Abiturprüfung 2023

INFORMATIK

Arbeitszeit: 180 Minuten

Der Fachausschuss wählt je eine Aufgabe aus den Gebieten
Inf1 und Inf2 zur Bearbeitung aus.

Der Fachausschuss ergänzt im folgenden Feld die erlaubten
objektorientierten Programmiersprachen:

INF1. MODELLIERUNG UND PROGRAMMIERUNG

I.

BE

Im Rahmen eines P-Seminars soll eine Software entwickelt werden, mit der sich verschiedene schulische Angebote leichter organisieren lassen. Die Anwendung soll verschiedene Module zur Verfügung stellen; unter anderem sind als Module ein Planer für Studienfahrten und eine Nachhilfebörse vorgesehen. Sowohl Schülerinnen und Schüler als auch Lehrkräfte sollen die Anwendung nutzen können.

6 1. Das Modell für die Software soll eine Klasse PORTAL enthalten, die sowohl die Nutzenden als auch die Module verwaltet. Nutzende sollen die Möglichkeit haben, ihr Passwort zu ändern. Sie sollen bestimmte Module verwenden können, die sie zuvor für sich freischalten müssen.

   Modellieren Sie das beschriebene Szenario in einem Klassendiagramm. Verzichten Sie auf die Angabe von Attributen. Wenden Sie dabei an mindestens einer Stelle das Prinzip der Vererbung an.

2. Die Nutzenden, die beim Portal registriert sind, werden in einem lexikografisch geordneten Binärbaum verwaltet. Die Ordnung beruht auf dem eindeutigen Loginnamen. Ein erster Vorschlag zur Umsetzung sieht vor, dass die Nutzenden sich selbst registrieren. Dabei wird automatisch ein Loginname generiert und anschließend das zugehörige Nutzer-Objekt in den Baum sortiert eingefügt.

5 a) Die Schülerin Ada gibt zu bedenken, dass sich die Einfügereihenfolge bei der Verwendung des Baums als ungünstig erweisen kann. Zeichnen Sie für die folgenden Beispiele jeweils den Baum, der beim Einfügen der Nutzer-Objekte entsteht. Erläutern Sie, inwiefern Adas Einwand berechtigt ist.

   - Einfügereihenfolge 1:
     mayer03, schubert02, lange01, bucher01, neupert01
   - Einfügereihenfolge 2:
     bucher01, lange01, neupert01, mayer03, schubert02

4 b) Gehen Sie nun davon aus, dass sich bisher 600 Einträge im Baum befinden. Geben Sie begründet an, um wie viele Ebenen der Baum minimal und maximal erweitert werden muss, wenn weitere 220 Einträge eingefügt werden.

(Fortsetzung nächste Seite)

BE

Einem zweiten Vorschlag folgend sollen die Daten aller Nutzenden vorab ins Portal importiert werden. Das Verwaltungsprogramm der Schule bietet die Möglichkeit, eine Liste aller Mitglieder der Schulfamilie zu exportieren. Um diese Liste für den Import verwenden zu können, werden automatisch eindeutige Loginnamen ergänzt und die Liste nach diesen lexikografisch sortiert.

4

c) Um zu vermeiden, dass beim Einfügen ein zur Liste entarteter Baum entsteht, schlägt der Schüler Alan vor, mit dem mittleren Element bzw. einem der beiden mittleren Elemente der Liste zu beginnen und die restlichen Elemente in lexikografischer Reihenfolge zu importieren. Begründen Sie, dass dieses Vorgehen hinsichtlich des Aufbaus des Baums nicht optimal ist. Geben Sie auch an, wie viele Ebenen ein solcher Baum in Abhängigkeit von der Länge *n* der Liste hat.

6

d) Der beschriebene Binärbaum wird mit einer Klasse BAUM verwaltet. In dieser Klasse soll eine rekursive Methode *alleEinfügen(liste)* zur Verfügung stehen, die eine übergebene, lexikografisch sortierte Liste *liste* in einen Binärbaum mit möglichst wenig Ebenen überführt. Formulieren Sie auf Basis der Idee, ein mittleres Element zu verwenden, einen Algorithmus für diese Methode, z. B. in Pseudocode.

Gehen Sie davon aus, dass der Baum beim ersten Aufruf der Methode *alleEinfügen(liste)* leer ist.

Es stehen folgende Methoden bereits zur Verfügung:

In der Klasse BAUM:

- *einfügen(element)* fügt das übergebene Objekt *element* sortiert ein.

In der Klasse LISTE:

- *mittleresElementGeben()* gibt das mittlere Element der Liste zurück. Bei Listen mit geradzahliger Länge wird das erste der beiden infrage kommenden Elemente zurückgegeben.
- *linkeTeillisteGeben()* gibt die linke Teilliste ohne das gemäß *mittleresElementGeben* festgelegte mittlere Element zurück.
- *rechteTeillisteGeben()* gibt die rechte Teilliste ohne das gemäß *mittleresElementGeben* festgelegte mittlere Element zurück.
- *istLeer()* liefert genau dann *wahr*, wenn die aufrufende Liste leer ist.

(Fortsetzung nächste Seite)

BE

Nach dem Import wird für alle Nutzenden dasselbe Standardpasswort gesetzt, das gleich beim ersten Login geändert werden muss. Passwörter werden verschlüsselt gespeichert; der verschlüsselte Wert des Standardpassworts ist „GH7uu4“.

2 e) Bewerten Sie das oben skizzierte Vorgehen zur Verteilung von Standardpasswörtern hinsichtlich Datenschutz und Datensicherheit unter der Voraussetzung, dass die Loginnamen nach einem allgemein bekannten Schema aus den Klarnamen gebildet werden.

8 f) Es soll nun überprüft werden, welche Nutzenden das Standardpasswort noch nicht geändert haben. Dazu soll es eine Methode *nutzerMitStandardpasswortAusgeben()* geben, die die Loginnamen der betroffenen Nutzenden in lexikografischer Reihenfolge ausgibt.

Ein Nutzer-Objekt hat eine Methode *passwortGeben()*, die den aktuellen Wert des verschlüsselten Passworts als Zeichenkette zurückgibt. Des Weiteren existiert eine Methode *ausgeben()*, die den Loginnamen des entsprechenden Nutzenden ausgibt.

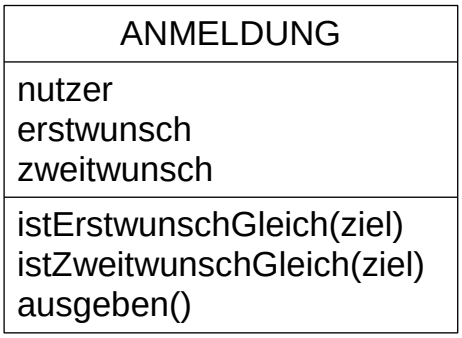
Gehen Sie im Folgenden von einem Baum aus, der unter Verwendung des Entwurfsmusters Kompositum mit den Klassen BAUM, BAUMELEMENT, ABSCHLUSS und KNOTEN implementiert wurde. Die Trennung von Struktur und Daten ist durch ein Referenzattribut *inhalt* der Klasse KNOTEN realisiert.

Notieren Sie in einer auf dem Deckblatt angegebenen Programmiersprache eine Implementierung der Methode *nutzerMitStandardpasswortAusgeben()* in der Klasse BAUM und aller dafür benötigten Methoden in den Klassen der Baumstruktur. Verwenden Sie das Prinzip der Rekursion.

(Fortsetzung nächste Seite)

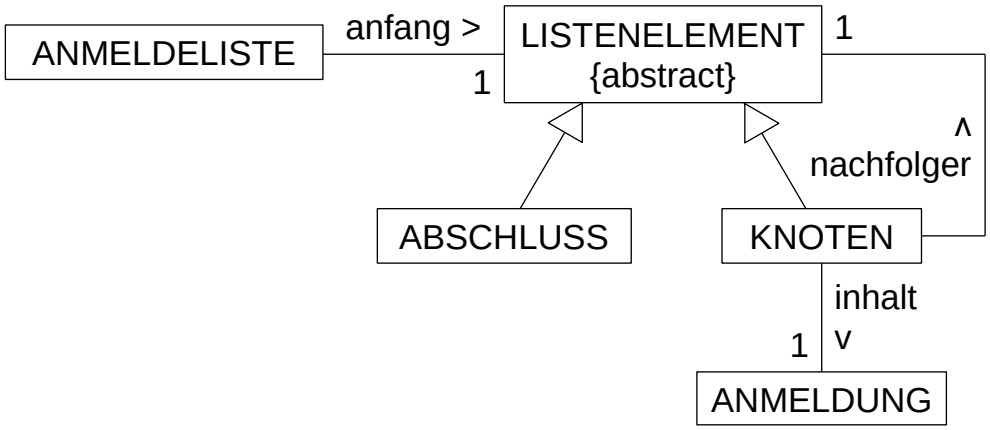
BE

3. Im Planer für Studienfahrten tragen Lehrkräfte zunächst die angebotenen Ziele ein. Anschließend melden sich die Schülerinnen und Schüler unter Abgabe eines Erstwunsches und eines Zweitwunsches zur Studienfahrt an. Anmeldungen werden in einer Klasse ANMELDUNG verwaltet:



- *istErstwunschGleich(ziel)* gibt genau dann *wahr* zurück, wenn die Zeichenkette *ziel* dem in der Anmeldung gespeicherten Erstwunsch entspricht.
- *istZweitwunschGleich(ziel)* gibt genau dann *wahr* zurück, wenn die Zeichenkette *ziel* dem in der Anmeldung gespeicherten Zweitwunsch entspricht.
- *ausgeben()* gibt die Informationen der Anmeldung formatiert aus.

Die Anmeldungen werden in einer einfach verketteten Liste gemäß folgendem Klassendiagramm gespeichert:



Eine neue Anmeldung wird jeweils hinten in die Liste eingefügt.

(Fortsetzung nächste Seite)

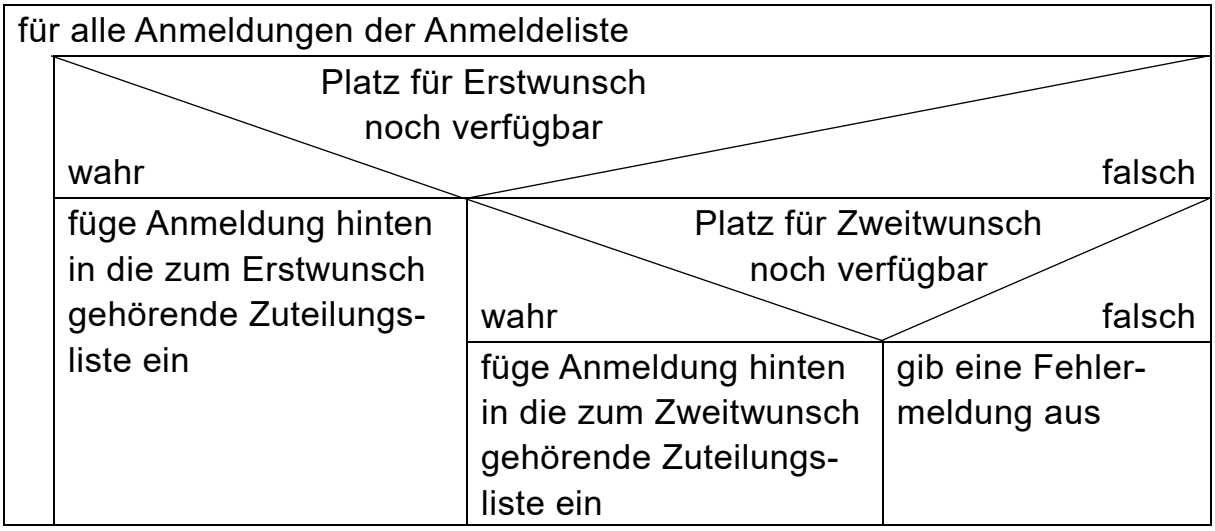
BE

10

a) Für den Fall, dass eine Studienfahrt mit einem bestimmten Ziel ausfällt, sollen alle Anmeldungen, die dieses Ziel als Erst- oder Zweitwunsch eingetragen haben, ausgegeben und aus der Liste gelöscht werden. Dazu soll in der Klasse ANMELDELISTE eine Methode *ausgebenUndLöschen(ziel)* zur Verfügung stehen.

Notieren Sie in einer auf dem Deckblatt angegebenen Programmiersprache eine mögliche Implementierung dieser Methode und aller dafür benötigten Methoden in den Klassen der Listenstruktur. Verwenden Sie dabei das Prinzip der Rekursion.

Sämtliche Anmeldungen werden in einer gemeinsamen Anmeldeliste verwaltet. Nachdem sich alle Schülerinnen und Schüler angemeldet haben, wird versucht, die Anmeldungen den Zielen zuzuordnen. Für jedes Ziel steht nur eine begrenzte Anzahl an Plätzen zur Verfügung. Das Struktogramm beschreibt einen möglichen Algorithmus für die Zuteilung, der für jedes Ziel eine eigene, zunächst leere Zuteilungsliste befüllt.



(Fortsetzung nächste Seite)

BE

4

b) Folgende Tabelle zeigt einen Ausschnitt der Anmeldeliste:

Name	**Erstwunsch**	**Zweitwunsch**
…	…	…
Albert Tross	Rom	Berlin
Thor Schluss	Paris	Rom
Martha Pfahl	Paris	Rom
Dennis Schläger	Rom	Berlin
…	…	…

Auf den ersten Teil der Anmeldungen bis vor Albert Tross soll der Algorithmus bereits angewendet worden sein. In der Liste für das Ziel Rom sind dadurch schon 23 von 25 möglichen Plätzen belegt. Für die Fahrt nach Paris gibt es nur noch einen freien Platz. Die Liste für Berlin ist bereits voll.

Führen Sie den gegebenen Algorithmus für die vier in der Tabelle aufgeführten Anmeldungen fort und geben Sie an, welchen Zielen die vier Schülerinnen und Schüler ggf. zugeordnet werden.

2

c) Geben Sie an, welche Anmeldungen der angegebene Algorithmus bevorzugt, und beschreiben Sie eine kleine Modifikation, die diese Bevorzugung verhindern würde.

9

d) Zur Verwaltung der Zuteilungslisten gibt es eine Klasse ZUTEILUNGSLISTE, deren Struktur derjenigen der Anmeldeliste entspricht. Das Ziel einer Zuteilungsliste ist im Attribut *ziel* dieser Klasse gespeichert.

Für jede Zuteilungsliste soll ermittelt werden, wie groß der Anteil an Zuteilungen ist, bei denen der Erstwunsch in der Anmeldung mit dem Ziel dieser Zuteilungsliste übereinstimmt. Notieren Sie dazu in einer auf dem Deckblatt angegebenen Programmiersprache eine mögliche Implementierung der Methode *anteilErfüllteErstwünsche()* in der Klasse ZUTEILUNGSLISTE, die den gewünschten Anteil berechnet und zurückgibt, sowie aller dazu nötigen Methoden in den weiteren Klassen der Listenstruktur. Verwenden Sie dabei das Prinzip der Rekursion.

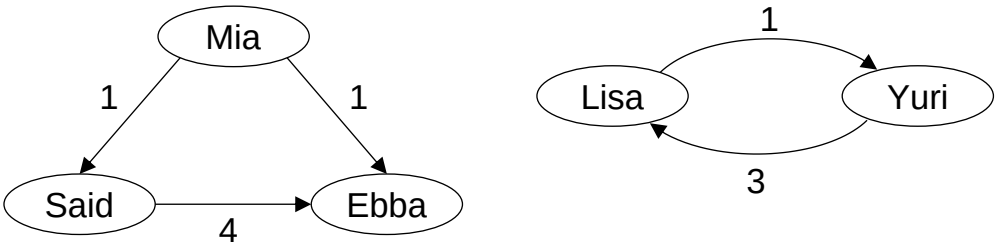
Gehen Sie davon aus, dass in der Klasse ZUTEILUNGSLISTE eine Methode *längeGeben()* zur Bestimmung der Länge der Liste zur Verfügung steht.

(Fortsetzung nächste Seite)

BE

4. Bei der Nachhilfebörse können sich nur Schülerinnen und Schüler anmelden. Es soll gespeichert werden, wer in welchem Fach Nachhilfestunden erteilt bzw. erhält.

Der Schüler Antoine schlägt vor, dazu einen gerichteten und gewichteten Graphen zu verwenden. Dabei zeigt die Kante von der Person, die Nachhilfe erteilt, zu der, die von ihr Nachhilfe erhält; die Kantengewichte stehen für den Code des jeweiligen Fachs. Für einige Beispieldaten hat er bereits den folgenden Graphen gezeichnet:



Fächer sind darin wie folgt mit Zahlen codiert:

Fach	**Code**
Mathematik	1
Deutsch	2
Latein	3
Englisch	4
…	…

Zur Implementierung solcher Graphen existiert die Klasse GRAPH. Sie soll eine Methode *nachhilfeschülerAusgeben(code)* besitzen, welche die Namen all derer auf dem Bildschirm ausgibt, die Nachhilfe in dem als Code übergebenen Fach erhalten.

4 a) Beschreiben Sie die Semantik der Beziehungen am Beispiel von Said, Ebba und Mia.

Ein Algorithmus zum Graphendurchlauf ist für die Umsetzung der Methode *nachhilfeschülerAusgeben* nicht geeignet. Erklären Sie dies anhand zweier Aspekte, die in obigem Graphen auftreten.

4 b) Übertragen Sie den gegebenen Graphen in eine Adjazenzmatrix.

(Fortsetzung nächste Seite)

BE	
	Die Klasse GRAPH speichert die Anzahl der Knoten in einem Attribut *anzahlKnoten*, die Knoten in einem Feld *knoten* und die Adjazenzmatrix in dem zweidimensionalen Feld *matrix*.
5	c) Formulieren Sie z. B. in Pseudocode einen Algorithmus für die Methode *nachhilfeschülerAusgeben(code)*. Sie können dabei davon ausgehen, dass in der Klasse KNOTEN eine Methode *nameAusgeben()* zur Verfügung steht, die den Namen der Schülerin bzw. des Schülers auf dem Bildschirm ausgibt, den der jeweilige Knoten repräsentiert. Die mehrfache Ausgabe desselben Namens muss nicht verhindert werden.
7	d) Die Information, wer wem in welchem Fach Nachhilfe erteilt, kann unter Umständen in einem Graphen, wie er von Antoine vorgeschlagen wurde, nicht vollständig abgebildet werden. Formulieren Sie im Sachzusammenhang ein Szenario, für das der von Antoine vorgeschlagene Graph nicht ausreicht. Erläutern Sie zudem, wie Sie die Nachhilfebörse ohne Verwendung eines Graphen modellieren würden, um dieses Problem zu lösen.
80	

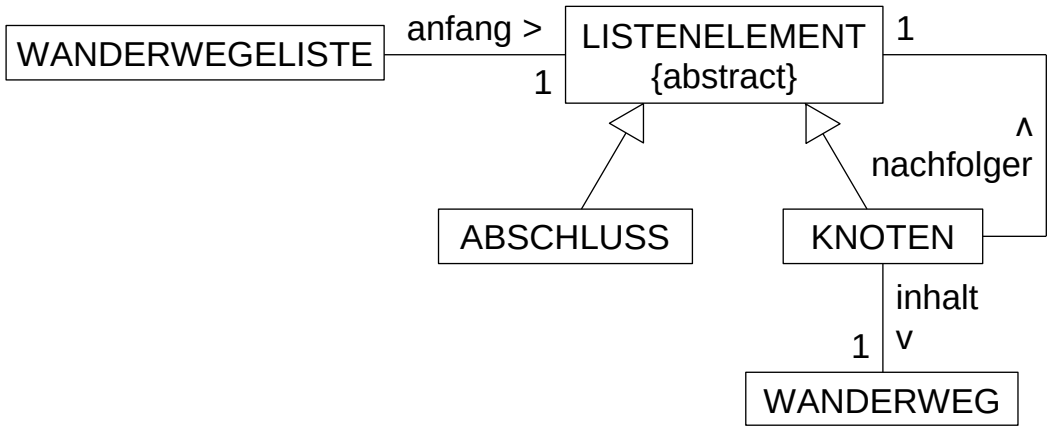
INF1. MODELLIERUNG UND PROGRAMMIERUNG

II.

BE

Der Naturpark Bayerischer Wald rund um den Großen Arber erfreut sich in den letzten Jahren immer größerer touristischer Beliebtheit. Die Tourismusabteilung plant aus diesem Grund u. a. die Bereitstellung einer eigenen „Arberland-App“, die den Nutzerinnen und Nutzern interessante Informationen beispielsweise zu Wanderwegen und Attraktionen im Naturparkgebiet bietet. Weiter soll ein zentrales Buchungssystem für Übernachtungen in der Region eingeführt werden.

1. Zur Verwaltung der Wanderwege, die sich innerhalb des Nationalparks befinden und in der App beworben werden sollen, wird eine einfach verkettete Liste gemäß folgendem Klassendiagramm eingesetzt.



In der Klasse WANDERWEG werden im Attribut *name* der Name des Wanderwegs, im Attribut *länge* seine Länge in Kilometern, im Attribut *höhe* die zu bewältigenden Höhenmeter und im Attribut *bewertung* die durchschnittliche Bewertung zwischen 0 und 5 Sternen gespeichert. Zudem hat die Klasse Methoden zum Geben der Attributwerte.

4 a) Erstellen Sie gemäß dem gegebenen Klassendiagramm ein Objektdiagramm einer Wanderwegeliste *wwl*, die genau die beiden Wanderwege „Steinbachfälle“ (5,7 km lang, 560 Höhenmeter, 4,5 Sterne) und „Rachelsteig“ (7,3 km lang, 620 Höhenmeter, 4,3 Sterne) enthält. Geben Sie jeweils an, zu welcher Klasse die Objekte gehören.

(Fortsetzung nächste Seite)

BE

10

b) In der Arberland-App soll mit der Gesamtlänge aller in der Liste gespeicherten Wanderwege geworben werden. Hierzu dient die rekursive Methode *gesamtlängeGeben()* der Klasse WANDERWEGELISTE, die die Summe aller Wanderwegelängen berechnet und zurückgibt.

Stellen Sie den Ablauf, der durch den Aufruf der Methode *gesamtlängeGeben()* für das Objekt *wwl* der Klasse WANDERWEGELISTE aus Teilaufgabe 1a ausgelöst wird, grafisch (z. B. in einem Sequenzdiagramm) dar. Berücksichtigen Sie dabei alle Objekte des Objektdiagramms.

Begründen Sie im Sachzusammenhang, dass die tatsächliche Gesamtlänge des erfassten Wegenetzes möglicherweise geringer ausfällt als die durch die Methode *gesamtlängeGeben* berechnete.

5

c) Um den Naturpark leichter mit anderen Wanderregionen vergleichen zu können, soll der Durchschnitt der Schwierigkeiten aller Wanderwege berechnet werden. Dabei wird die Schwierigkeit eines Wanderwegs durch den Quotienten aus den zu bewältigenden Höhenmetern und seiner Länge (in Kilometern) festgelegt.

Beschreiben Sie in eigenen Worten, wie Sie diese Berechnung mit der gegebenen Listenstruktur realisieren würden.

8

d) Nach einiger Zeit sollen unbeliebte Wanderwege (z. B. mit einer Bewertung von weniger als 2,5 Sternen) nicht mehr in der App angezeigt werden. Dazu dient eine rekursive Methode *wanderwegeLöschen(wert)*, die alle Wanderwege, deren Bewertung kleiner als der übergebene Wert ist, aus der Liste löscht.

Notieren Sie mithilfe einer auf dem Deckblatt angegebenen Programmiersprache gemäß dem gegebenen Klassendiagramm eine Implementierung der Methode *wanderwegeLöschen* in der Klasse WANDERWEGELISTE und der dafür nötigen Methoden aller weiteren Klassen der Listenstruktur. Wenden Sie so weit wie möglich das Prinzip der Rekursion an. Die Klasse WANDERWEG kann als vollständig implementiert vorausgesetzt werden.

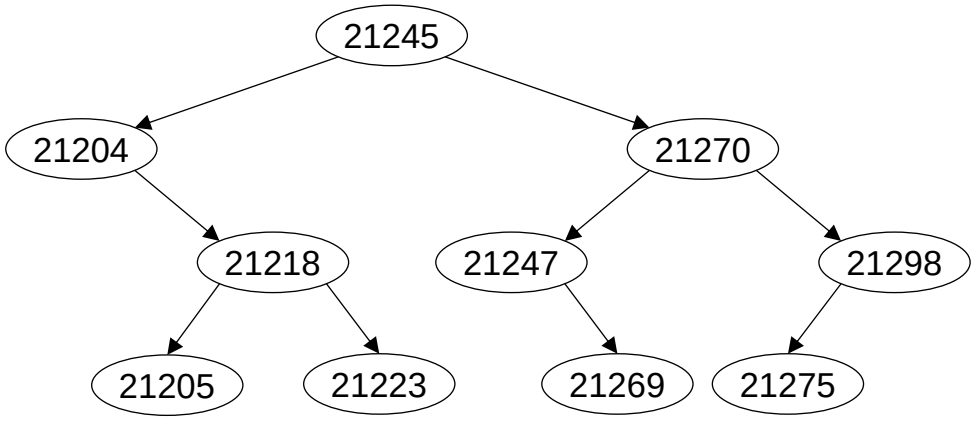
(Fortsetzung nächste Seite)

BE

2. Zur Verwaltung der Übernachtungen entscheidet sich die Tourismusabteilung für den Einsatz eines Buchungssystems, das die Buchungen in einem geordneten Binärbaum erfasst. Als Schlüssel wird hierfür eine für jede Buchung eindeutige, positive ganzzahlige Buchungs-ID verwendet. Außerdem werden u. a. der Name der für die Buchung verantwortlichen Person, die Anzahl der Nächte, die Anzahl der Personen und das Jahr des Anreisetags gespeichert.

2 a) Es werden Buchungen in einen zunächst leeren, geordneten Binärbaum eingefügt. Erläutern Sie, inwiefern dadurch ein Baum entstehen kann, der für die Suche nach einer Buchungs-ID nicht optimal aufgebaut ist.

4 b) Im folgenden geordneten Binärbaum sind bereits zehn Buchungen eingefügt worden. Zur Vereinfachung werden diese nur durch die jeweilige Buchungs-ID dargestellt.



In den gegebenen Baum sollen möglichst viele weitere Buchungen eingefügt werden, ohne dass weitere Ebenen hinzukommen. Geben Sie mögliche Buchungs-IDs zusammen mit einer geeigneten Einfügereihenfolge an, wenn nur Buchungs-IDs größer als 21200 vergeben werden sollen.

(Fortsetzung nächste Seite)

BE

5

c) Im Buchungssystem sind mittlerweile 25 000 Buchungen erfasst. Statt eines geordneten Binärbaums hätte sich die Tourismusabteilung auch für eine einfach verkettete Liste entscheiden können. Diese bietet aber weder bei der Suche nach einer Buchung anhand der Buchungs-ID noch bei der Suche nach dem Namen einer für die Buchung verantwortlichen Person einen Vorteil gegenüber einem optimal aufgebauten geordneten Binärbaum.

Begründen Sie dies, indem Sie für beide Suchvorgänge die maximale Anzahl der Vergleichsoperationen in einem optimal aufgebauten geordneten Binärbaum mit der maximalen Anzahl der Vergleichsoperationen in einer einfach verketteten Liste vergleichen.

11

d) Bei der Implementierung des geordneten Binärbaums wurde das Entwurfsmuster Kompositum verwendet sowie die Trennung von Struktur und Daten berücksichtigt. Die dabei verwendeten Klassen heißen BUCHUNGSVERZEICHNIS, BAUMELEMENT, KNOTEN, ABSCHLUSS und BUCHUNG.

Die Klasse BUCHUNGSVERZEICHNIS soll die Methode *gesamtzahlÜbernachtungenGeben(jahr)* besitzen, welche die Anzahl an Übernachtungen für ein bestimmtes Jahr zurückgibt. Berücksichtigen Sie hierbei insbesondere, dass eine Buchung mehrere Nächte und mehrere Personen beinhalten kann. Buchungen über den Jahreswechsel werden dabei vollständig zum alten Jahr gezählt.

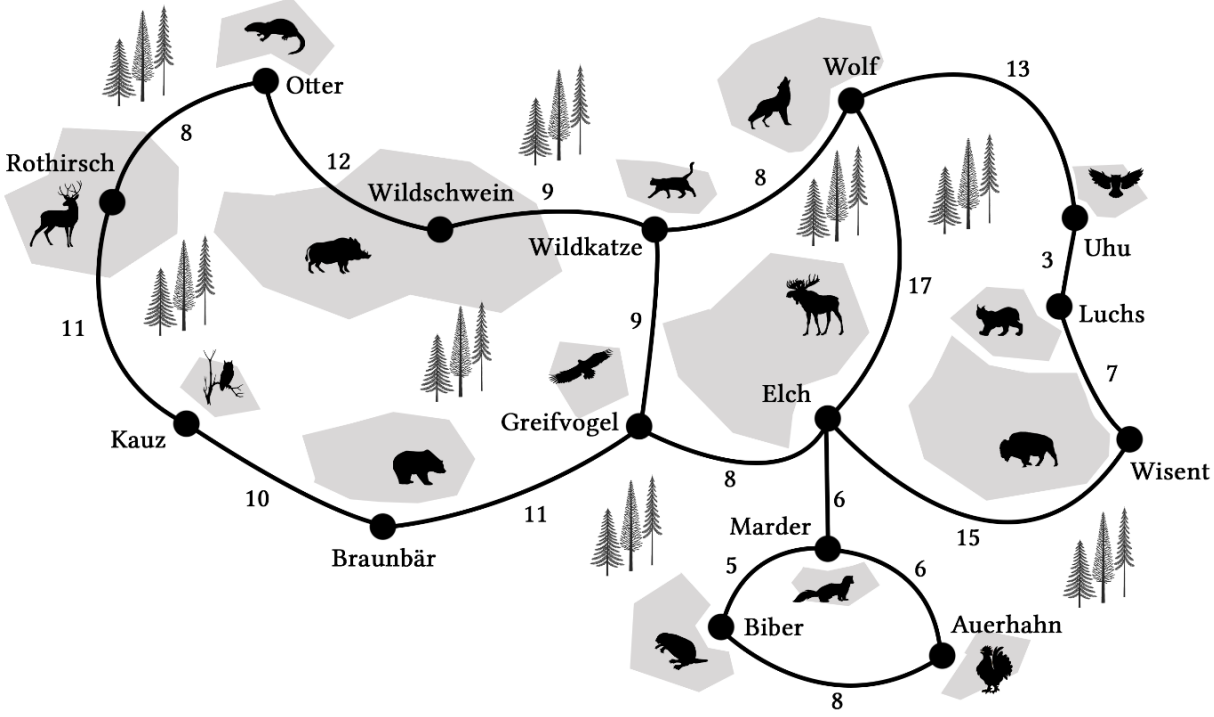
Notieren Sie in einer auf dem Deckblatt angegebenen Programmiersprache eine Implementierung der betroffenen Klassen der Baumstruktur. Beschränken Sie sich hierbei auf diejenigen Attribute und Methoden, die zur Umsetzung der Methode *gesamtzahlÜbernachtungenGeben* erforderlich sind. Verwenden Sie dabei so weit wie möglich das Prinzip der Rekursion.

Die Klasse BUCHUNG darf einschließlich der Methoden zum Geben der Attributwerte als vollständig implementiert vorausgesetzt werden.

(Fortsetzung nächste Seite)

BE

3. Als besondere Attraktion des Naturparks Bayerischer Wald gilt ein Tierfreigelände, das auf rund 250 Hektar verschiedene Gehege für heimische Vogel- und Säugetierarten bietet. Der folgende Plan zeigt eine Übersicht über das Tierfreigelände, wobei die Zahlen die ungefähren Gehzeiten (in Minuten) zwischen den Gehegen angeben.



Der dargestellte Plan kann als Graph aufgefasst werden.

Aufgrund der großen Besucherzahlen hat die Tourismusabteilung vor, die Wege im Tierfreigelände nur noch in einer Richtung begehbar zu machen.

6 a) Geben Sie eine Eigenschaft des Graphen an, die sich ändert, wenn die Wege nur noch in einer Richtung begehbar sind. Zeichnen Sie außerdem einen Ausschnitt des Graphen mit den Gehegen von Elch, Greifvogel, Marder und Wolf nach einer möglichen Umsetzung des Vorhabens der Tourismusabteilung und geben Sie für diesen Ausschnitt die Adjazenzmatrix an. Markieren Sie in Ihrer Adjazenzmatrix diejenigen Einträge, die sich im Vergleich zur Adjazenzmatrix des gegebenen Graphen geändert haben, und beschreiben Sie, wie sich die Umsetzung des Vorhabens allgemein auf die Adjazenzmatrix auswirkt.

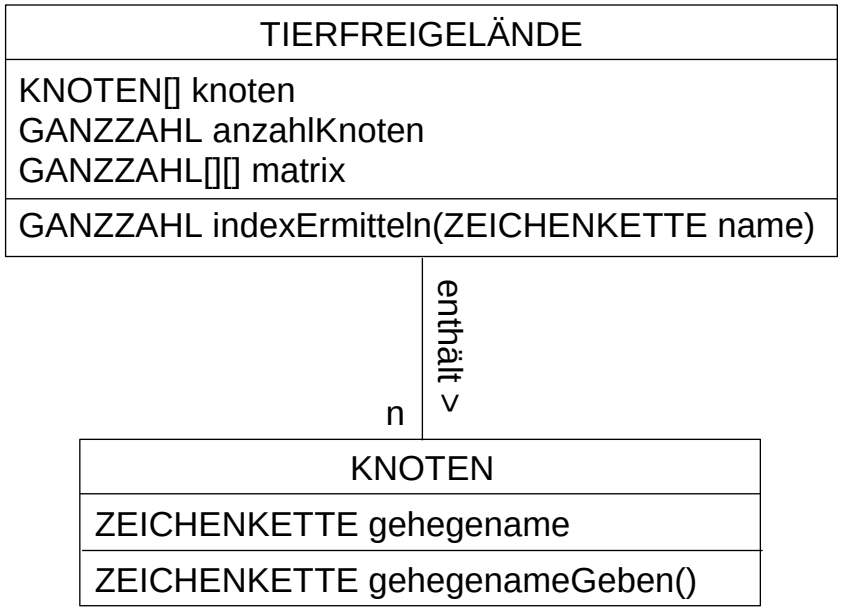
(Fortsetzung nächste Seite)

BE	
5	b) Begründen Sie, dass das Vorhaben, alle Wege nur noch in einer Richtung begehbar zu machen, nicht so umgesetzt werden kann, dass ein Rundweg möglich ist. Ein Rundweg bedeutet, dass die Tour bei demselben Gehege endet, an dem sie begonnen wurde, wobei alle weiteren Gehege genau einmal besucht werden. Geben Sie so wenig Wege wie möglich an, die von einem Gehege zu einem anderen neu angelegt werden müssen, sodass es eine Umsetzung des Vorhabens gibt, die einen Rundweg ermöglicht.
3	c) Zeigen Sie, dass es möglich ist, Laufrichtungen so zu vergeben, dass ein Gehege von keinem anderen aus mehr erreichbar ist. Beschreiben Sie, wie man ein solches Gehege anhand der Adjazenzmatrix identifizieren kann.
5	d) Erläutern Sie, inwiefern ein Verfahren zum Graphendurchlauf dazu herangezogen werden kann, um zu erkennen, ob nach einer Umsetzung des Vorhabens alle Gehege von allen anderen Gehegen aus erreichbar sind. Eine Implementierung dieses Verfahrens ist nicht verlangt.

(Fortsetzung nächste Seite)

BE

Der Implementierung des Graphen, mit dem das Wegenetz im Tierfreigelände verwaltet wird, liegt folgendes Klassendiagramm zugrunde:



Das Attribut *anzahlKnoten* der Klasse TIERFREIGELÄNDE gibt an, wie viele Knoten aktuell im Feld *knoten* gespeichert sind. Das zweidimensionale Feld *matrix* repräsentiert die Adjazenzmatrix.

Die Klasse TIERFREIGELÄNDE verfügt darüber hinaus über eine Methode *indexErmitteln(name)*, die den Feldindex des Knotens mit dem übergebenen Gehegenamen zurückgibt. Falls kein Knoten mit dem entsprechenden Namen existiert, wird −1 zurückgegeben.

Die Klassen TIERFREIGELÄNDE und KNOTEN mit den in obigem Klassendiagramm angegebenen Methoden dürfen im Folgenden als bereits implementiert vorausgesetzt werden.

(Fortsetzung nächste Seite)

BE	
7	e) Notieren Sie in einer auf dem Deckblatt angegebenen Programmiersprache eine Implementierung der Methode *wegEinfügen(name1, name2, gehzeit, ungerichtet)* der Klasse TIERFREIGELÄNDE. Diese Methode fügt eine Kante mit dem Kantengewicht *gehzeit* zwischen zwei Knoten, deren Werte im Attribut *gehegename* mit den übergebenen Namen übereinstimmen, ein bzw. aktualisiert sie. Hat der Parameter *ungerichtet* den Wahrheitswert *wahr*, ist die Kante ungerichtet, ansonsten verläuft die Kante vom Knoten mit dem Gehegenamen *name1* zum Knoten mit dem Gehegenamen *name2*. Falls zu einem der übergebenen Namen kein Knoten existiert, beide Knoten identisch sind oder die übergebene Gehzeit negativ ist, soll der Zustand des Graphen nicht verändert und *falsch* zurückgegeben werden, ansonsten *wahr*.
5	f) Als neues Feature der Arberland-App sollen den Nutzerinnen und Nutzern vom aktuellen Gehege aus die direkt benachbarten Gehege mit der jeweiligen Gehzeit angezeigt werden. Dazu muss eine Methode *alleNachbargehegeGeben(name)* in der Klasse TIERFREIGELÄNDE zur Verfügung stehen, die eine entsprechende Zeichenkette, bestehend aus den Gehegenamen und den dazugehörigen Gehzeiten, erzeugt und zurückgibt. Dabei wird im Parameter *name* der Name des Geheges übergeben, dessen Nachbargehege gegeben werden sollen. Notieren Sie in einer auf dem Deckblatt angegebenen Programmiersprache eine Implementierung dieser Methode.
80	

III.

BE

Die Firma „Bike-a-Bit“ betreibt einen Fahrradverleih. An verschiedenen Standorten befinden sich jeweils mehrere Fahrräder unterschiedlichen Typs, die über ein Online-Reservierungssystem gebucht werden können.

1. Zur Auswahl stehen konventionelle Fahrräder und E-Bikes für Erwachsene sowie Kinderfahrräder in verschiedenen Größen.

   Zur internen Verwaltung wird jedem Fahrrad eine eindeutige Kennung zugewiesen. Sie besteht aus einem oder zwei Großbuchstaben für den Standort, gefolgt von einer natürlichen Zahl von 1 bis 99. Handelt es sich um ein E-Bike, so folgt noch der Buchstabe E. Handelt es sich um ein Kinderfahrrad, so folgen nach der Zahl der Buchstabe K und eine Größenangabe, die durch den Reifendurchmesser (in Zoll) mit den möglichen Werten 12, 14, 16, …, 24 beschrieben wird. Die Menge aller gültigen Kennungen bildet die formale Sprache L über dem Alphabet {0, 1, 2, …, 9, A, B, …, Z}.

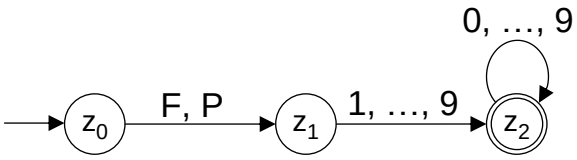
5 a) Notieren Sie in einer formalen Textnotation (z. B. EBNF) die Produktionsregeln einer Grammatik, die L erzeugt.

7 b) Geben Sie ein Zustandsübergangsdiagramm eines endlichen Automaten an, der genau die Wörter von L akzeptiert.

(Fortsetzung nächste Seite)

BE

8 c) Die Firma „Bike-a-Bit“ unterscheidet zwischen Firmen- und Privatkunden. Mögliche Kundennummern werden durch den folgenden Automaten beschrieben:



Formulieren Sie in einer auf dem Deckblatt angegebenen Programmiersprache eine Implementierung einer Klasse AUTOMAT, die den gegebenen Automaten umsetzt. Dabei soll es unter anderem eine Methode *istKundennummer(eingabe)* geben, die genau dann *wahr* zurückgibt, wenn die Zeichenkette *eingabe* eine mögliche Kundennummer ist.

Sie können davon ausgehen, dass in dieser Klasse eine Methode *istZiffer1bis9(z)* zur Verfügung steht, die genau dann *wahr* zurückgibt, wenn das Zeichen *z* eines der Zeichen von '1' bis '9' ist.

Außerdem dürfen Sie in der Klasse ZEICHENKETTE folgende Methoden als bereits vollständig implementiert voraussetzen:

- *länge()* gibt die Länge der Zeichenkette zurück.
- *zeichenAn(n)* gibt das *n*-te Zeichen der Zeichenkette zurück, wobei die Zählung bei 0 beginnt.

2. Über das Online-Reservierungssystem der Firma „Bike-a-Bit“ kann die Kundschaft jeweils für einen Tag ein bestimmtes Fahrrad mieten. Das System zeigt dazu eine Übersicht aller Fahrräder an einem gewählten Standort an. Nach Auswahl eines Fahrrads und Eingabe des gewünschten Datums wird geprüft, ob dieses Fahrrad an diesem Tag verfügbar ist. In diesem Fall erscheint ein Button „reservieren“, über den das Fahrrad gebucht werden kann. Nach Klick auf den Button wird die Verfügbarkeit erneut geprüft. Ist das Fahrrad immer noch verfügbar, wird die Buchung bestätigt, andernfalls wird eine Meldung ausgegeben, dass das Fahrrad nicht mehr verfügbar ist.

4 a) Stellen Sie ein Szenario grafisch (z. B. in einem Sequenzdiagramm) dar, das zu der Meldung führt, dass das Fahrrad nicht mehr verfügbar ist.

3 b) Es kann unter Umständen das Problem auftreten, dass ein Fahrrad für einen bestimmten Tag mehrfach gebucht wird. Beschreiben Sie den zugehörigen kritischen Abschnitt und begründen Sie, dass das Problem durch einen Monitor verhindert werden kann.

(Fortsetzung nächste Seite)

BE

3. Um die Fahrräder vor dem Zugriff von Unbefugten zu schützen, ist jedes Fahrrad mit einer dreistelligen PIN aus den Ziffern von 0 bis 9 gesichert. Nach erfolgreicher Reservierung erhält man die PIN, mit der man das Fahrrad am entsprechenden Tag über ein Tastenfeld entsperren kann.

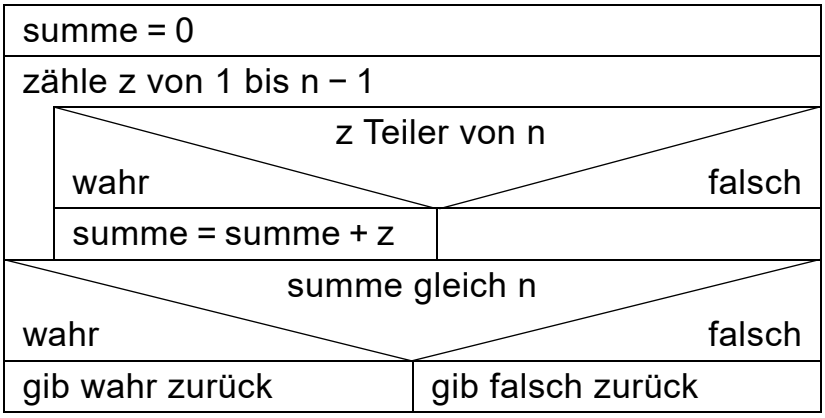
2 a) Berechnen Sie, wie lange ein Unbefugter maximal benötigt, um ein Fahrrad zu entsperren, wenn er für die Eingabe einer PIN 0,9 Sekunden benötigt, und beurteilen Sie die Sicherheit dieses Verfahrens.

3 b) Ein Mitarbeiter der Firma macht zur Erhöhung der Sicherheit den Vorschlag, dass ein Fahrrad, dessen PIN fünfmal falsch eingegeben wurde, für den Rest des Tages gesperrt wird.

Beschreiben Sie im Sachzusammenhang ein Problem, das auftreten kann, falls der Vorschlag umgesetzt würde. Geben Sie eine bessere Möglichkeit an, wie man die Sicherheit erhöhen kann, ohne die Vorgaben zur Bildung einer PIN zu ändern. Begründen Sie Ihre Antwort.

8 4. Eine natürliche Zahl *n* heißt vollkommen, wenn sie gleich der Summe ihrer Teiler außer sich selbst ist. Zum Beispiel ist 28 eine vollkommene Zahl, denn sie besitzt die Teiler 1, 2, 4, 7, 14, 28 und 1 + 2 + 4 + 7 + 14 = 28.

Der abgebildete Algorithmus überprüft, ob eine natürliche Zahl *n* vollkommen ist:



(Fortsetzung nächste Seite)

BE

Schreiben Sie ein Programm für eine Registermaschine mit dem folgenden Befehlssatz, das den gegebenen Algorithmus umsetzt.

Gehen Sie davon aus, dass der Wert für *n* bereits in Speicherzelle 100 abgelegt ist. Machen Sie deutlich, in welcher Speicherzelle der Rückgabewert nach Ablauf Ihres Programms steht, und geben Sie die Bedeutung der möglichen Rückgabewerte an.

load x	kopiert den Wert aus der Speicherzelle x in den Akkumulator
loadi n	lädt die ganze Zahl n in den Akkumulator
store x	kopiert den Wert aus dem Akkumulator in die Speicherzelle x
inc	erhöht den Wert im Akkumulator um 1
dec	verringert den Wert im Akkumulator um 1
add x	addiert den Wert aus der Speicherzelle x zum Wert im Akkumulator
sub x	subtrahiert den Wert aus der Speicherzelle x vom Wert im Akkumulator
mul x	multipliziert den Wert im Akkumulator mit dem Wert in Speicherzelle x
div x	dividiert den Wert im Akkumulator durch den Wert in Speicherzelle x (ganzzahlige Division)
jmp x	springt zum Befehl in Speicherzelle x
jeq x	springt zum Befehl in Speicherzelle x, falls der Wert im Akkumulator gleich 0 ist
jne x	springt zum Befehl in Speicherzelle x, falls der Wert im Akkumulator ungleich 0 ist
jge x	springt zum Befehl in Speicherzelle x, falls der Wert im Akkumulator positiv oder gleich 0 ist
jle x	springt zum Befehl in Speicherzelle x, falls der Wert im Akkumulator negativ oder gleich 0 ist
jgt x	springt zum Befehl in Speicherzelle x, falls der Wert im Akkumulator positiv ist
jlt x	springt zum Befehl in Speicherzelle x, falls der Wert im Akkumulator negativ ist
hold	beendet die Abarbeitung des Programms

40

IV.

BE

Beim Gütertransport mit Lkws gibt es, abhängig von der beförderten Ladung, verschiedene einzuhaltende Richtlinien.

1. Wird Gefahrgut mit einem Lkw transportiert, muss dieses Fahrzeug mit einer Gefahrentafel gekennzeichnet werden.

   Bei Beförderung von unverpacktem Gefahrgut (z. B. Benzin in Tanklastwagen) wird die Tafel mit zwei übereinander angebrachten Codes versehen. Der obere Code, die Gefahrnummer, legt die Art der Gefahr fest; der untere Code, die UN-Nummer, gibt Aufschluss über den beförderten Stoff.

   Die Gefahrnummer besteht aus zwei oder drei Ziffern. Die Ziffer 1 wird wegen möglicher Verwechslungsgefahr mit der Ziffer 7 nicht verwendet. Die Ziffer 0 ist außerdem nur als zweite Ziffer zulässig; in diesem Fall folgt auf die 0 keine weitere Ziffer. Wenn der transportierte Stoff auf gefährliche Weise mit Wasser reagiert, beginnt die Gefahrnummer zusätzlich mit dem Buchstaben X.

   Die UN-Nummer ist immer vierstellig, wobei für jede Stelle alle Ziffern einschließlich 0 zulässig sind.

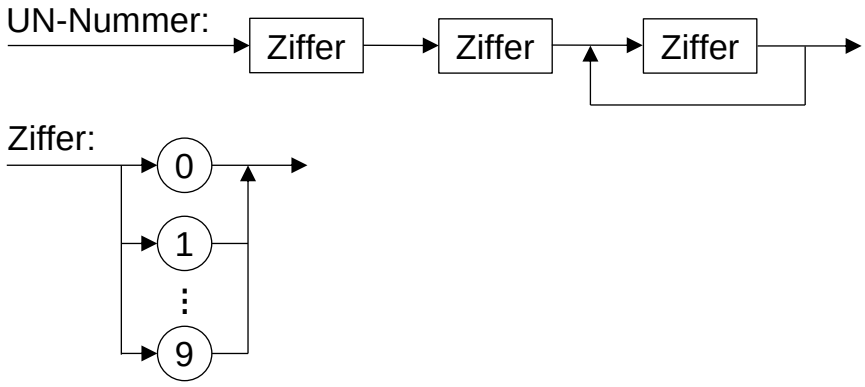
   Die Gefahrentafel rechts kennzeichnet beispielsweise den Transport von Benzin (UN-Nummer 1203). Benzin ist ein leicht entzündlicher, flüssiger Stoff, der nicht auf gefährliche Weise mit Wasser reagiert (Gefahrnummer 33).

33
1203

   Die Menge aller derartigen Beschriftungen für Gefahrentafeln bildet die formale Sprache L, wobei die Trennung zwischen Gefahrnummer und UN-Nummer durch einen Schrägstrich dargestellt wird. Beispielsweise wird die oben gegebene Gefahrentafel in der Form „33/1203“ notiert.

(Fortsetzung nächste Seite)

BE	
2	a) Mit folgendem Syntaxdiagramm soll versucht werden, Produktionsregeln für UN-Nummern anzugeben:



Obiges Diagramm erzeugt auch UN-Nummern, die nicht der gegebenen Beschreibung entsprechen. Nennen Sie eine solche UN-Nummer, die mit obigem Syntaxdiagramm erzeugt wird, und begründen Sie, dass diese nicht der gegebenen Beschreibung entspricht.

7 b) Notieren Sie in einer formalen Textnotation (z. B. in EBNF) die Produktionsregeln einer Grammatik, die L erzeugt. Geben Sie darüber hinaus ein Syntaxdiagramm an, das die Gefahrnummern (d. h. die oberen Codes der Gefahrentafeln) erzeugt.

5 c) Geben Sie das Zustandsübergangsdiagramm eines endlichen Automaten an, der genau die Gefahrnummern akzeptiert.

3 d) Erklären Sie die Begriffe Syntax und Semantik und nennen Sie ein Beispiel für eine syntaktisch korrekte Gefahrentafel mit fehlerhafter Semantik.

(Fortsetzung nächste Seite)

BE

2. Beim Transport von Gefahrgut muss ein Set bestehend aus mehreren Ausrüstungsgegenständen im Fahrzeug verpflichtend mitgeführt werden. So sind z. B. eine Warnweste, ein Paar Schutzhandschuhe, eine dicht schließende Schutzbrille und ein Atemschutz Bestandteil eines solchen Sets.

   Bei einer Logistikfirma werden alle notwendigen Gegenstände in einem zentralen Lager aufbewahrt. Muss Gefahrgut transportiert werden, holt sich der Fahrer bzw. die Fahrerin beim Eingang des Lagers ein Set ab. Ein Roboter trägt die entsprechenden Gegenstände in zufälliger Reihenfolge aus dem Lager zusammen und händigt sie anschließend als komplettes Set aus.

   Nach einer Tour wird das Set wieder im Lager abgegeben. Ein Roboter sortiert die einzelnen Bestandteile anschließend wieder in zufälliger Reihenfolge in die entsprechenden Lagerplätze ein.

   Es gibt zwei Roboter, wobei jeder Roboter Ausrüstungsgegenstände sowohl zusammenstellen als auch zurückbringen kann. Die Zusammenstellung eines Sets kann nicht unterbrochen werden.

3 a) Erklären Sie anhand eines konkreten Beispiels, warum es bei dieser Vorgehensweise zu einer Verklemmung kommen kann.

2 b) Erläutern Sie im Sachzusammenhang zwei Möglichkeiten, wie eine derartige Verklemmung verhindert werden kann.

(Fortsetzung nächste Seite)

BE

3. Beim Gütertransport dürfen Verpackungen gleicher Art bis zu einer Höhe von maximal 3 Metern gestapelt werden. Zunächst sollen würfelförmige Kisten mit einer Kantenlänge von 60 cm transportiert werden.

   Mithilfe der folgenden Methode kann die Anzahl an Stapeln bestimmt werden, die mindestens gebildet werden müssen. Der Parameter *z* steht dabei für die stets positive Anzahl der zu transportierenden Kisten.

```
Methode anzahlStapelGeben(z)
    ergebnis = 1
    wiederhole solange z > 5
        z = z − 5
        ergebnis = ergebnis + 1
    endeWiederhole
    gib ergebnis zurück
endeMethode
```

2 a) Geben Sie das Laufzeitverhalten von *anzahlStapelGeben* in Abhängigkeit von *z* an und begründen Sie Ihre Aussage.

3 b) Stellen Sie einen rekursiven Algorithmus grafisch (z. B. als Struktogramm) dar, der für alle zulässigen Eingaben von *z* denselben Rückgabewert liefert wie die Methode *anzahlStapelGeben*.

(Fortsetzung nächste Seite)

BE
3

c) Gegeben ist folgende Methode:

```
Methode anzahlStapelGeben2(z)
    wenn z mod 5 gleich 0 dann
        gib z / 5 zurück
    sonst
        gib z / 5 + 1 zurück
    endeWenn
endeMethode
```

Hinweis: Der Operator „/“ liefert den ganzzahligen Wert des Quotienten (ohne Rest) und der Operator „mod“ den Rest. Beispielsweise gilt: 17 / 5 = 3 und 17 mod 5 = 2.

Begründen Sie, dass diese Methode für positive ganzzahlige Eingaben das Gleiche leistet wie die Methode *anzahlStapelGeben*.

Geben Sie das Laufzeitverhalten von *anzahlStapelGeben2* in Abhängigkeit von *z* an und begründen Sie Ihre Aussage.

Es müssen nun quaderförmige Kisten transportiert werden, die die gleiche Grundfläche, aber unterschiedliche Höhen *h1* und *h2* haben.

Folgende Methode berechnet zu einer vorgegebenen maximalen Ladehöhe des Lkws die größtmögliche Höhe eines Stapels, der aus den beiden Kistenarten gebildet werden kann.

```
Methode maxStapelhöheGeben(h1, h2, ladehöhe)
    minResthöhe = ladehöhe
    zähle k von 0 bis ladehöhe / h1
        resthöhe = (ladehöhe − k * h1) mod h2
        wenn resthöhe < minResthöhe dann
            minResthöhe = resthöhe
        endeWenn
    endeZähle
    gib ladehöhe − minResthöhe zurück
endeMethode
```

Hinweis: Der Operator „/“ liefert den ganzzahligen Wert des Quotienten (ohne Rest) und der Operator „mod“ den Rest.

(Fortsetzung nächste Seite)

BE	
2	d) Geben Sie für den Methodenaufruf *maxStapelhöheGeben(80, 55, 200)* die Werte der Variablen *minResthöhe* an, die sie beim Ablauf der Methode der Reihe nach annimmt.
8	e) Schreiben Sie ein Programm für eine Registermaschine mit folgendem Befehlssatz, das die gegebene Methode *maxStapelhöheGeben* umsetzt. Machen Sie deutlich (z. B. durch symbolische Adressierung), in welchen Speicherzellen die Werte der Variablen und der Rückgabewert stehen.
40	

load x	kopiert den Wert aus der Speicherzelle x in den Akkumulator
loadi n	lädt die ganze Zahl n in den Akkumulator
store x	kopiert den Wert aus dem Akkumulator in die Speicherzelle x
add x	addiert den Wert aus der Speicherzelle x zum Wert im Akkumulator
addi n	addiert die ganze Zahl n zum Wert im Akkumulator
sub x	subtrahiert den Wert aus der Speicherzelle x vom Wert im Akkumulator
subi n	subtrahiert die ganze Zahl n vom Wert im Akkumulator
mul x	multipliziert den Wert im Akkumulator mit dem Wert in Speicherzelle x
muli n	multipliziert den Wert im Akkumulator mit der ganzen Zahl n
div x	dividiert den Wert im Akkumulator durch den Wert in Speicherzelle x (ganzzahlige Division)
mod x	berechnet den Rest bei der ganzzahligen Division des Werts im Akkumulator durch den Wert in Speicherzelle x
jmp x	springt zum Befehl in Speicherzelle x
jeq x	springt zum Befehl in Speicherzelle x, falls der Wert im Akkumulator gleich 0 ist
jne x	springt zum Befehl in Speicherzelle x, falls der Wert im Akkumulator ungleich 0 ist
jge x	springt zum Befehl in Speicherzelle x, falls der Wert im Akkumulator positiv oder gleich 0 ist
jle x	springt zum Befehl in Speicherzelle x, falls der Wert im Akkumulator negativ oder gleich 0 ist
jgt x	springt zum Befehl in Speicherzelle x, falls der Wert im Akkumulator positiv ist
jlt x	springt zum Befehl in Speicherzelle x, falls der Wert im Akkumulator negativ ist
hold	beendet die Abarbeitung des Programms