligen Preis an Tisch 7 bestellt?
- An welchen Tischen wurde jeweils wie viel von dem Getränk mit der Bezeichnung „Cola 0,2 l“ bestellt?

In SQL gibt es den IN-Operator, der in WHERE-Klauseln verwendet werden kann. Er testet, ob ein Wert in einer Liste von anderen Werten vorkommt. Diese Liste kann zum Beispiel durch eine weitere SELECT-Abfrage erzeugt werden.

(c) Geben Sie an, was die folgende SQL-Anweisung leistet:

```
UPDATE bestehtAus SET istBezahlt = TRUE
  WHERE BestellNr
  IN (SELECT BestellNr FROM Bestellung, Gruppe
        WHERE Bestellung.GruppenNr = Gruppe.GruppenNr
              AND Gruppe.TischNr = 23)
```

(9 VP)

Meta 1: Im neuen Abiturformat werden neben ER-Diagrammen auch UML-Diagramme zur Modellierung angegeben werden. In der UML-Darstellung werden sowohl weiterhin Primärschlüssel durch Unterstreichung gekennzeichnet, als auch Assoziationen mit Bezeichnungen und Kardinalitäten versehen. Benötigt die Assoziation eigene Attribute, muss eine eigene Assoziationsklasse verwendet werden.
Die SuS müssen sowohl ER- als auch UML-Diagramme lesen und erweitern können. Bei der Modellierung eigener Diagramme dürfen die SuS wählen, ob sie ER- oder UML-Darstellung verwenden wollen.

Meta 2: Während UPDATE- und DELETE-Befehle (auf einer einzelnen Tabelle) von den SuS auch selbst zu schreiben sind, wird nicht erwartet, INSERT-Befehle selbst zu schreiben, da mit den einfachen Sprachmitteln keine interessanten Fragestellungen möglich sind. Es könnte allerdings vorkommen, dass ein komplexerer INSERT-Befehl zu analysieren ist, z. B.

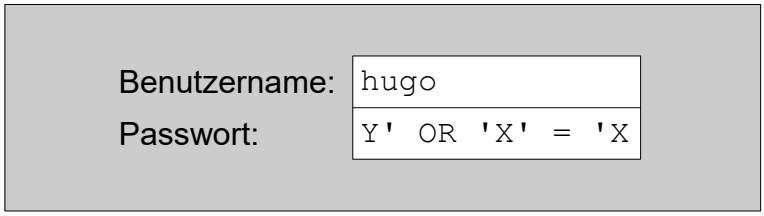
```
INSERT INTO bestehtAus (BestellNr, SpGetrNr, anzahl, istBezahlt)
  SELECT 37, SpGetrNr, 2, FALSE
    FROM SpeisenGetraenke
    WHERE Bezeichnung = 'Pizza Margherita'
```

B4.2 Das Personal im Restaurant muss sich über eine Weboberfläche anmelden, um Bestellungen einzutragen und abzurechnen. Dazu gibt es in der Datenbank eine Tabelle `Benutzer` mit den Spalten `BenutzerNr`, `Name` und `Passwort`. Beim Anmeldevorgang wird auf dem Server die folgende SQL-Abfrage ausgeführt:

```
SELECT BenutzerNr FROM Benutzer
WHERE Name = '{0}' AND Passwort = '{1}'
```

Dabei wird `{0}` durch den auf der Weboberfläche eingegebenen Benutzernamen und `{1}` durch das Passwort ersetzt. Wenn diese Abfrage mindestens einen Datensatz zurückliefert, wird der erste gefundene Benutzer eingeloggt.

(a) Geben Sie den SQL-Befehl an, den der Server nach der Ersetzung von `{0}` und `{1}` bei folgender Eingabe ausführt:



(b) Erläutern Sie, warum es sich bei dieser Eingabe um einen Angriff handelt und welchen Effekt er hat.

Ein Angreifer hat durch einen anderen Angriff eine Liste von Benutzernamen und Passwörtern aus der Datenbank erbeutet. Er stellt aber fest, dass die Passwörter nicht im Klartext, sondern gehasht gespeichert wurden.

(c) Erläutern Sie die wesentliche Eigenschaft einer kryptografischen Hashfunktion für das Speichern von Passwörtern.

Im Internet kursieren Listen mit den 100.000 am häufigsten verwendeten Passwörtern.

(d) Beschreiben Sie, wie er mit Hilfe einer solchen Liste trotz Hashing Rückschlüsse auf die Passwörter ziehen kann.

(6 VP)

Prüfungsfach: Informatik

Musteraufgabensatz 2023ff. **Lösungshinweise** **Blatt 1 von 8**

Für die Fachlehrerin, den Fachlehrer
Die Lösungshinweise stellen nur eine mögliche Aufgabenlösung dar. Andere Lösungsmöglichkeiten sind zuzulassen, wenn sie der Aufgabenstellung entsprechen und sachlich richtig sind. Die Erstkorrektorin / der Erstkorrektor kann in diesem Fall für die Zweitkorrektorin/ den Zweitkorrektor eine Begründung beigeben (anonym und auf einem gesonderten Blatt).
Sofern in den Aufgaben nach Programmcode gefragt ist, kann über kleinere Syntaxfehler hinweg gesehen werden.

Es dürfen für die Teilaufgaben ganze oder halbe Verrechnungspunkte vergeben werden.
Erst die Endsumme aller erteilten Verrechnungspunkte (max. 60 VP) ist ggf. aufzurunden.

Hinweis:
Die Implementierung von Programmen oder Programmteilen erfolgt jeweils in der im Kurs eingeführten Programmiersprache. Bei der Programmierung ist davon auszugehen, dass sämtliche erforderlichen Daten in einem vernünftigen Rahmen eingegeben werden. An eine Fehlerbehandlung von Falscheingaben ist nicht gedacht.

Pflichtaufgabe A:

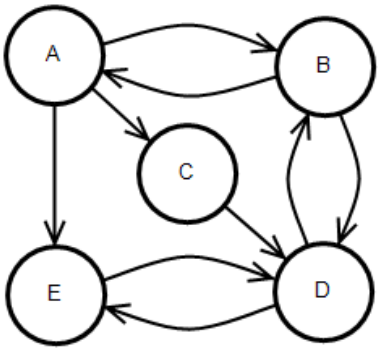
A.1

(a) 13_{10} = 0000 1101_2; Einerkomplement von 0000 1101_2 = 1111 0010_2;
Zweierkomplement von 0000 1101_2 = 1111 0011_2 (1 VP)

(b)
```
   0001 1000
+  1111 0011
 1 111
(1)0000 1011
```
Das Endergebnis bei 8-Bit-Darstellung ist `0000 1011`$_2$. (1 VP)

A.2

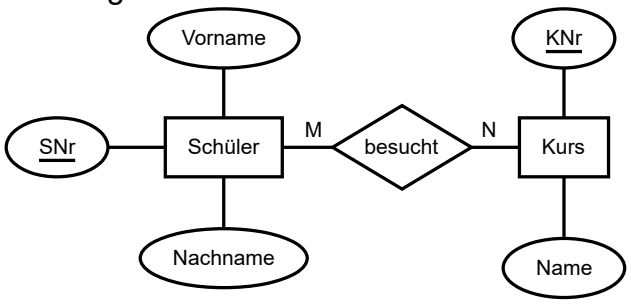
(a) (2 VP)



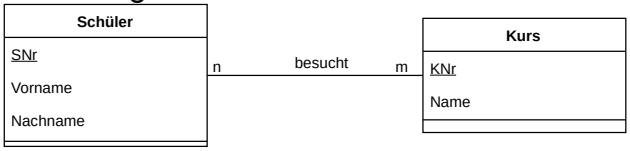
(b) Die Adjazenzmatrix ist nicht symmetrisch, daher muss der Graph gerichtet sein. (1 VP)

A.3 (1,5 VP)

ER-Diagramm:



UML-Diagramm:



A.4 (2 VP)

Ohne Collections und Lambda-Ausdrücke:

```
public List<Integer> filterGerade(List<Integer> e) {
  List<Integer> ergebnis = new ArrayList<Integer>();
  for (int n : e) {
    if (n % 2 == 0) {
      ergebnis.add(n);
    }
  }
  return ergebnis;
}
```

Mit Collections und Lambda-Ausdruck:

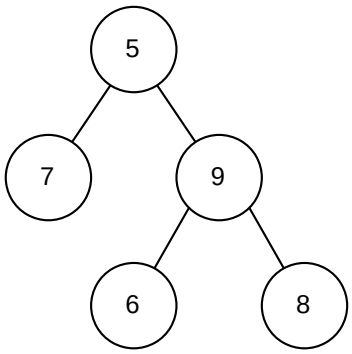
```
public List<Integer> filterGerade(List<Integer> e) {
  return e.stream().filter(x -> x % 2 == 0).collect(Collectors.toList());
}
```

Hinweis: Es ist nicht nötig, die Import-Befehle der Standard-Bibliotheken anzugeben.

A.5

(a) Preorder-Traversierung: 5 – 7 – 9 – 6 – 8
Postorder-Traversierung: 9 – 6 – 7 – 8 – 5 (je 0,5 VP)

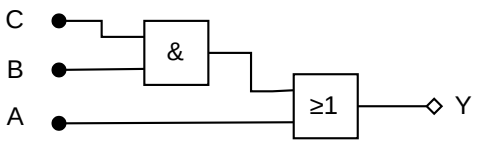
(b) Eine mögliche Lösung: (1 VP)



A.6

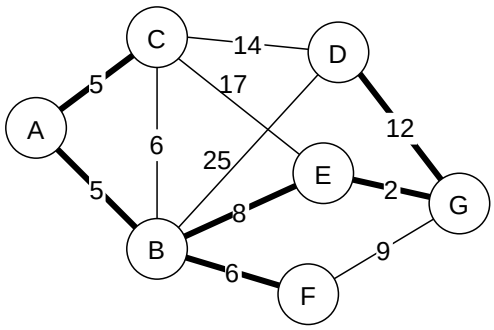
(a) $A \vee \neg(\neg B \wedge C) \wedge C = A \vee (B \vee \neg C) \wedge C = A \vee (B \wedge C) \vee (\neg C \wedge C) = A \vee B \wedge C$ (1,5 VP)

(b) (1 VP)



A.7

(a) (1 VP)



(b) Beispiel: Bau eines Glasfasernetzwerks; in einer Stadt sollen alle bereits bestehenden Verteilerkästen mit Glasfaserleitungen verbunden werden. Da das Verlegen der Glasfaserleitungen sehr teuer ist, sucht man nach den Verbindungen mit den geringsten Gesamtkosten.

Ähnliche Situationen: Planung eines Stromnetzes, Bau einer Pipeline.

Hinweis: Generell kommen minimale spannende Bäume zum Einsatz, wenn es „billig“ ist, die Strecke zu benutzen, aber „teuer“, sie zu bauen. (1 VP)

Wahlaufgabe B1:

B1.1

(a) Implementierung der Methode `bubbleSort(daten: int[])` (4 VP):

```
public void bubbleSort(int[] daten) {
  for (int i = daten.length-1; i > 0; i--) {
    for (int j = 0; j < i; j++) {
      if (daten[j] > daten[j+1]) {
        // Dreieckstausch
        int hilf = daten[j];
        daten[j] = daten[j+1];
        daten[j+1] = hilf;
      }
    }
  }
}
```

(b) Darstellung des Arrayinhalts beim Shaker-Sort-Algorithmus(2 VP):

45	12	99	37	2
12	45	99	37	2
12	45	37	99	2
12	45	37	2	99
12	45	2	37	99
12	2	45	37	99
2	12	45	37	99
2	12	37	45	99

(c) Erläuterung von "worst case" (0,5 VP):

"Worst case" ist die schlechteste oder ungünstigste Ausgangssituation bzw. Belegung des Datenfeldes, die im zu betrachtenden Algorithmus ein Maximum der jeweiligen Ressource wie Laufzeit oder Speicherplatz benötigt.

(d) Begründung für den worst case für { 2, 3, 4, 5, ..., 99, 100, 1 } (1,5 VP)

Bubblesort hat im schlechtesten Fall eine Laufzeit in der Ordnung $O(n^2)$. Dieser tritt auf, wenn der kleinste Wert zu Beginn am Ende des Arrays steht, weil er bei jedem Durchlauf der äußeren Schleife nur um eine Stelle nach vorne rückt. Dieser Fall liegt im gegebenen Array vor.
Shakersort hat das Feld nach zwei Durchläufen vollständig sortiert und bleibt damit weit unter der größtmöglichen Laufzeit.

B1.2

(a) Beschreibung des Algorithmus Mergesort (2 VP):
Das Verfahren beruht auf dem Divide-and-Conquer-Prinzip (*teile und herrsche*). Das zu sortierende Feld wird zunächst in zwei Hälften aufgeteilt (*divide*), die jeweils für sich durch einen rekursiven Aufruf von Mergesort sortiert werden (*conquer*). Dann werden die sortierten Hälften zu einer insgesamt sortierten Folge verschmolzen (*combine*). Dabei werden wiederholt jeweils die vordersten Elemente der Hälften verglichen und das kleinere von beiden in das sortierte Feld übertragen.

(b) Erläuterung der Merge-Methode (3 VP)
Die Methode bekommt die Verweise auf den jeweiligen Anfang der Listen `a` und `b` übergeben. Wenn eine der beiden Listen leer ist, ist die jeweils andere Liste die Ergebnisliste und es wird der Verweis auf deren Anfang zurückgegeben.
Andernfalls werden die Daten der Listenköpfe verglichen. Der kleinere der beiden Listenköpfe wird als neuer Listenkopf des Ergebnisses genommen und die durch den rekursiven Aufruf verschmolzenen restlichen Elemente dahinter gehängt.

(c) Analyse der rekursiven Aufrufe (2 VP)
Rekursive Aufrufe erfolgen nur, solange beide Listen noch Elemente enthalten. Im besten Fall wird deswegen die kürzere Liste in jedem Schritt verkürzt; die Anzahl der Aufrufe ist daher mindestens so groß wie deren Länge.
Im schlechtesten Fall bleiben bis zum Schluss in beiden Listen Elemente übrig, deshalb benötigt man m+n rekursive Aufrufe:
$\min(n,m) \leq$ Anzahl der Aufrufe $\leq m+n$

Wahlaufgabe B2:

B2.1

(a) Implementierung der Klasse Listenknoten<T> (1 VP):

```
public class Listenknoten<T> {
  public T daten;
  public Listenknoten<T> nachfolger;
  public Listenknoten(T daten, Listenknoten<T> nachfolger) {
    this.daten = daten;
    this.nachfolger = nachfolger;
  }
}
```

(b) Implementierung von `enqueue` und `dequeue` (2 VP):

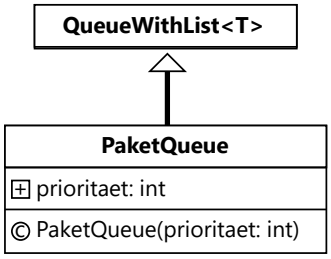
```
public void enqueue(T x) {
  dieListe.anhaengen(x);
}
public T dequeue() {
  T x = dieListe.get(0);
  dieListe.entferneBei(0);
  return x;
}
```

(c) Implementierung von `anhaengen` (2 VP):

```
public void anhaengen(T val) {
  if (anfang == null) {
    anfang = new Listenknoten<T>(val, null);
  } else {
    Listenknoten<T> k = anfang;
    while (k.nachfolger != null) {
      k = k.nachfolger;
    }
    k.nachfolger = new Listenknoten<T>(val, null);
  }
}
```

B2.2

(a) Modellierung als UML-Diagramm (1 VP):



(b) Implementierung der Methode `paketEinfuegen` (3 VP):

```
public void paketEinfuegen(Netzwerkpaket p) {
  PaketQueue pq = null;
  for (int i = 0; i < queues.laenge(); i++) {
    PaketQueue q = queues.get(i);
    if (q.prioritaet == p.prioritaet) {
      pq = q;
    }
  }
  if (pq == null) {
    pq = new PaketQueue(p.prioritaet);
    queues.anhaengen(pq);
  }
  pq.enqueue(p);
}
```

(c) Implementierung der Methode `paketEntnehmen` (3 VP):

```
public Netzwerkpaket paketEntnehmen() {
  PaketQueue max = null;
  for (int i = 0; i < queues.laenge(); i++) {
    PaketQueue q = queues.get(i);
    if (!q.isEmpty() && (max == null || q.prioritaet > max.prioritaet)) {
      max = q;
    }
  }
  if (max == null) {
    return null;
  }
  return max.dequeue();
}
```

Für die Fachlehrerin, den Fachlehrer

B2.3

(a) Bewertung des Vorhabens (3 VP):

Provider: Das Vorhaben verspricht zusätzliche Einnahmen.

Streamingdienste: Es entstehen zusätzliche Kosten, die auf die Kunden umgelegt werden. Nicht alle Dienste können sich diesen Premium-Service leisten, was ein Wettbewerbsnachteil für kleinere Anbieter ist. Wer den Premium-Service in Anspruch nimmt, kann seinen Endkunden bessere Qualität bieten.

Endkunden: Wer einen Streamingdienst bezieht, der den Premium-Service gebucht hat, profitiert von der besseren Qualität. Allerdings müssen sie mit höheren Kosten rechnen. Für Kunden anderer Streamingdienste verschlechtert sich das Angebot, so dass Anreize entstehen, zu den größeren Diensten zu wechseln, was zu einer Monopolbildung führen kann.

Hinweis: Es müssen alle drei Beteiligten genannt werden. Andere schlüssige Argumentationen sind zulässig.

Wahlaufgabe B3:

B3.1

(a) Wahrheitstafel (2 VP):

	x2	x1	x0	s
000	0	0	0	1
001	0	0	1	0
010	0	1	0	1
011	0	1	1	0
100	1	0	0	0
101	1	0	1	0
110	1	1	0	1
111	1	1	1	0

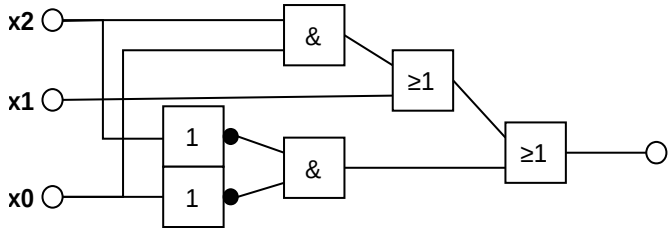
(b) Auswahl des Segments (1 VP):
Es handelt sich um Segment s4.

(c) KV-Diagramm für die boolesche Funktion von s0 (1,5 VP):

	x1	x1	$\overline{x1}$	$\overline{x1}$
x0	1	1	0	1
$\overline{x0}$	1	1	1	0
	x2	$\overline{x2}$	$\overline{x2}$	x2

(d) Disjunktive Normalform (2 VP):
$x1 \vee (\neg x2 \wedge \neg x0) \vee (x0 \wedge x2)$

(e) Schaltbild (1,5 VP):



B3.2

(a) Grammatik für L_1 (2 VP):
$G_1 = \{ \Sigma, V, S, P \}$ mit
$\Sigma = \{ a, b, x \}$
$V = \{ S \}$
Startsymbol S
$P = \{ S \rightarrow aSb \mid x \}$

(b) Begründung, warum es keinen regulären Ausdruck geben kann (1,5 VP):
Mit regulären Ausdrücken können genau die regulären Sprachen beschrieben werden. Da aber hier unbegrenzt viele a gezählt und mit den b verglichen werden müssen, kann es keinen erkennenden Automat mit endlich vielen Zuständen geben, der diese Sprache erkennt. Sie ist damit nicht regulär.

(c) Beschreibung des Kellerautomaten (2 VP):
Der Kellerautomat würde die Eingabe von vorne nach hinten verarbeiten. Immer wenn ein a kommt, wird ein Symbol dafür im Keller abgelegt. Ein b darf zu diesem Zeitpunkt noch nicht vorkommen. Mit dem x geht er in einen neuen Zustand über. In diesem Zustand dürfen nur noch b in der Eingabe vorkommen. Mit jedem b wird ein Symbol aus dem Keller entfernt. Ist der Keller genau am Ende des Eingabewortes leer, akzeptiert der Kellerautomat das Wort.

(d) Begründung, dass der Zwei-Keller-Automat mächtiger ist (1,5 VP)
Der Zwei-Keller-Automat kann z.B. die Sprache $L = \{ abc, aabbcc, aaabbbccc, \ldots \} = \{ a^n b^n c^n \mid n > 0 \}$ erkennen. Dieser würde beim Lesen der a Symbole auf den ersten Stack legen. Beim Lesen des ersten b würde er den Zustand wechseln und für jedes gelesene b ein Symbol vom ersten Stack nehmen und eines auf den zweiten Stack legen. Beim Lesen des ersten c würde er Symbole vom zweiten Stack nehmen.
Alternativ: Der Zwei-Keller-Automat ist äquivalent zur Turing-Maschine. Diese kann mehr Sprachen erkennen als der Kellerautomat.

Wahlaufgabe B4:

B4.1

(a) Überführung in ein optimiertes relationales Datenbankschema (2 VP):

Tisch:	TischNr, Kapazitaet
Gruppe:	GruppenNr, Personenzahl, TischNr ↑
Bestellung:	BestellNr, Datum, GruppenNr ↑
SpeisenGetraenke:	SpGetrNr ↑, istSpeise, Preis, Bezeichnung
bestehtAus	BestellNr ↑, SpGetrNr ↑, anzahl, istBezahlt

(b) Select-Befehle (1 VP + 2 VP + 2VP):

```
SELECT COUNT(*)
FROM SpeisenGetraenke
WHERE istSpeise = TRUE

SELECT Bezeichnung, Preis
FROM Tisch, Gruppe, Bestellung, SpeisGetr, bestehtAus
WHERE TischNr=7
AND Tisch.TischNr = Gruppe.TischNr
AND Gruppe.GruppenNR = Bestellung.GruppenNR
AND Bestellung.BestellNR = bestehtAus.BestellNR
AND bestehtAus.spGetrNr = SpeisenGetraenke.spGetrNr
```

```
SELECT Tisch.TischNR, COUNT(*)
FROM Tisch, Gruppe, Bestellung, bestehtAus, SpeisenGetraenke
WHERE Tisch.TischNr = Gruppe.TischNR
AND Bestellung.GruppenNR = Gruppe.GruppenNR
AND Bestellung.bestehtAus = SpeisenGetraenke.SpGetNr
AND SpeisenGetraenke.Bezeichnung = "Cola 0,2l"
GROUP BY Tisch.TischNR
```

(c) Angabe der SQL-Anweisung (2 VP):
Diese SQL-Anweisung markiert alle Bestellungen von Tisch 23 als bezahlt.

B4.2

(a) Ausgeführter SQL-Befehl (1 VP):

```
SELECT BenutzerNr FROM Benutzer
WHERE Name = 'hugo' AND Passwort = 'Y' OR 'X' = 'X'
```

(b) Erläuterung des Angriffs (2 VP):
Die Abfrage liefert alle Benutzernummern. Der Term `'X' = 'X'` wird zu `TRUE` ausgewertet, damit ist die gesamte `WHERE`-Bedingung für jeden Datensatz `TRUE`, da `AND` stärker als `OR` bindet.

Es handelt sich um einen Angriff, weil das System bei dieser Eingabe anders arbeitet als vom Entwickler bzw. Betreiber beabsichtigt, konkret weil der Schutzzweck des Einloggens mit Passwort hier unterlaufen wird.

(c) Erläuterung einer Hashfunktion (1 VP):
Die Hashfunktion für die Speicherung von Passwörtern darf in erster Linie nicht umkehrbar sein: Aus dem Hashwert darf niemand Rückschlüsse auf das gehashte Passwort ziehen können.

(d) Beschreibung, wie auf Passwörter geschlossen werden kann (2 VP):
Der Angreifer vergleicht alle Hashwerte der erbeuteten Passwortdatenbank mit allen Hashwerten der öffentlich verfügbaren Liste. Jede Übereinstimmung der Hashwerte liefert ihm ein Passwort aus der Datenbank.

A Pflichtteil

A1 Erläutern Sie das Grundprinzip der Komprimierung beim LZW-Verfahren.

(2 VP)

A2 Begründen Sie, dass das LZW-Verfahren nicht jede Eingabe komprimieren kann.

(2 VP)

A3 Implementieren Sie eine Methode, die alle durch 17 teilbaren Zahlen zwischen 10.000 und 100.000 ausgibt. Sie können für die Ausgabe eine Methode `ausgeben(n: int)` verwenden.

(1 VP)

A4 Implementieren Sie eine Methode `caesar1(klartext: String): String`, die eine Cäsar-Verschlüsselung mit dem Schlüssel 1 durchführt und den Geheimtext zurückgibt.
Beispiel: Aus ZAUN wird ABVO.
Sie können davon ausgehen, dass der Klartext nur Großbuchstaben zwischen A und Z enthält.

(2 VP)

A5 Erläutern Sie den Begriff „Polymorphie“.

(1 VP)

A6 Vergleichen Sie symmetrische und asymmetrische Verschlüsselung. Gehen Sie insbesondere auch auf den Schlüsselaustausch ein.

(2 VP)

A7 Implementieren Sie die Fakultätsfunktion $f(n) = n \cdot (n-1) \cdot \ldots \cdot 3 \cdot 2 \cdot 1$ rekursiv.

(1 VP)

A8 Beschreiben Sie die grundlegenden Eigenschaften und Operationen des ADTs Stack.

(1 VP)

A9

(a) Zeichnen Sie das Schaltbild eines Halbaddierers.

(b) Zeichnen Sie das Schaltbild eines Volladdierers.

(2 VP)

A10 Führen Sie das Vigenère-Verfahren mit dem Klartext NASHORN und dem Schlüsselwort RABE durch.

(1 VP)

A11 Beschreiben Sie ein Schwarz-Weiß-Bild, das gut mit Lauflängencodierung komprimiert werden kann und eines, das schlecht mit Lauflängencodierung komprimiert werden kann.

(2 VP)

A12

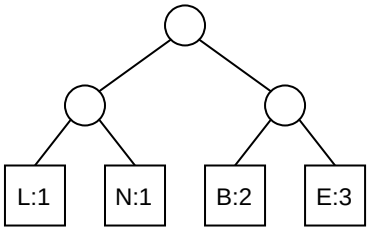
(a) Überführen Sie die Festkommazahl $101{,}01_2$ in eine Dezimalzahl.

(b) Geben Sie an, welche der folgenden Zahlen sich exakt als Festkommazahl darstellen lassen:

- 1,75
- 0,3
- $\frac{17}{64}$

(2 VP)

A13 Gegeben ist der folgende Baum:



(a) Begründen Sie, warum der Baum kein gültiger Huffman-Baum ist.

(b) Erläutern Sie den Nachteil, der beim Codieren des Wortes „BELEBEN“ mit diesem Baum auftritt.

(c) Geben Sie einen korrekten Huffman-Baum für die angegebenen Buchstabenhäufigkeiten an.

(3 VP)

A14 Ein Palindrom ist ein Wort, das von vorne und von hinten gelesen gleich ist (z .B. ANNA, RENTNER, LAGERREGAL).
Implementieren Sie eine Methode `istPalindrom(eingabe: String): boolean`, die `true` zurückgibt, wenn es sich bei `eingabe` um ein Palindrom handelt, sonst `false`. Sie können davon ausgehen, dass die Eingabe nur aus Großbuchstaben besteht.

(2 VP)

A15 Gegeben ist ein KV-Diagramm für eine logische Funktion f:
Geben Sie die minimale disjunktive Normalform für die Funktion f an.

f		x0 x1			
		00	**01**	**11**	**10**
x2	**0**	1	1	1	1
	1	0	0	1	0

(2 VP)

A16 Minimieren Sie den folgenden boolschen Term mithilfe eines KV-Diagramms und vereinfachen Sie, wenn möglich, das Ergebnis:
X = $(A \wedge \neg B \wedge C \wedge D) \vee (A \wedge \neg B \wedge \neg C \wedge D) \vee (A \wedge \neg B \wedge C \wedge \neg D) \vee$
$(A \wedge \neg B \wedge \neg C \wedge \neg D) \vee (\neg A \wedge \neg B \wedge \neg C \wedge \neg D) \vee (\neg A \wedge \neg B \wedge C \wedge \neg D)$

(2,5 VP)

A17 Gegeben ist der reguläre Ausdruck **(b(ba)*)***
Geben Sie zu jedem der folgenden Wörter an, ob es zu ihm passt:

- bbabab
- bab
- Bbbab

(1,5 VP)

A18 Gegeben ist ein endlicher Automat M = (Σ, Z, S, E, δ)
mit Σ = {a, b}, Z = {qo, q1, q2, q3}, S = qo, E = {q2} und

δ	a	b
q0	q1	-
q1	-	q2
q2	q1	-

(a) Zeichnen Sie das zugehörige Zustandsübergangsdiagramm.

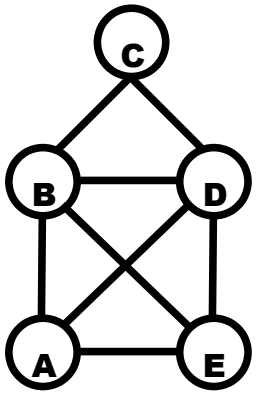
(b) Überprüfen Sie, ob die folgenden Eingaben von M akzeptiert werden.

- `bbaabb`
- `ababab`

(c) Beschreiben Sie die Sprache L(M) durch einen regulären Ausdruck.

(3 VP)

A19 Gegeben ist der nebenstehende Graph.



(a) Geben Sie den Breitensuchbaum an, der bei einer Breitensuche mit dem Startpunkt C entsteht.
Lexikographisch kleinere Buchstaben sollen zuerst bearbeitet werden.

Ein Eulerkreis bezeichnet einen Weg durch einen Graphen, der jede Kante exakt einmal durchläuft und zum Startknoten zurückkehrt. Fällt die Bedingung Startknoten = Endknoten weg, spricht man von einem Eulerweg.

(b) Geben Sie einen Eulerweg durch den angegebenen Graphen an.

(c) Begründen Sie, dass es durch diesen Graphen keinen Eulerkreis geben kann.

(3 VP)

A20 Zum Sortieren eines Arrays A der Länge n sei der folgende Algorithmus gegeben

```
wiederhole für i von 1 bis n
     wiederhole für j von 1 bis n – 1
          wenn (A[j-1] > A[j]) {
               vertausche(A[j-1],A[j])
          }
     }
}
```

(a) Geben Sie die Laufzeit des Algorithmus in O(n)-Schreibweise an und begründen Sie Ihre Angabe.

Wird innerhalb einer Iteration keine Vertauschung durchgeführt, so finden auch in den restlichen Iterationen keine mehr statt. Der äußere Schleifendurchlauf könnte also direkt beendet werden.

(b) Ergänzen Sie den Algorithmus so, dass diese Eigenschaft ausgenutzt und somit die Laufzeit verbessert wird,

(2 VP)

Meta: Wenn hier Standardalgorithmen vermisst werden, sollte man sich klar machen, dass diese im Bildungsplan nur aufzählend mit dem Zusatz „zum Beispiel“ erwähnt werden, z. B. Prim, Kruskal, Dijkstra, Bellman-Ford, Greedy, Türme von Hanoi, Fibonacci-Zahlen, usw.. Da daher nicht davon auszugehen ist, dass alle diese Algorithmen im Unterricht behandelt wurden, müsste der jeweilige Algorithmus im Aufgabentext explizit beschrieben werden.

B1 Entwurf / Analyse und Programmiertechniken

B1.1 Die Fibonacci-Funktion (auch als Fibonacci-Folge bezeichnet) ist definiert als:

$$f(n)=\begin{cases} 0 & \text{wenn } n = 0 \\ 1 & \text{wenn } n = 1 \\ f(n-1)+f(n-2) & \text{sonst} \end{cases}$$

Die ersten Werte der Fibonacci-Funktion sind 0, 1, 1, 2, 3, 5, 8, 13, …
Die Definition kann einfach in eine rekursive Methode übersetzt werden:

```
fib(n: int): int
  falls n ≤ 2:
    gib n zurück
  sonst:
    gib fib(n-1) + fib(n-2) zurück
```

(a) Geben Sie die vier Werte der Fibonacci-Folge an, die auf das Folgenglied 13 folgen.

(b) Stellen Sie die ausgeführten Methodenaufrufe bei der Ausführung von `fib(4)` als Baum dar.

(c) Begründen Sie, warum die Anzahl der Methodenaufrufe für `fib(n)` weniger als 2^{n+1} beträgt.

(d) Implementieren Sie eine Methode `fibIterativ(n: int): int`, die dasselbe Ergebnis wie `fib` berechnet, die aber ohne Rekursion arbeitet.

Bei manchen aufwendigen Problemen kann man Zeit sparen, indem man geeignete Zwischenergebnisse abspeichert und später wiederverwendet. Diese Technik wird als *dynamische Programmierung* oder *Memoisation* bezeichnet:

```
int[] cache;

fibDyn(n: int): int
  falls n ≤ 1:
    gib n zurück
  cache = int-Array der Länge n+1
  cache[0] = 0;
  cache[1] = 1;
  for i = 2 bis n:
    cache[i] = -1;
  gib lookup(n) zurück

lookup(n: int): int
  falls cache[n] < 0:
    cache[n] = lookup(n-1) + lookup(n-2);
  gib cache[n] zurück
```

(e) Die Methode `fibDyn(4)` wird aufgerufen. Stellen Sie die ausgeführten Methodenaufrufe als Baum dar dar. Geben Sie bei jedem Knoten den Inhalt des Arrays `cache` an, nachdem der jeweilige Methodenaufruf beendet ist.

(f) Begründen Sie, warum die dynamische Implementierung effizienter ist als die erste.

(12 VP)

Meta: Die B-Aufgaben im Fundusteil sind nicht alle vollständig und würden in der Abiturprüfung auf 15 Punkte ergänzt.

B2 Algorithmen und Datenstrukturen – Variante 1

B2.1 Gegeben ist ein Graph in Form einer Adjazenzliste.

Knoten	Nachbarn
A	B, C
B	A, C, E, F
C	A, B, D, E
D	C, G
E	B, C, F, G
F	B, E, G
G	D, E, F

(a) Überführen Sie die Adjazenzliste in seine grafische Darstellung.

(b) Geben Sie für jeden Knoten des Graphen die Länge des kürzesten Pfades von Knoten A an.

(c) Beschreiben Sie, wie man den Breitensuche-Algorithmus verwenden kann, so dass man für ungewichtete Graphen den Abstand sowie den jeweils kürzesten Pfad von einem Startknoten aus erhält.

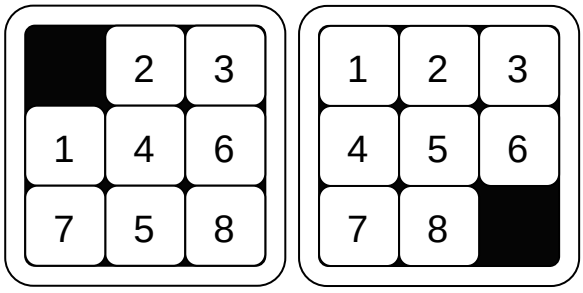
(d) Begründen Sie, warum der Algorithmus modifiziert werden muss, um die gleiche Aufgabe für gewichteten Graphen lösen zu können. Beschreiben Sie eine Modifikation.

(7 VP)

B2.2 In einem bekannten Geduldsspiel muss der Spieler die Puzzlestücke so verschieben, dass die Ziffern am Schluss wie im Bild gezeigt sortiert sind. Ein Zug besteht daraus, dass ein Spielstein auf die benachbarte Lücke verschoben wird.

Eine Spielstellung wird als `int`-Array mit 9 Elementen repräsentiert. Jede Zahl darin steht für einen Stein, wobei die Zahl 0 für die Lücke verwendet wird.
Die Ausgangsstellung im Bild wird also durch das Array `{ 0, 2, 3, 1, 4, 6, 7, 5, 8 }` dargestellt.

(a) Geben Sie die von der Stellung `{ 2, 4, 3, 0, 1, 6, 7, 5, 8 }` in einem Zug erreichbaren Stellungen als Array an.



Ausgangsstellung *Zielstellung*

Die Stellungen des Schiebepuzzles lassen sich als Knoten eines Graphen modellieren. Zwischen zwei Knoten besteht eine Kante, wenn die Stellungen durch einen Zug auseinander entstehen können. Um zu ermitteln, ob eine Kante vorhanden ist, wird die Methode `existiertKante(a: int[], b: int[]): boolean` und die Hilfsmethode `istNachbarfeld(i: int, j: int): boolean` benötigt.

Die Methode `istNachbarfeld(i: int, j: int): boolean` erhält zwei Indizes im Array (Zahlen von 0 bis 8) und gibt `true` zurück, wenn sie für benachbarte Positionen im Spielfeld stehen, sonst `false`.

(b) Implementieren Sie die Methode
`istNachbarfeld(i: int, j: int): boolean`.

Die Methode `existiertKante(a: int[], b: int[]): boolean` erhält die Repräsentation zweier Spielstellungen und gibt `true` zurück, wenn diese im Graph durch eine Kante verbunden sind, sonst `false`.

(c) Implementieren Sie die Methode
`existiertKante(a: int[], b: int[]): boolean` unter Verwendung der Methode `istNachbarfeld`.

Auf diesen Graphen kann man einen Algorithmus zur Bestimmung des kürzesten Weges anwenden. Dabei kann man die Suche entweder bei der Ausgangsstellung oder bei der Zielstellung beginnen lassen.

(d) Erläutern Sie jeweils, was die so ermittelten Pfade und ihre Längen im Kontext des Schiebepuzzles bedeuten.

(8 VP)

Es gibt Varianten des Schiebepuzzles, bei denen die Spielsteine herausgenommen werden und in einer beliebigen Anordnung wieder eingesetzt werden können. Eine Stellung gilt als lösbar, wenn man durch Verschieben die Zielstellung erreichen kann.

(e) Beschreiben Sie einen Algorithmus, der für ein solches Puzzle feststellt, ob es auch in unlösbare Stellungen gebracht werden kann.

(4 VP)

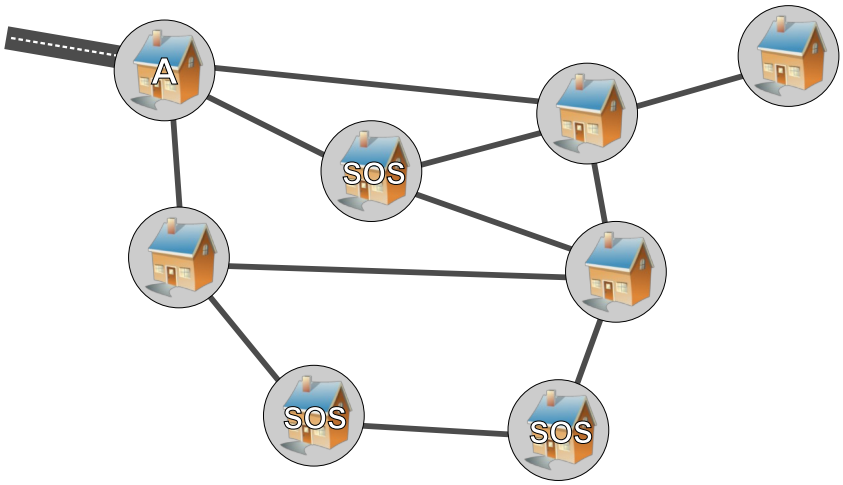
B2 Algorithmen und Datenstrukturen – Variante 2

B2.1 Einige Dörfer in den Bergen sind über ein Straßennetz zu erreichen. Eine Hauptstraße führt zum ersten Dorf des Tales. Nach einem Unwetter sind einige Dörfer aber nicht mehr erreichbar und melden SOS.
Daraus lässt sich auf den Zustand der Straßen schließen. Es gibt drei Möglichkeiten:

- 1 = Die Straße ist nicht befahrbar.
- 2 = Die Straße ist befahrbar.
- 3 = Ohne weitere Information ist nicht entscheidbar, ob sie befahrbar oder nicht befahrbar ist.

Die Hauptstraße ist immer befahrbar.

Unten ist ein Straßennetz abgebildet. Die Hauptstraße führt zu Dorf A ganz links oben.



(a) Geben Sie für jede andere Straße ihren Zustand an.

Mit Hilfe des folgenden Algorithmus kann man den Zustand der Straßen automatisch ermitteln:

1. Setze alle Straßen auf „nicht entscheidbar“.
2. Kennzeichne alle Straßen von einem unerreichbaren Dorf zu einem erreichbaren Dorf als "nicht befahrbar".
3. Für jede Straße S zwischen zwei erreichbaren Dörfern:
   a) Markiere S als "nicht befahrbar".
   b) Bestimme die Anzahl der Dörfer, die von Dorf A aus über "befahrbare" oder "nicht entscheidbare" Straßen erreichbar sind (inklusive A selbst).
   c) Stimmt die Anzahl mit der Anzahl der Dörfer ohne "SOS" überein, setze S auf „nicht entscheidbar“. Andernfalls setze S auf "befahrbar".

(b) Begründen Sie, warum eine Straße zwischen einem erreichbaren und einem nicht erreichbaren Dorf auf jeden Fall auf „nicht befahrbar“ gesetzt werden muss.

(c) Erläutern Sie die Breitensuche und wie sie in Schritt 3.b) zur Bestimmung der erreichbaren Dörfer eingesetzt werden kann.

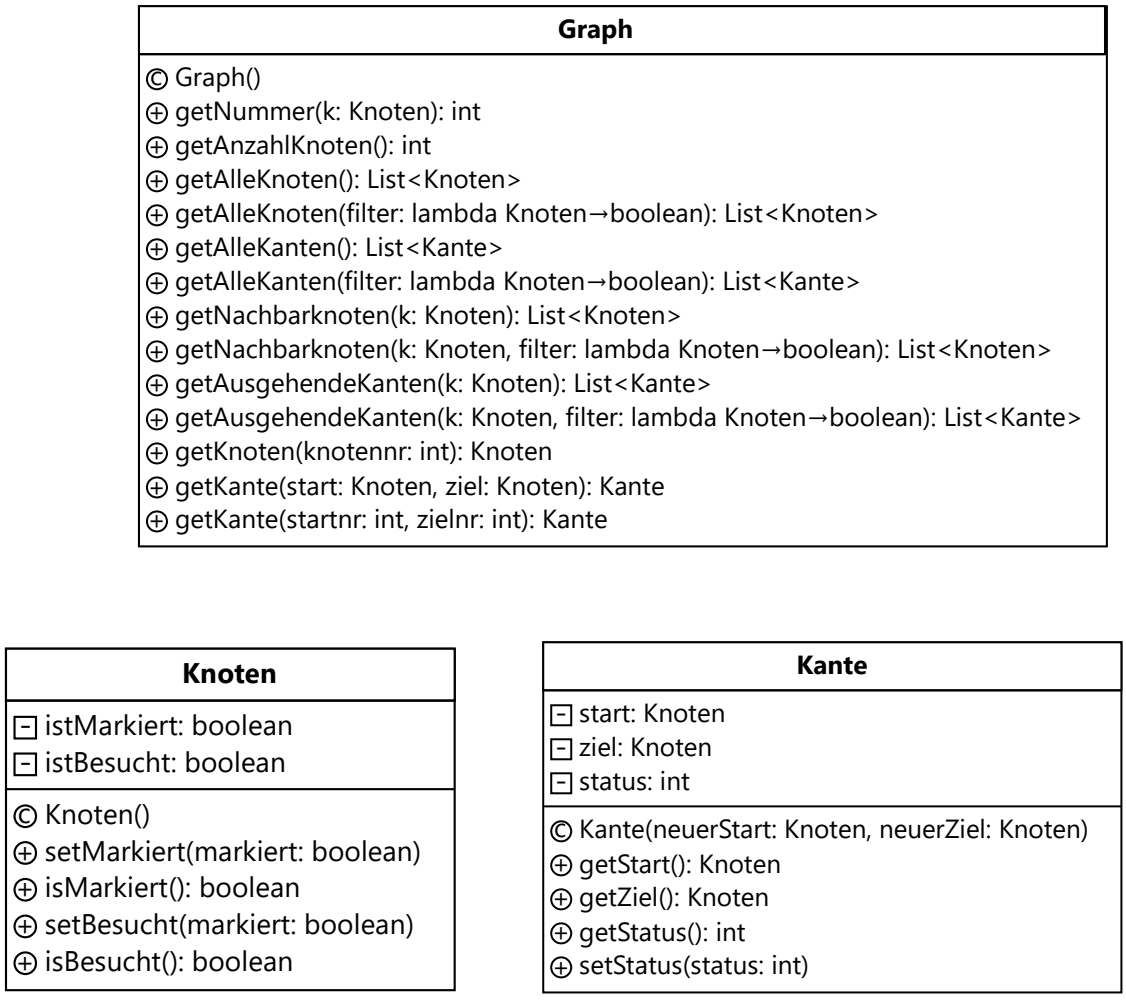
(d) Implementieren Sie die Methode `bestimmeStrassenzustaende(g: Graph, startknoten: Knoten)`, die den Straßenzustand aller Straßen als Status der Kanten im Graphen speichert. Knoten, die Städte mit "SOS" repräsentieren, sind markiert. Sie können davon ausgehen, dass die Methode `bestimmeAnzahlErreichbare(startknoten: Knoten): int`, die die Anzahl der erreichbaren Städte wie in Schritt 3.b) beschrieben bestimmt, schon fertig implementiert ist.

(11 VP)

Meta 1: Die in derartigen Aufgaben verwendeten Klassen entsprechen den Klassen aus den Materialien der ZPG Kursstufe. Dabei werden in den Abituraufgaben nur die benötigten Methoden und Attribute dargestellt.

Meta 2: Lambda-Ausdrücke sind bei derartigen Algorithmen nicht erforderlich, können die Implementierung aber erleichtern. Daher gibt es die entsprechenden Methoden in zwei Formen (mit und ohne Lambda-Ausdruck als Parameter). Parameter, die Lambda-Ausdrücke erfordern, werden in der oben dargestellten Form "lambda Parametertyp → Ergebnistyp" angeben.

Anlage Klassendiagramm der Klassen Graph, Knoten, Kante:



B3 Formale Sprachen / Automaten / Technische Informatik

B3.1 Ganze Zahlen lassen sich in verschiedenen Zahlensystemen darstellen. Außer dem bekannten Dezimal- bzw Binärsystem (die die Basen 10 bzw. 2 verwenden) und dem Hexadezimalsystem (mit der Basis 16) wird in der Informatik auch das Oktalsystem eingesetzt, das die Basis 8 verwendet:

Ganze Zahl in ...		
… Dezimaldarstellung	… Binärdarstellung	… Oktaldarstellung
0	$0_{(2)}$	$0_{(8)}$
1	$1_{(2)}$	$1_{(8)}$
2	$10_{(2)}$	$2_{(8)}$
…	…	…
7	$111_{(2)}$	$7_{(8)}$
8	$1000_{(2)}$	$10_{(8)}$
9	$1001_{(2)}$	$11_{(8)}$
10	$1010_{(2)}$	$12_{(8)}$
11	$1011_{(2)}$	$13_{(8)}$
…	…	…
20	$10100_{(2)}$	$24_{(8)}$

(a) Begründen Sie, warum sich das Oktalsystem immer dann besonders gut eignet, wenn in einer längeren Bitfolge jeweils drei Bit eine Gruppe bilden, die gemeinsam dargestellt werden soll.

(b) Geben Sie die Oktaldarstellung der Binärzahl $101100111_{(2)}$ an.

(c) Geben Sie den Übergangsgraphen eines Mealy-Automaten mit dem Eingabealphabet $\Sigma = \{ 0, 1 \}$ an, der zu einer Bitfolge der Länge 3 die entsprechende Oktalziffer ausgibt. Verwenden Sie den Platzhalter _ für Übergänge, an denen der Automat noch keine Oktalziffer ausgeben soll.

(d) Erweitern Sie diesen Automaten so, dass er beliebige Bitfolgen mit einer durch 3 teilbaren Länge ins Oktalsystem umwandelt.

(e) Geben Sie an, welche Ausgabe der Mealy-Automat erzeugt, wenn ihm die Bitfolge 10101110 übergeben wird. Erläutern Sie allgemein, warum es keinen Mealy-Automat geben kann, der Bitfolgen beliebiger Länge in Oktalzahlen umwandelt.

(9,5 VP)

Meta: Als Kontext für Mealy-Automaten sind auch Szenarien im Stil von Kara
https://www.swisseduc.ch/informatik/karatojava/kara/index.html *denkbar.*

B4 Datenbanken / Kryptologie / Datenschutz

B1.1 Bei einem Bewertungsportal einer Stadtbibliothek kann man Kinder- und Jugendbücher sowie Schullektüren mit Punkten von 0 (schlecht) bis 5 (sehr gut) bewerten.
Das System nutzt folgende Tabellen:

Leser

LID	**Vorname**	**Name**
123	Tim	Jones
273	Max	Schmid
655	Walter	Müller
...	...	...

Bewertung

LID	**BID**	**Punkte**
123	100	4
123	101	5
273	102	5
273	103	2
655	101	5
273	106	5
273	105	4
655	104	5
123	105	3

Buecher

BID	**Titel**	**Autor**	**Verlag**	**Jahr**	**Seiten**	**Preis**
100	Percy Jackson – Diebe im Olymp	Rick Riordan	Carlsen	2011	448	8,99
101	Der Räuber Hotzenplotz	Otfried Preußler	Carlsen	2015	128	5,99
102	Neues vom kleinen Nick	René Goscinny; J.-J. Sempé	Diogenes	2005	648	24,90
103	Das Parfüm	Patrick Süskind	Diogenes	1994	336	12,00
104	Homo faber	Max Frisch	Suhrkamp	1977	208	8,00
105	Krabat	Otfried Preußler	dtv	1971	352	8,95
106	Percy Jackson – Im Bann des Zyklopen	Rick Riordan	Carlsen	2011	336	7,99
...	...	...	...	...	...	...

Alle Fragen beziehen sich auf die in der obigen Datenbank gespeicherten Daten.

(a) Geben Sie jeweils die passende SQL-Abfrage an:

- Wie viele Bücher hat jeder einzelne Verlag im Zeitraum von 1990 bis 2016 veröffentlicht?
- Gesucht ist eine Liste der Lieblingsautoren jedes Lesers.
  Dazu soll für jeden Leser die durchschnittliche Bewertung jedes Autors angezeigt werden, aufsteigend nach dem Nachnamen und Vornamen des Lesers und absteigend nach der durchschnittlichen Bewertung pro Autor.

Die folgende SQL-Abfrage soll die durchschnittliche Bewertung für „Das Parfüm“ ermitteln:

```
SELECT AVG(Punkte) FROM Buecher, Bewertung
WHERE Titel = 'Das Parfüm'
```

Sie liefert das falsche Ergebnis 4,22222.

(b) Begründen Sie, warum es zu diesem falschen Ergebnis kommt.

(c) Ergänzen Sie die SQL-Abfrage so, dass sie das gewollte Ergebnis liefert.

Ein Verlag interessiert sich für die Datenbank des Bewertungsportals.

(d) Geben Sie zwei Beispiele für Informationen an, die der Verlag der Datenbank entnehmen kann, um sie für seine Werbestrategie zu nutzen.

(e) Begründen Sie, warum die Stadtbibliothek die Datenbank nicht herausgeben darf.

(9 VP)

B1.2 Eine Rezeptdatenbank sammelt Rezepte und stellt sie über das Internet zur Verfügung. Ein Rezept speichert, wie man ein Gericht zubereitet, für wie viele Personen es gedacht ist und wie lange die Zubereitung dauert.
Für ein Gericht benötigt man eine oder mehrere Zutaten. Zutaten können sich im Brennwert (angegeben in kcal pro 100 g) unterscheiden. Von jeder Zutat braucht man eine bestimmte Menge.
Die Zubereitung des Gerichts wird mit Fotos illustriert. Jedes Foto hat eine Beschriftung.

(a) Entwerfen Sie ein passendes ER-Modell oder UML-Modell für diese Rezeptdatenbank, markieren Sie dort die Primärschlüssel und bestimmen Sie die Kardinalitäten.

Omas geheimes Plätzchenrezept soll auch in die Datenbank übernommen werden. Aus Sicherheitsgründen hat sie das Rezept verschlüsselt (siehe unten). Dabei hat sie ein Verfahren mit monoalphabetischer Substitution verwendet.

(b) Erläutern Sie, wie Sie mit diesem Vorwissen eine Kryptoanalyse des folgenden Textes durchführen würden.

```
Bjkpktu:                  Bjltotnkjuc:
- 1000 c Vthw             Pwwtm bjmpvvtuoühotu, cjk
- 500 c Ljkkto            ejoahxutktu jue epuu pjz otnahwnah
- 6 Tnto                  Vthw pjmoswwtu. Pumahwntßtue Zsovtu
- 400 c Bjaxto            pjmmktahtu jue pjz tnu vnk
- 2 Rax. Lpaxrjwito       Lpaxrprnto pjmctwtcktm Lwtah wtctu.
- 2 Rax. Ipunwwtbjaxto    Pumahwntßtue otnu nu etu Sztu. Ltn
- 250 c Vpuetwu           ap. 180 °A Slto- jue Juktohnkbt ap.
                          10 - 12 Vnu. yt upah Lpaxsztu
                          lpaxtu.
                          Upah etv Lpaxtu xöuutu ent
                          Rwäkbahtu yt upah Ctmahvpax
                          cpountok gtoetu.
```

(6 VP)

Meta: Bei dieser Aufgabe handelt es sich um eine auf 15 VP gekürzte Nachterminaufgabe.

Prüfungsfach: Informatik

Fundusaufgaben 2023ff. Lösungshinweise

Für die Fachlehrerin, den Fachlehrer

Pflichtaufgabe A:

A.1 Das LZW-Verfahren ist ein Wörterbuchverfahren. Dabei wird beim Komprimieren eine Tabelle aus sich wiederholenden Zeichenfolgen aufgebaut, die nur noch durch ihre Nummer in der Tabelle dargestellt werden muss, wodurch man Speicherplatz spart. (2 VP)

A.2 Das LZW-Verfahren ist ein verlustfreies Verfahren, d.h. jede Eingabe ist eindeutig wiederherstellbar. Zu jeder komprimierten Bitfolge gehört damit genau eine Eingabe. Es kann kein verlustfreies Verfahren geben, das jede Eingabe komprimiert.
Begründung 1: Wenn es ein Verfahren gäbe, das jede Eingabe verkürzen kann, könnte man dieses wiederholt anwenden, bis die Ausgabe nur noch 1 Bit lang wäre. Diese könnte genau zwei Werte annehmen, 0 oder 1. Daraus könnte man aber höchstens zwei Eingaben rekonstruieren.
Begründung 2: Angenommen, es gäbe ein Verfahren, das jede Eingabe der Länge n um mindestens 1 Bit verkürzen könnte. Es gibt 2^n mögliche Eingaben der Länge n. Da die Ausgabe höchstens n-1 Bit lang ist, gibt es dafür $1 + 2 + 4 + 8 + \ldots + 2^{n-1} = 2^n - 1$ Möglichkeiten. Nach dem Schubfachprinzip muss es dann mindestens eine Ausgabe geben, die aus zwei verschiedenen Eingaben entstanden ist. Diese Ausgabe ist damit nicht mehr eindeutig rekonstruierbar. (2 VP)

A.3
```
public void teilbar() {
  for (int i = 10000; i <= 100000; i++) {
    if (i % 17 == 0) {
      ausgeben(i);
    }
  }
}
```
(1 VP)

A.4
```
public String caesar1(String klartext) {
  String geheim = "";
  for (char c : klartext.toCharArray()) {
    c++;
    if (c > 'Z') {
      c = 'A';
    }
    geheim += c;
  }
  return geheim;
}
```
(2 VP)

A.5 Methoden einer Klasse werden manchmal in Unterklassen überschrieben. Polymorphie bedeutet, dass diejenige Implementation gewählt wird, die der tatsächlichen Klasse des Objekts entspricht. (1 VP)

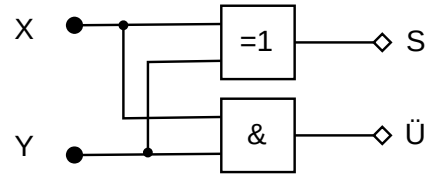
A.6 Symmetrische Verschlüsselungsverfahren verwenden den gleichen Schlüssel für Ver- und Entschlüsselung. Asymmetrische Verfahren verwenden zwei Schlüssel, einen öffentlichen zum Verschlüsseln und einen privaten zum Entschlüsseln. Den öffentliche Schlüssel darf jeder kennen, der private Schlüssel muss beim Eigentümer des Schlüssels bleiben. Zum Austausch eines symmetrischen Schlüssels muss ein sicheres Schlüsseltauschprotokoll verwendet werden. Der öffentliche Teil eines asymmetrischen Schlüssels kann über unsichere Kanäle verschickt werden, wobei jedoch die Authentizität des empfangenen Schlüssels sichergestellt werden muss. (2 VP)

A.7
```
public int fak(int n) {
  if (n == 0) {
    return 1;
  }
  return n*fak(n-1);
}
```
(1 VP)

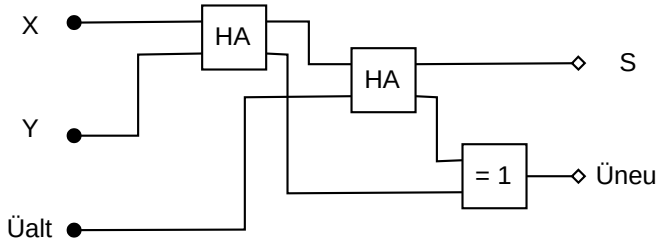
A.8 Ein Stack („Stapel“) ist ein so genannter „LIFO“(Last In, First Out)-Speicher: Er gibt die darin eingefügten Elemente bei der Entnahme in umgekehrter Reihenfolge wieder zurück, später eingefügte kommen also früher heraus. Die Methode `push(x: T)` legt einen neuen Wert auf den Stapel, die Methode `pop(): T` nimmt den zuletzt eingefügten Wert vom Stapel und gibt ihn dem Aufrufer zurück. (1 VP)

A.9

(a) (1 VP)



(b) (1 VP)



A.10 `NASHORN`
`RABERAB`

`EATLFRO` (1 VP)

A.11 Gut komprimierbar ist ein einfarbiges Bild; schlecht komprimierbar ist ein Schachbrettmuster, bei dem sich die Farbe bei jedem Pixel ändert. (1 VP)

A.12 (1 + 1 VP)

(a) 5,25

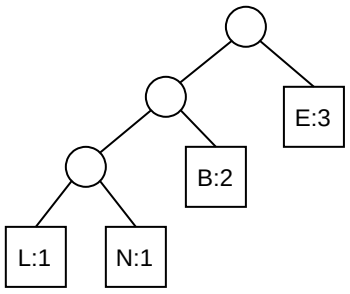
(b) 1,75 und $\frac{17}{64}$ sind exakt darstellbar; 0,3 ist es nicht.

A.13 (1 + 1 + 1 VP)

(a) Der linke Teilbaum hat das Gewicht 2 und hätte mit B zusammengefasst werden müssen.

(b) Jedes Codewort hat die Länge 2, damit wird das Wort mit insgesamt 14 Bit gespeichert. Dieser Wert ist nicht optimal.

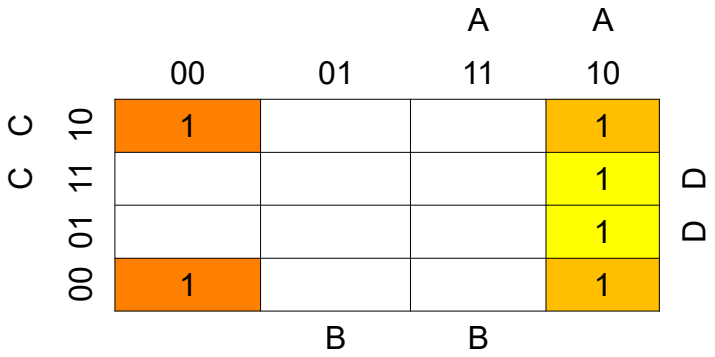
(c)



A.14
```
public boolean istPalindrom(String eingabe) {
    for (int i = 0; i < eingabe.length()/2; i++) {
      if (eingabe.charAt(i) != eingabe.charAt(eingabe.length()-i-1))
      {
        return false;
      }
    }
    return true;
}
```
(2 VP)

A.15 $f = x0 \wedge x1 \vee \neg x2$ (2 VP)

A.16 (2 VP)



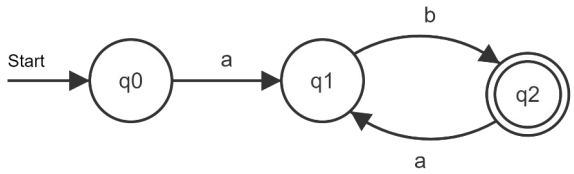
4er-Gruppe in der rechten Spalte eliminiert C und D, also $A \wedge \neg B$
4er-Gruppe in den Ecken eliminiert A und C, also $\neg B \wedge \neg D$
Damit X = $(A \wedge \neg B) \vee (\neg B \wedge \neg C)$ = $\neg B \wedge (A \vee \neg D)$.

Meta: Die Anordnung der Wahrheitswerte im Karnaugh-Veitch-Diagramm ist in der Literatur nicht einheitlich. Die Darstellung hier folgt derjenigen in Wikipedia.

A.17 babab – ja; bab – nein; bbbab – ja (1,5 VP)

A.18 (1 + 1 + 1 VP)

(a) Zustandsübergangsdiagramm (ohne Fehlerzustand)

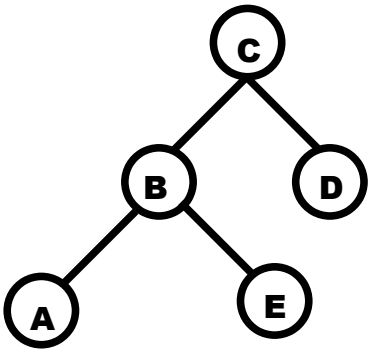


(b) Das Wort `bbaabb` wird nicht akzeptiert.
Das Wort `ababab` wird akzeptiert.

(c) regulärer Ausdruck: L = $(ab)^+$

A.19 (1 + 1 + 1 VP)

(a) Breitensuchbaum



(b) Eulerweg: viele Lösungsmöglichkeiten, z.B. ABCDEBDAE

(c) Es gibt keinen Eulerkreis, denn für einen Eulerkreis darf es keine Knoten mit ungeradem Grad im Graphen geben.
(Ein Eulerweg existiert, wenn neben den Knoten mit geradem Grad nur exakt zwei Knoten einen ungeraden Grad besitzen. Diese sind dann die Start und Endknoten.)

A.20 (1 + 1 VP)

(a) Laufzeit: Die äußere Schleife wird n-mal durchlaufen, die innere Schleife (n-1)-mal.
Damit findet der Vergleich innerhalb der inneren Schleife n*(n-1) = (n_2 – n)-mal statt.
Da der Summand mit der höchsten Potenz ausschlaggebend ist, liegt die Laufzeit in $O(n^2)$.

(b) Im Algorithmus wird eine boolsche Variable `vertauscht` eingeführt, die überprüft, ob eine Vertauschung durchgeführt wurde oder nicht. Der äußere Durchlauf muss dann nicht immer n-mal durchlaufen werden. Ein angepasster Algorithmus kann wie folgt aussehen:

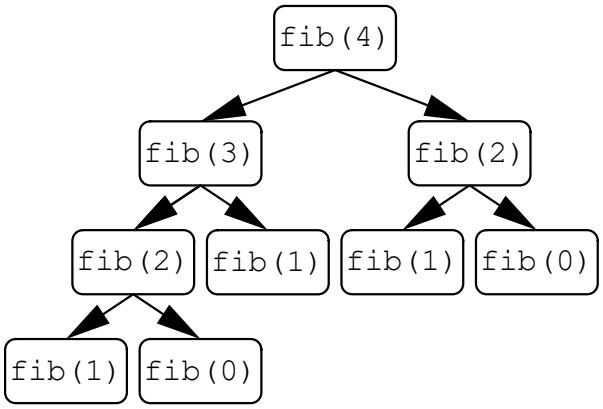
```
setze vertauscht auf false
wiederhole {
         wiederhole für j von 1 bis n – 1
               wenn (A[j-1] > A[j]) {
                     vertausche(A[j-1],A[j])
                     setze vertauscht auf true
               }
         }
   } solange (vertauscht)
```

<u>Wahlaufgabe B1:</u>

B1.1

a) <u>Angabe der nächsten vier Folgenglieder (1 VP):</u>
21, 34, 55, 89

b) <u>Darstellung der Methodenaufrufe (2 VP):</u>



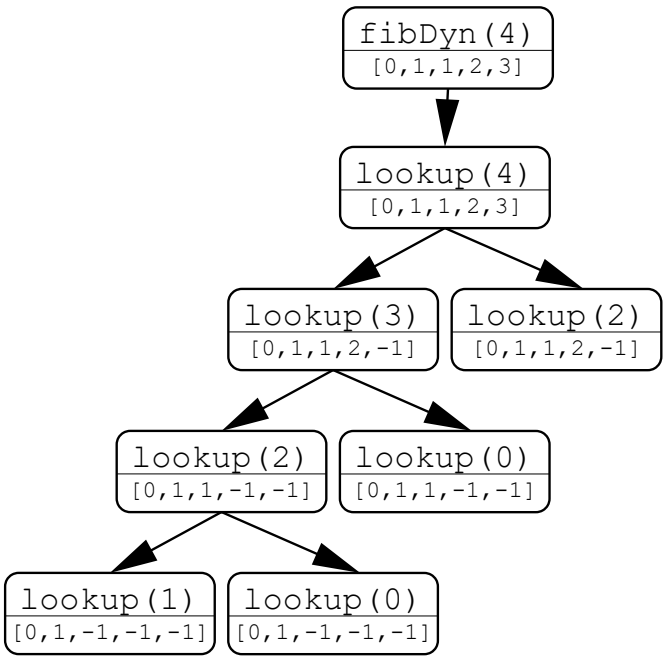
c) Begründung der Anzahl der Methodenaufrufe (2 VP):

Wenn die Methode `fib` in die letzte Zeile kommt, werden zwei rekursive Aufrufe ausgeführt. Dadurch bilden die rekursiven Aufrufe einen unvollständigen Binärbaum. Seine Höhe beträgt n+1. Ein Baum dieser Höhe hat weniger als 2^{n+1} Knoten.

d) Implementation von `fibIterativ` (2,5 VP):

```
public int fibIterativ(int n) {
  if (n < 2) {
    return n;
  }
  int f2 = 0;
  int f1 = 1;
  int f = 1;
  for (int i = 2; i < n; i++) {
    f2 = f1;
    f1 = f;
    f = f1+f2;
  }
  return f;
}
```

e) Darstellung der Methodenaufrufe (3 VP)



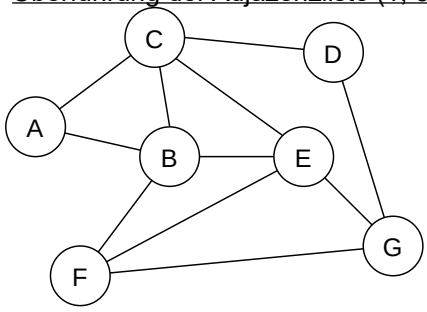
f) Begründung der Effizienz (1,5 VP)

Die ursprüngliche Implementation berechnet fast alle Funktionswerte mehrfach. Durch die Verwendung des `cache`-Arrays wird ein einmal berechneter Wert direkt nachgeschlagen und eine Rekursion ist nicht nötig.

Wahlaufgabe B2 – Variante 1:

B2.1

a) Überführung der Adjazenzliste (1, 5 VP):



b) Länge der kürzesten Pfade für jeden Knoten von A aus (1VP):
A – 0, B – 1, C – 1, D – 2, E – 2, F – 2, G – 3

c) Beschreibung der Breitensuche (2 VP):
Man startet die Breitensuche beim Startknoten und trägt dort die Entfernung 0 ein. Während der Breitensuche werden die Knoten des Graphen nacheinander besucht. Vom jeweils aktuellen Knoten aus werden alle Nachbarn betrachtet und bei den neu besuchten Knoten wird als Entfernung der Wert „1 + Entfernung des aktuellen Knotens“ eingetragen. Außerdem wird der aktuelle Knoten als Vorgänger der neu besuchten Knoten eingetragen.

d) Begründung der Modifikation (2,5 VP):
Der Algorithmus muss modifiziert werden, weil Knoten, die später besucht werden, eine niedrigere Distanz haben können als früher besuchte. Daher muss sichergestellt werden, dass stets der Knoten als nächstes abgearbeitet wird, der die niedrigste Distanz hat.

Meta: Im Bildungsplan werden Dijkstra und Bellman-Ford nur als Beispiele genannt.
Daher können sich Fragen im Abitur nicht auf einen konkreten Algorithmus beziehen. Die obige Antwort bezieht sich auf den Dijkstra-Algorithmus.

B2.2

a) Erreichbare Stellungen (1, 5 VP):

```
{ 0, 4, 3, 2, 1, 6, 7, 5, 8 }
{ 2, 4, 3, 1, 0, 6, 7, 5, 8 }
{ 2, 4, 3, 7, 1, 6, 0, 5, 8 }
```

b) Implementierung der Methode `istNachbarfeld` (2,5 VP):

```
boolean istNachbarfeld(int a, int b) {
  int zeileA = a / 3;
  int spalteA = a % 3;
  int zeileB = b / 3;
  int spalteB = b % 3;
  return zeileA == zeileB && Math.abs(spalteA - spalteB) == 1 ||
         spalteA == spalteB && Math.abs(zeileA - zeileB) == 1;
}
```

c) Implementierung der Methode `existiertKante` (4 VP):

```
boolean existiertKante(int[] a, int[] b) {
  ArrayList<Integer> unterschiede = new ArrayList<Integer>();
  for (int i = 0;  i < 9; i++) {
    if (a[i] != b[i]) {
      unterschiede.add(i);
    }
  }
  if (unterschiede.size() != 2) {
    return false;
  }
  int u1 = unterschiede.get(0);
  int u2 = unterschiede.get(1);
  if (!istNachbarfeld(u1, u2)) {
    return false;
  }
  // Sind beide Werte != 0, ist das Produkt es auch
  // Damit ist sichergestellt, dass die 0 wirklich vorkommt
  if (a[u1] * b[u1] != 0 || a[u2] * b[u2] != 0) {
    return false;
  }
  if (a[u1] != b[u2] || a[u2] != b[u1]) {
    return false;
  }
  return true;
}
```

Meta: Die Programmieraufgabe (c) ist zu schwer. Die Abiturkommission hat sie bewusst mit angegeben, um den erwarteten Schwierigkeitsgrad nach oben hin abzugrenzen.

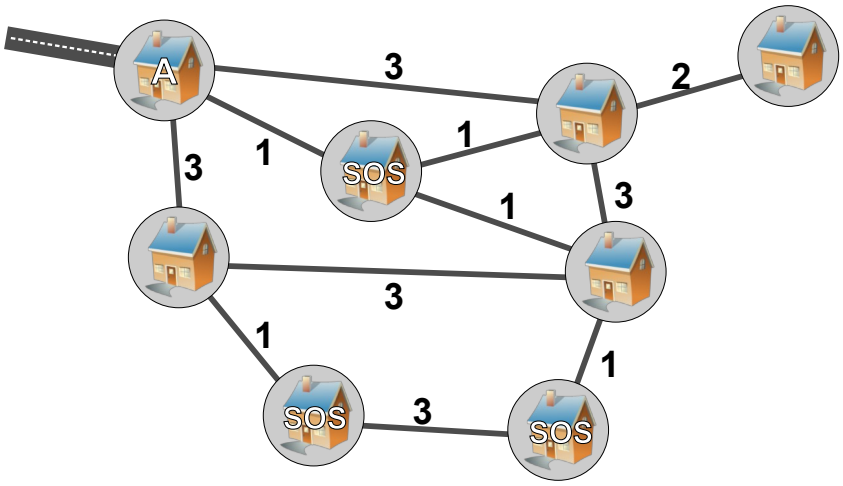
d) Erläuterung der Pfade und ihrer Längen (2 VP):
Startet man mit der Anfangsstellung, erhält man die Zugfolgen zu jeder erreichbaren Stellung und die Anzahl der nötigen Züge. Startet man mit der Zielstellung, erhält man von jeder lösbaren Stellung aus die kürzeste Zugfolge zur Zielstellung.

e) Beschreibung eines Algorithmus zur Identifikation unlösbarer Stellungen (2 VP):
Beginnend mit der Zielstellung führt man eine Breiten- oder Tiefensuche durch und markiert dabei die besuchten Knoten. Gibt es danach noch unmarkierte Knoten, so gibt es unlösbare Stellungen.

Wahlaufgabe B2 – Variante 2:

B2.1

a) Bestimmung der Zustände der Straßen (2 VP):



b) Begründung der Regel (1 VP)

Gäbe es eine benutzbare Straße von einer erreichbaren Stadt zu einer unerreichbaren, dann wäre diese automatisch selbst erreichbar. Die Straße muss also unbenutzbar sein.

c) Erläuterung der Breitensuche und der Anwendung im gegebenen Problem (3 VP)

Die Breitensuche geht von einem Knoten aus. Man fügt die Startstadt A zu einer ToDo-Liste hinzu und kennzeichnet sie als besucht. Um die Anzahl der Städte zu bestimmen, verwendet man einen Zähler und setzt ihn auf 0.
Während der Breitensuche werden die Knoten in der Reihenfolge, wie sie der ToDo-Liste hinzugefügt wurden, abgearbeitet und neue, noch nicht bearbeitete Knoten der ToDo-Liste am Ende hinzugefügt.

Man nimmt also den ersten Knoten aus der ToDo-Liste und erhöht den Zähler. Dann geht man alle Kanten entlang, die nicht als unbenutzbar markiert sind. Ist der Nachbarknoten noch nicht als besucht markiert, markiert man ihn und fügt man ihn der ToDo-Liste hinzu. Dies wiederholt man solange, bis die ToDo-Liste leer ist. Der Zähler gibt am Ende an, wie viele Städte erreichbar sind.

d) Implementierung von `bestimmeStrassenzustand` (5 VP):

```
public void bestimmeStrassenzustand(Graph g, Knoten startknoten)
  List<Knoten> nichtMarkiert = g.getAlleKnoten(k -> !k.isMarkiert());
  int anz = nichtMarkiert.size();
  List<Kante> alleKanten = g.getAlleKanten();
  for(Kante e : alleKanten) {
    e.setStatus(3);
    if(e.getStart().isMarkiert() != e.getZiel().isMarkiert()) {
      e.setStatus(1);
    }
  }
```

```
    for(Kante e : alleKanten) {
      if(!e.getStart().isMarkiert() && !e.getZiel().isMarkiert()) {
        e.setStatus(1);
        int anz2 = bestimmeAnzahlErreichbare(startKnoten);
        if(anz2 < anz) {
          e.setStatus(2);
        } else {
          e.setStatus(3);
        }
      }
    }
  }
```

Wahlaufgabe B3:

B3.1

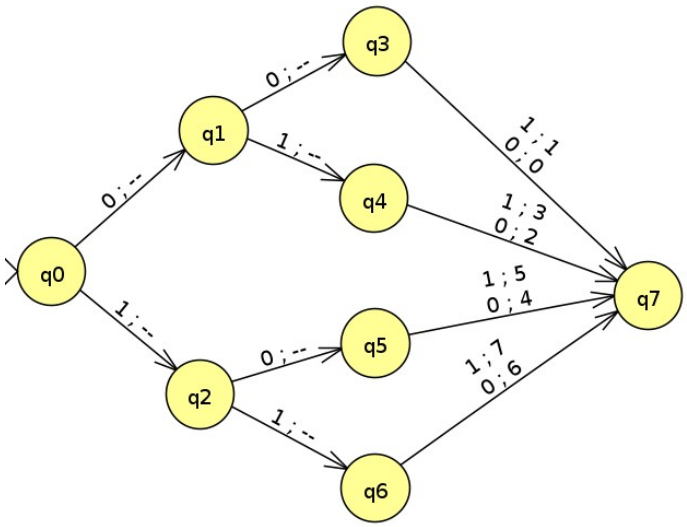
(a) Begründung (1,5 VP)

Für jede dieser Drei-Bit-Gruppen gibt es 8 Möglichkeiten (000 bis 111); jede dieser Gruppen lässt sich daher durch genau eine Oktalziffer platzsparend codieren.

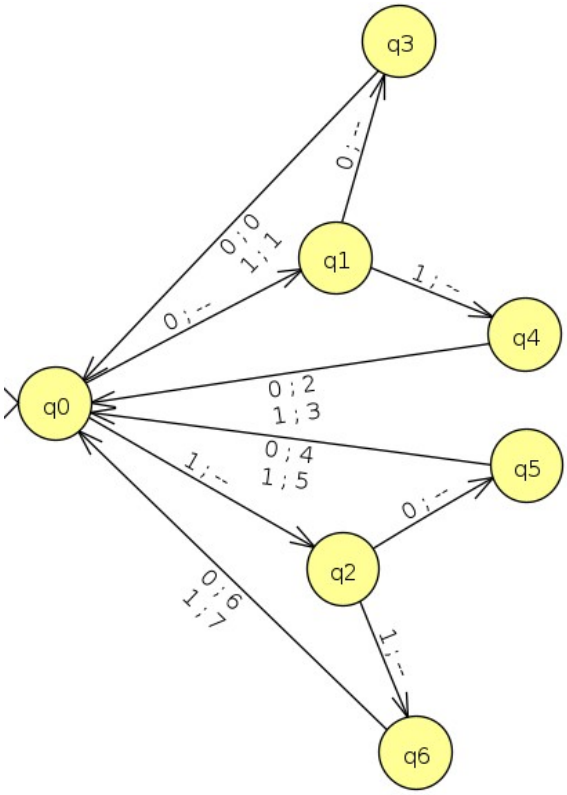
(b) Angabe der Oktaldarstellung (1 VP):

$547_{(8)}$

(c) Mealy-Automat für 3 Bit (3 VP):



(d) Erweiterung des Mealy-Automaten (1 VP):



(e) Verhalten bei beliebigen Längen (3 VP):
Der Automat gibt die Oktalzahl 53 aus. Wenn die Länge der Eingabe nicht durch 3 teilbar ist, bleibt er mitten in der Bearbeitung stehen. Da er nicht voraussehen kann, wie lang die Eingabe ist, kann er sie nicht passend in Dreiergruppen aufteilen. Er kann sie auch nicht rückwärts durchlaufen, weil er keinen Stack oder eine ähnliche Datenstruktur besitzt. Er kann einmal ausgegebene Symbole auch nicht wieder zurücknehmen.

Wahlaufgabe B4:

B4.1

(a) SQL-Abfragen

- `SELECT Titel, Preis FROM Buecher WHERE Verlag='Carlsen'`
- Veröffentlichte Bücher (2 VP)

```
SELECT Verlag, COUNT(*) AS Anzahl FROM Buecher
WHERE Jahr >= 1990 AND Jahr <= 2016 GROUP BY Verlag
```

- Lieblingsautoren pro Leser (3 VP)

```
SELECT Nachname, Vorname, Autor, AVG(Punkte) AS Mittelwert
FROM Leser, Buecher, Bewertung
WHERE Leser.LID = Bewertung.LID AND Buecher.BID = Bewertung.BID
GROUP BY Bewertung.LID, Buecher.Autor
ORDER BY Nachname, Vorname, Mittelwert DESC
```

Hinweis: Alternative Lösungen mit JOIN sind auch gültig.

(b) Begründung des falschen Durchschnitts (1 VP)

Durch das Kreuzprodukt der Tabellen „Bewertung“ und „Buecher“ gibt es 5·9 = 45 Zeilen. Durch die Selektion (WHERE-Klausel) gibt es nur noch 1·9 = 9 Zeilen. Die Funktion AVG berechnet nun mit allen 9 Punktbewertungen in der Tabelle „Bewertung“ den Durchschnitt $\frac{38}{9} \approx 4{,}22222$.

(c) Korrekter Durchschnitt (1 VP)

```
SELECT AVG(Punkte) FROM Buecher, Bewertung
WHERE Bewertung.BID = Buecher.BID AND Titel='Das Parfüm'
```

Hinweis: Alternative Lösungen mit JOIN sind auch gültig.

(d) Beispiele für Interessen des Verlags (1 VP)

- Welcher Leser interessiert sich für welche Art von Büchern?
- Welche Bücher im Sortiment sind populär, welche weniger?

usw.

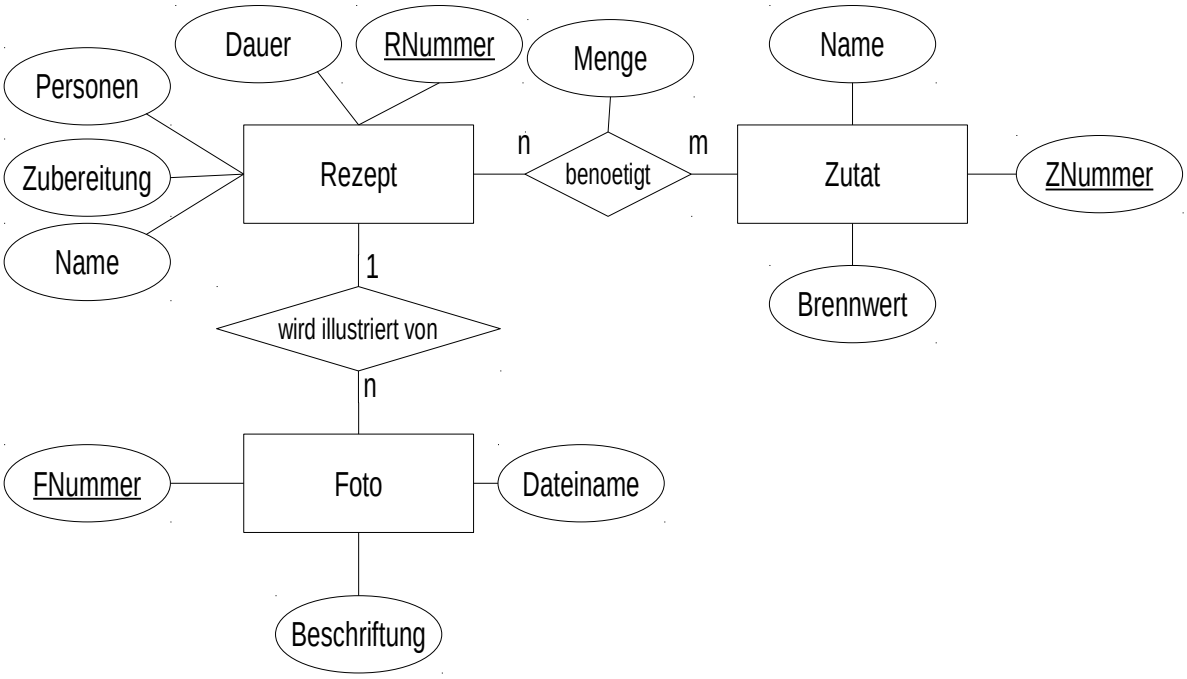
Viele plausible Antworten sind denkbar.

(e) Gründe gegen die Herausgabe der Datenbank (1 VP)

- Es handelt sich hier um persönliche Daten der Leser, die zudem mit Klarnamen versehen sind. Ohne deren Zustimmung dürfen diese nicht an Dritte weitergegeben werden, es sei denn, die Leser haben ausdrücklich zugestimmt. Dies gilt aufgrund des Bundes- bzw. Landesdatenschutzgesetzes.
- Selbst wenn die Daten anonymisiert werden, kann man unter Umständen andere Datensätze damit verknüpfen und auf die Vorlieben der Leser schließen.

B4.2

(a) Entity-Relationship-Diagramm mit Primärschlüssel und Kardinalitäten (4 VP)



Hinweis: Die Speicherung eines Fotos in der Datenbank anstelle eines Dateinamens ist auch korrekt.

(b) Entschlüsselung des Rezepts (2 VP)

Da man weiß, dass es ein Plätzchenrezept ist, kann man davon ausgehen, dass Wörter wie "Zutaten", "Mehl", "Zucker" oder "Eier" im Text enthalten sind. Das Wort "Zutaten" findet man sofort in der linken Spalte und hat damit bereits sechs Buchstaben identifiziert. Anhand dieser kann man noch mehr Wörter teilweise entschlüsseln und auf weitere Ersetzungen schließen.
Alternativ: Eine Häufigkeitsanalyse könnte genauso zum Erfolg führen, um z.B. die Buchstaben "e", "n", "r", "s" und "t" zu identifizieren.