Abiturprüfung 2018

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	
	Um eine Alternative zu bestehenden sozialen Netzwerken zu bieten, entwickelt ein Team von Softwareentwicklern im Rahmen eines Softwareprojekts das Netzwerk *Abi18Net*.
5	1. a) Nennen und beschreiben Sie kurz die typischen Phasen zur Realisierung eines Softwareprojekts.
	Abi18Net stellt jedem Mitglied des Netzwerks eine Pinnwand zur Verfügung, auf der auch andere Mitglieder Nachrichten hinterlassen können. Die Nachrichten einer Pinnwand werden in einer einfach verketteten Liste gespeichert, die auf dem Softwaremuster Kompositum basiert und die Trennung von Struktur und Daten berücksichtigt.
5	b) Zeichnen Sie nach diesen Vorgaben ein Klassendiagramm, das unter anderem die Klassen PINNWAND und NACHRICHT enthält und auf dessen Grundlage die Pinnwand implementiert werden kann. Auf die Angabe von Attributen und Methoden kann verzichtet werden.
2	c) Nehmen Sie anhand zweier Argumente Stellung, inwiefern die Entscheidung für eine einfach verkettete Liste anstelle eines Feldes gerechtfertigt ist.
4	d) Erstellen Sie ein Objektdiagramm für eine Pinnwand mit drei Nachrichten gemäß dem Klassendiagramm aus Teilaufgabe 1b. Auf die Angabe von Attributen kann bei Objekten der Klasse NACHRICHT verzichtet werden. Geben Sie jeweils an, zu welcher Klasse die Objekte gehören.

(Fortsetzung nächste Seite)

BE	
8	e) Die Methode *anzahlNachrichtenGeben()* der Klasse PINNWAND gibt die Gesamtzahl aller auf der Pinnwand aktuell vorhandenen Nachrichten zurück. Notieren Sie mithilfe einer auf dem Deckblatt angegebenen Programmiersprache gemäß Ihrem Klassendiagramm aus Teilaufgabe 1b eine Implementierung der betroffenen Klassen der Listenstruktur. Beschränken Sie sich hierbei auf diejenigen Attribute und Methoden, die zur Umsetzung der Methode *anzahlNachrichtenGeben* erforderlich sind. Wenden Sie soweit wie möglich das Prinzip der Rekursion an. Die Klasse NACHRICHT darf dabei als vollständig implementiert vorausgesetzt werden.
4	f) Die Daten der Mitglieder werden in Objekten der Klasse MITGLIED verwaltet. Diese soll nun zusätzlich im Klassendiagramm aus Teilaufgabe 1b berücksichtigt werden. Geben Sie an, wie man in diesem Diagramm die bisher bekannten und die folgenden Anforderungen modellieren kann: ● Jede Pinnwand gehört genau einem Mitglied; ● jedes Mitglied hat Zugriff auf mehrere Pinnwände; ● jede Nachricht wurde von genau einem Mitglied verfasst.
	2. Um alle Mitglieder zu verwalten, wird bei *Abi18Net* ein geordneter Binärbaum verwendet. Als Schlüssel dient dabei eine für jedes Mitglied eindeutige Mitgliedsnummer.
2	a) Es wird zunächst vorgeschlagen, die Mitgliedsnummern in aufsteigender Reihenfolge gemäß der Anmeldung zu vergeben. Nehmen Sie zu diesem Vorschlag im Kontext der vorgesehenen Datenverwaltung Stellung.
4	b) Die Softwareentwickler beschließen abweichend vom Vorschlag aus Teilaufgabe 2a, dass die Mitgliedsnummern zufällige, noch nicht vergebene natürliche Zahlen sein sollen. Beschreiben Sie einen Algorithmus, der eine derartige Mitgliedsnummer findet, wenn Methoden zum Erzeugen einer Zufallszahl und zum Suchen einer Mitgliedsnummer im Binärbaum bereits zur Verfügung stehen.

(Fortsetzung nächste Seite)

BE	
3	c) Zeichnen Sie den entstehenden Binärbaum, wenn die Mitglieder mit den in Klammern angegebenen Mitgliedsnummern in dieser Reihenfolge in einen zunächst leeren Binärbaum eingefügt werden: Karla (3257), Alexej (66013), Bert (116), Dagmar (5485), Holger (2485), Jörg (833271), Paula (73680). Vereinfachend kann jeder Knoten durch die zugehörige Mitgliedsnummer bezeichnet werden.
3	d) Geben Sie die Bearbeitungsreihenfolge der Knoten an, wenn der Baum aus Teilaufgabe 2c nach der Postorder-Strategie traversiert wird.
3	e) *Abi18Net* rechnet mit 250000 Mitgliedern. Berechnen Sie, wie viele Ebenen ein Binärbaum mindestens haben muss, um darin die Daten von 250000 Mitgliedern verwalten zu können.
12	f) Der Binärbaum wird unter Verwendung der Klassen BAUM und MITGLIED sowie des Softwaremusters Kompositum mit Berücksichtigung der Trennung von Struktur und Daten umgesetzt. Die Klasse MITGLIED wird durch nachstehendes Diagramm beschrieben:

MITGLIED
mitgliedsnummer nachname vorname
alleDatenAusgeben() istNachnameGleich(suchnachname) istVornameGleich(suchvorname)

- Die Methode *alleDatenAusgeben()* gibt die Daten des Mitglieds aus.
- Die Methode *istNachnameGleich(suchnachname)* gibt genau dann *wahr* zurück, wenn der Parameter *suchnachname* den gleichen Wert wie das Attribut *nachname* des ausführenden Objekts hat.
- Die Methode *istVornameGleich(suchvorname)* gibt genau dann *wahr* zurück, wenn der Parameter *suchvorname* den gleichen Wert wie das Attribut *vorname* des ausführenden Objekts hat.

(Fortsetzung nächste Seite)

BE	
	Die Klasse BAUM soll eine Methode *mitgliederAusgeben(suchnachname, suchvorname)* besitzen, welche – geordnet nach den Mitgliedsnummern – die Daten derjenigen Mitglieder ausgibt, die die in den Parametern übergebenen Namen haben. Notieren Sie in einer auf dem Deckblatt angegebenen Programmiersprache eine mögliche Implementierung für die Klasse BAUM und alle weiteren Klassen der Baumstruktur. Beschränken Sie sich dabei auf diejenigen Attribute und Methoden, die zur Umsetzung der Methode *mitgliederAusgeben* erforderlich sind. Die Klasse MITGLIED kann als bereits implementiert vorausgesetzt werden.
3	g) Eine Datenbank enthält die Tabelle *mitglieder*, deren Spaltenbezeichner den Attributbezeichnern der Klasse MITGLIED entsprechen (vgl. Teilaufgabe 2f). Formulieren Sie eine Datenbankabfrage (z. B. in SQL), um alle Daten der Mitglieder mit dem Nachnamen „Meier“ und dem Vornamen „Laura“ zu erhalten.
	3. In *Abi18Net* ist es den Mitgliedern möglich, Freundschaftsanfragen an andere Mitglieder zu stellen. Wird eine solche Anfrage bestätigt, gelten die beiden Mitglieder als Freunde. Unter den Mitgliedern Alexej, Bert, Dagmar, Holger, Jörg, Karla und Paula bestehen aktuell folgende Beziehungen: Karla ist mit Alexej, Bert und Paula befreundet; darüber hinaus ist Paula mit Dagmar und Jörg, Jörg wiederum mit Bert und Holger befreundet. Außerdem hat Paula eine Freundschaftsanfrage an Alexej gerichtet, die noch nicht bestätigt wurde.
4	a) Stellen Sie die beschriebene Situation in einem Graphen dar und begründen Sie, warum ein ungerichteter Graph dafür ungeeignet ist. Verwenden Sie als Bezeichner für die Knoten die Anfangsbuchstaben der genannten Vornamen.
3	b) Geben Sie eine zu Ihrem Graphen aus Teilaufgabe 3a gehörende Adjazenzmatrix an.

(Fortsetzung nächste Seite)

BE

Die Klasse GRAPH speichert die benötigten Kanten in der zweidimensionalen Adjazenzmatrix *matrix* mit *n* Zeilen und *n* Spalten und die benötigten Knoten in einem Feld *knoten* der Länge *n* ab, um den Graphen aus Teilaufgabe 3a zu implementieren.

5 c) Beschreiben Sie die der folgenden Methode *methode1* zugrunde liegende Vorgehensweise und interpretieren Sie den Rückgabewert der Methode beim Aufruf für einen beliebigen Knoten *kn* im vorliegenden Kontext.

```
Methode methode1(kn)
    rück = 0
    i = Index von kn im Feld knoten
    wiederhole für j von 0 bis n-1
        wenn matrix[j][i] den Wert wahr hat
                    und matrix[i][j] den Wert wahr hat
            erhöhe rück um 1
        endeWenn
    endeWiederhole
    gib den Wert von rück zurück
endeMethode
```

Als Freundschaftsentfernung eines Knotens von einem anderen wird die Länge des kürzesten Pfades, der die beiden Knoten über bestätigte Freundschaftsanfragen miteinander verbindet, bezeichnet. Dabei bedeutet „Länge eines Pfades“ die Anzahl der aufeinanderfolgenden Kanten des Pfades. Die Freundschaftsentfernung eines Knotens zu sich selbst ist Null.

2 d) Ergänzen Sie alle Knoten Ihres Graphen aus Teilaufgabe 3a um ihre Freundschaftsentfernung vom Knoten *Karla*.

8 e) Freunde von Freunden sind im Graphen durch Knoten mit Freundschaftsentfernung 2 repräsentiert. Diese sollen einem Mitglied als mögliche neue Freunde vorgeschlagen werden. Beschreiben Sie dazu einen Algorithmus (z. B. in Pseudocode), der eine Methode *freundeVonFreunden(Knoten kn)* umsetzt; diese gibt alle Freunde von Freunden von *kn* aus, die nicht bereits direkt mit *kn* befreundet sind. Vereinfachend genügt es, nur die Indizes der betreffenden Knoten auszugeben.

80

II.

BE	
	1. Die Bayerische Verwaltung der staatlichen Schlösser, Gärten und Seen (kurz: Bayerische Schlösserverwaltung) ist unter anderem für den Betrieb der auf dem Königssee, Ammersee, Starnberger See und Tegernsee eingesetzten Passagierschiffsflotte zuständig. Jedes Schiff der Flotte ist dabei einem dieser Seen zugeteilt und hat neben seinem Namen eine Reihe von bauartbedingten Eigenschaften. Die Zentralverwaltung in München beabsichtigt, auf ein neues Softwaresystem umzustellen, das alle für die Organisation relevanten Aspekte berücksichtigt.
5	a) Modellieren Sie für den Neuentwurf des Systems die oben beschriebene Situation als Klassendiagramm unter Verwendung der Klassen SCHIFF, SEE und ANLEGESTELLE. Ergänzen Sie für jede Klasse mindestens zwei sinnvolle Attribute.
5	b) Aufgrund von Unterschieden in Betrieb und Wartung lässt sich der Flottenbestand in Elektroboote, Dieselmotorschiffe und mit Dieselmotoren ausgestattete Schaufelraddampfer gliedern. Passen Sie den betreffenden Teil Ihres Klassendiagramms so an, dass die unterschiedlichen Eigenschaften und Funktionalitäten der Schiffstypen berücksichtigt werden können. Nutzen Sie dabei das Prinzip von Spezialisierung und Generalisierung soweit wie möglich. Ergänzen Sie zudem passend die Methode *restbetriebsstundenBerechnen()*, die für Elektroboote und Schiffe mit Dieselmotoren auf unterschiedliche Art die restlichen Betriebsstunden ermittelt.

(Fortsetzung nächste Seite)

BE

Bei der Planung des Linienangebots geht man davon aus, dass ein Schiff ab seinem Wartungstermin für unbestimmte Zeit nicht mehr zur Verfügung steht. Der nächste Wartungstermin wird bei jedem Schiff in einem Attribut gespeichert und nach der Wartung aktualisiert.
Sie können im Folgenden davon ausgehen, dass bereits eine Klasse DATUM vorliegt, deren Objekte Datumswerte repräsentieren und welche die folgenden Methoden bereitstellt:

- Die Methode *istVor(DATUM termin)* gibt genau dann den Wahrheitswert *wahr* zurück, wenn das repräsentierte Datum vor dem übergebenen Termin liegt.
- Die Methode *istNach(DATUM termin)* gibt genau dann den Wahrheitswert *wahr* zurück, wenn das repräsentierte Datum nach dem übergebenen Termin liegt.
- Die Methode *dauer(DATUM termin)* liefert die Anzahl der Tage vom repräsentierten Datum bis einschließlich des übergebenen Termins.

5 c) In der Klasse SCHIFF soll es eine Methode *anzahlTage(saisonstart, saisonende)* geben, der das Datum des ersten und das Datum des letzten Tages der Saison übergeben werden und die in einer ganzen Zahl zurückgibt, wie viele Tage das Schiff unter Berücksichtigung einer möglichen Wartung in dieser Saison voraussichtlich zur Verfügung steht.
Notieren Sie in einer auf dem Deckblatt angegebenen Programmiersprache eine Implementierung dieser Methode. Es kann von sinnvollen Eingaben ausgegangen werden.

Die Schiffe werden in einer sortierten, einfach verketteten Liste verwaltet. Als Sortierkriterium wird dabei aus betrieblichen Überlegungen der Termin der nächsten Wartung gewählt. Für die Implementierung soll das Softwaremuster Kompositum eingesetzt und zudem das Konzept der Trennung von Daten und Struktur berücksichtigt werden.

3 d) Erläutern Sie, was man unter einem Softwaremuster versteht und inwiefern dessen Einsatz die Softwareentwicklung vereinfacht.

5 e) Zeichnen Sie ein Klassendiagramm ohne Attribute und Methoden, welches die beschriebene Modellierung der Schiffsliste darstellt.

(Fortsetzung nächste Seite)

BE

14 f) Die Methode *anzahlSchiffeBestimmen()* soll die Anzahl der Schiffe der Flotte zurückgeben; die Methode *einfuegen(schiff)* soll das übergebene Schiffsobjekt zur sortierten Datenstruktur hinzufügen. Notieren Sie in einer auf dem Deckblatt angegebenen Programmiersprache eine mögliche Implementierung dieser sowie aller dafür erforderlichen Methoden in den betroffenen Klassen der Listenstruktur. Verwenden Sie dabei die in Teilaufgabe 1e eingeführten Bezeichner. Wenden Sie soweit wie möglich das Prinzip der Rekursion an. Die Klasse SCHIFF darf dabei als vollständig implementiert vorausgesetzt werden.

2. In der zentralen Personalverwaltung sind die Daten aller Mitarbeiter der Bayerischen Schlösserverwaltung in einer relationalen Datenbank abgelegt, wobei als Primärschlüssel eine Personalnummer dient. Exemplarisch ist hier ein Ausschnitt aus der Tabelle *mitarbeiter* gezeigt:

persNr	name	vorname	dienstort	sonstige Spalten
152	Forstinger	Ludwig	Tegernsee	···
190	Wörgl	Johanna	Königssee	···
177	Hackl	Josefine	Königssee	···
109	Leitner	Karl	Tegernsee	···
188	Unterhuber	Florian	Ammersee	···
110	Obermair	Franziska	Ammersee	···
335	Ferstl	Hubert	Tegernsee	···

3 a) Formulieren Sie eine Datenbankabfrage (z. B. in SQL), welche die Personalnummer, den Namen und den Vornamen der Mitarbeiter mit Dienstort „Königssee“ ausgibt.

Die entwickelte Software liest die Mitarbeiterdaten aus der Datenbanktabelle in einen nach der Personalnummer geordneten Binärbaum ein.

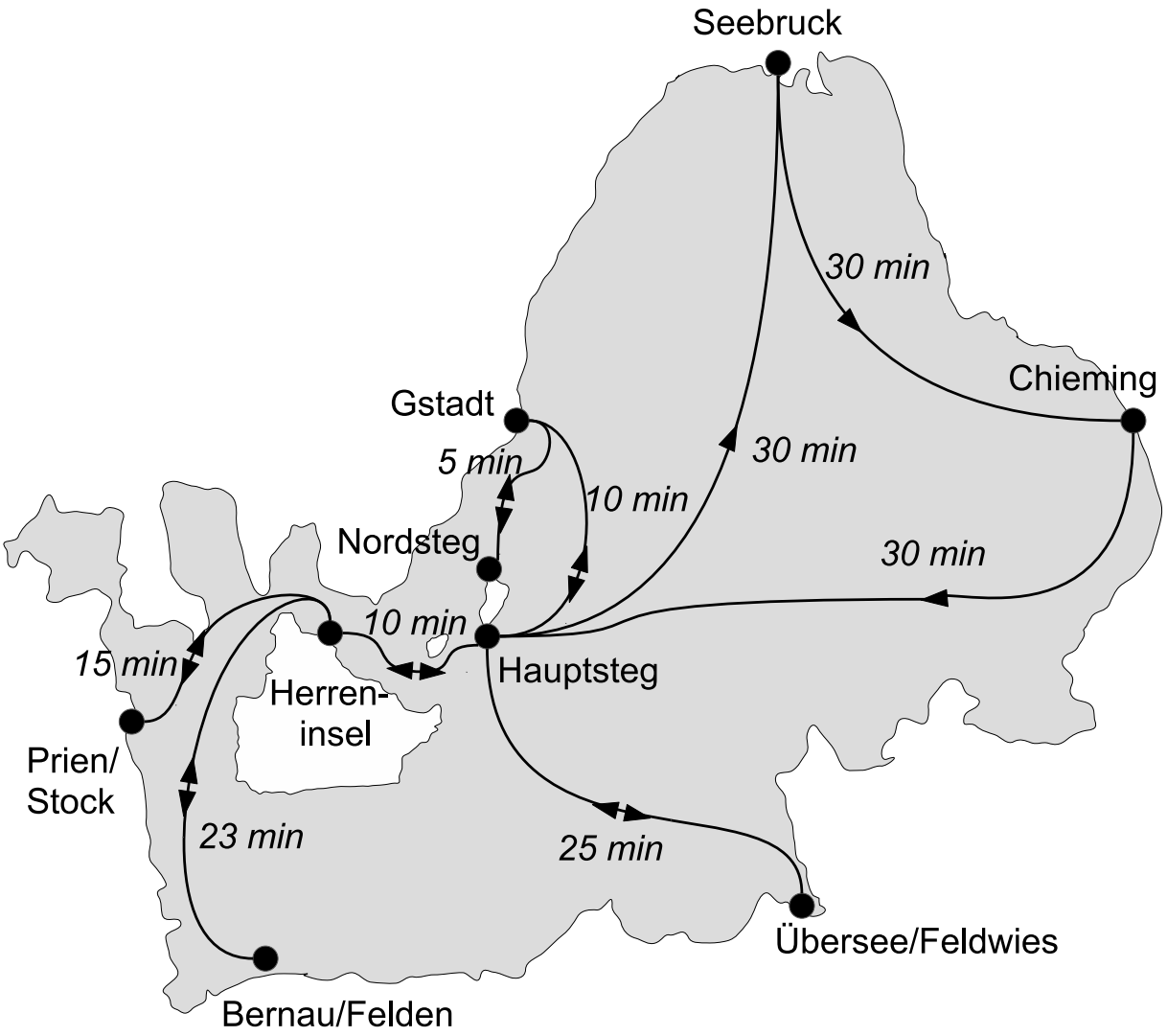
6 b) Zeichnen Sie den entstehenden geordneten Binärbaum für den Fall, dass die Daten der Mitarbeiter aus obigem Tabellenausschnitt in der angegebenen Reihenfolge in den Baum eingefügt werden. Beschränken Sie sich zur Knotenbeschriftung auf die Personalnummern.
Geben Sie außerdem an, in welcher Reihenfolge die Knoten des entstandenen Baums bei Postorder-Traversierung bearbeitet werden.

(Fortsetzung nächste Seite)

BE
4

c) Baut man den geordneten Binärbaum auf, indem man nacheinander die nach dem Primärschlüssel sortierten Datensätze der Datenbanktabelle einfügt, kommt es zu einem unerwünschten Effekt. Beschreiben Sie diesen und geben Sie eine mögliche Vorgehensweise an, dieses Problem zu vermeiden.

3. Die Chiemseeschifffahrt ist lokal organisiert und damit von der Zentralverwaltung unabhängig. Auch hier ist zur Unterstützung bei Verwaltungsaufgaben ein Softwaresystem im Einsatz.
   Im Sommerfahrbetrieb ist der unten abgebildete Linienplan gültig, der die befahrenen Strecken mit ihren regulären Fahrzeiten zeigt.



(Fortsetzung nächste Seite)

BE	
5	a) Der dargestellte Plan kann als Repräsentation eines Graphen aufgefasst werden. Nennen Sie zwei wesentliche Merkmale dieses Graphen und beschreiben Sie ihn durch eine passende Adjazenzmatrix.
4	b) Die große Chiemseetour soll bis auf den Nordsteg alle acht übrigen Anlegestellen mindestens einmal erreichen, nicht auf einer Insel beginnen, zum Schluss wieder zum Ausgangshafen zurückkehren und an jedem Landungspunkt zehn Minuten Aufenthalt vorsehen. Ermitteln Sie, wie lange eine große Chiemseetour mindestens dauert, und geben Sie auch die Stationen einer solchen kürzesten Tour an.

Um Graphen wie z. B. den in Teilaufgabe 3a beschriebenen zu implementieren, soll eine Klasse LINIENNETZ entwickelt werden. Die Anlegestellen werden mithilfe einer Klasse KNOTEN verwaltet. Diese hat als Schlüsselmerkmal das Attribut *name* und eventuell weitere Attribute sowie alle zum Geben und Setzen von Attributwerten notwendigen Methoden.

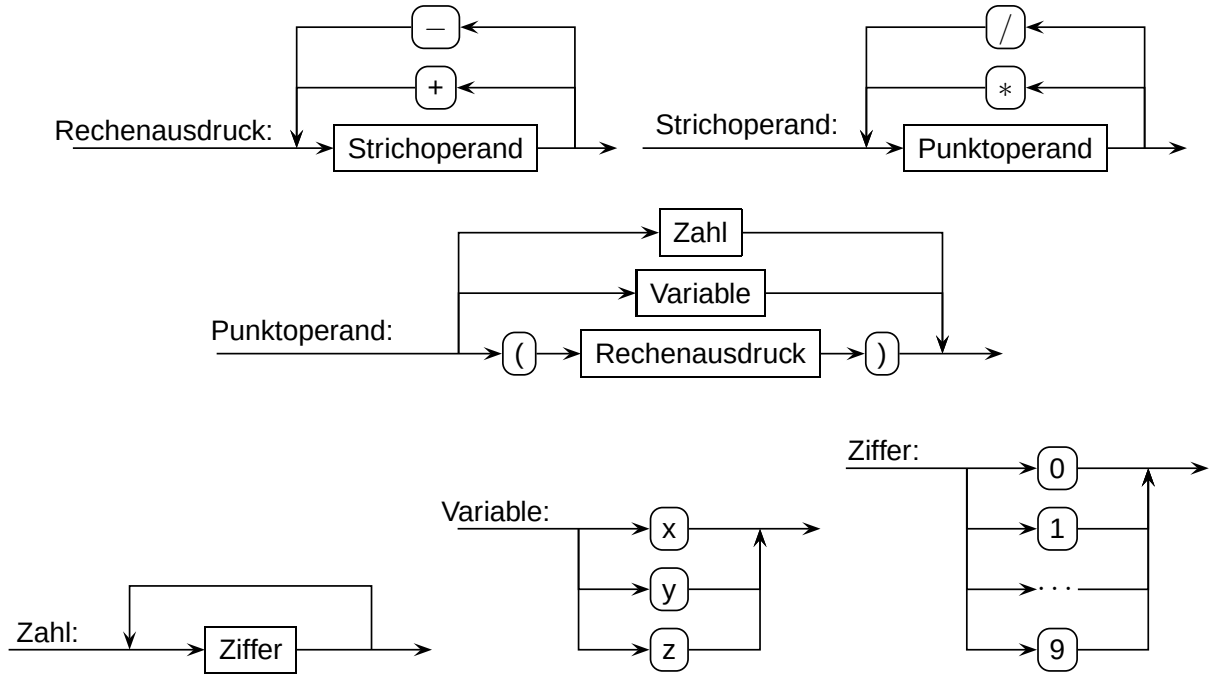
6	c) Notieren Sie in einer auf dem Deckblatt angegebenen Programmiersprache die Attribute einer Klasse LINIENNETZ, mit der ein Graph verwaltet werden kann, dessen Knoten in einem Feld und dessen Kantengewichte in einer Adjazenzmatrix gespeichert werden. Ergänzen Sie auch einen Konstruktor, sodass alle Einträge in der Adjazenzmatrix anfangs den Wert -1 haben.
5	d) Erweitern Sie Ihre Implementierung aus Teilaufgabe 3c um eine Methode *indexSuchen(KNOTEN k)*, die den Index des übergebenen Knotens im Knotenfeld zurückgibt. Achten Sie auf einen sinnvollen Rückgabewert, falls der übergebene Knoten nicht im Knotenfeld enthalten ist.
10	e) Im Gegensatz zum bisher betrachteten Graphen kann ein Graph in mehrere Teile zerfallen. Eine Methode *traversieren(KNOTEN start)* soll die Namen aller von *start* erreichbaren Anlegestellen ausgeben. Erweitern Sie Ihre Implementierung aus den Teilaufgaben 3c und 3d entsprechend. Geben Sie gegebenenfalls auch nötige Erweiterungen der Klasse KNOTEN an.
80	

III.

BE

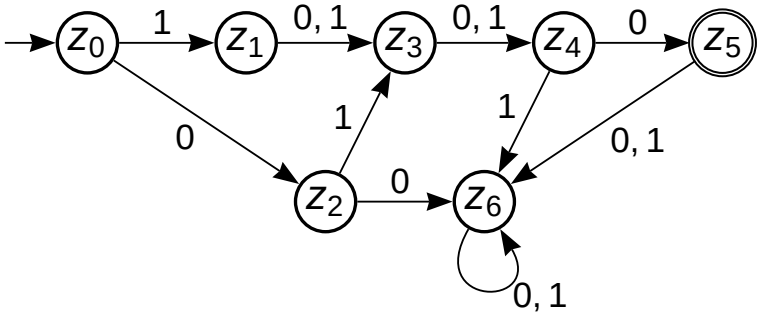
4

1. Eine Teilmenge korrekt gebildeter Rechenausdrücke wird durch das folgende Syntaxdiagramm beschrieben.



Geben Sie die Syntax der dadurch festgelegten Sprache in formaler Textnotation (z. B. in erweiterter Backus-Naur-Form) an.

2. Der abgebildete Automat erkennt spezielle 4-Bit-Binärzahlen, d. h. Zeichenketten der Länge 4, die nur aus 0 und 1 bestehen.



Hinweis: In diesem Diagramm wird ein Bit durch das Zeichen 0 oder 1 repräsentiert.

(Fortsetzung nächste Seite)

BE	
2	a) Geben Sie an, welche der folgenden Zahlen von diesem Automaten akzeptiert werden und welche nicht. (i) 0110 (ii) 1001 (iii) 0101 (iv) 1010
2	b) Beschreiben Sie die charakteristischen Eigenschaften der vom Automaten akzeptierten Zeichenketten.
6	c) Notieren Sie eine mögliche Implementierung des zuvor beschriebenen Automaten in einer auf dem Deckblatt angegebenen Programmiersprache. Dabei soll es u. a. eine Methode *binaerzahlPruefen(eingabe)* geben, die überprüft, ob die übergebene Zeichenkette zur vom Automaten erkannten Sprache gehört, und einen entsprechenden Wahrheitswert zurückgibt. Beschränken Sie sich bei der Behandlung der Zustandsübergänge vereinfachend auf den Zustand z_2. *Hinweis:* Sie dürfen folgende Methoden einer Klasse ZEICHENKETTE als bereits vollständig implementiert voraussetzen: ● *laenge()* gibt die Länge der Zeichenkette zurück, ● *zeichenAn(n)* gibt das *n*-te Zeichen der Zeichenkette zurück, wobei die Zählung bei 0 beginnt.
5	3. Bei der Analyse von Nachrichten mit möglicherweise problematischen Inhalten ist es sehr wichtig, verdächtige Zeichenfolgen zu erkennen. Es ist eine Zeichenkette zu untersuchen, die nur aus den Zeichen 0 und 1 besteht. Geben Sie das Zustandsübergangsdiagramm eines endlichen Automaten an, der erkennt, ob die Zeichenfolge 1001 in der zu untersuchenden Zeichenkette enthalten ist.

(Fortsetzung nächste Seite)

BE

4. Informationen über den Zustand einer Registermaschine werden unter anderem im Statusregister gespeichert. Für jede einzelne dieser Informationen genügt dabei ein Bit, das sogenannte Flag. Die folgende Registermaschine verfügt im Statusregister über ein Negativflag (N) und ein Zeroflag (Z). Das N-Flag bzw. das Z-Flag wird gesetzt, d. h. es erhält den Wert 1, wenn das Ergebnis der letzten Vergleichsoperation negativ bzw. null war, andernfalls wird es zurückgesetzt, d. h. es erhält den Wert 0. Der Befehlssatz der Registermaschine umfasst folgende Befehle:

load x	kopiert den Wert aus der Speicherzelle x in den Akkumulator
loadi n	kopiert die 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 Zahl n zum Wert im Akkumulator
sub x	subtrahiert den Wert aus der Speicherzelle x vom Wert im Akkumulator
subi n	subtrahiert die Zahl n vom Wert im Akkumulator
mul x	multipliziert den Wert im Akkumulator mit dem Wert aus der Speicherzelle x
muli n	multipliziert den Wert im Akkumulator mit der Zahl n
mod x	dividiert den Wert im Akkumulator durch den Wert aus der Speicherzelle x und speichert den ganzzahligen Rest dieser Division im Akkumulator
modi n	dividiert den Wert im Akkumulator durch die Zahl n und speichert den ganzzahligen Rest dieser Division im Akkumulator
div x	dividiert den Wert im Akkumulator durch den Wert aus der Speicherzelle x (ganzzahlige Division)
divi n	dividiert den Wert im Akkumulator durch die Zahl n (ganzzahlige Division)

(Fortsetzung nächste Seite)

BE	

cmp x	vergleicht den Wert im Akkumulator mit dem Wert in der Speicherzelle x. Dafür wird vom Wert im Akkumulator der Wert in der Speicherzelle x subtrahiert. Dabei werden die Flags angepasst; der Wert im Akkumulator bleibt unverändert.
cmpi n	vergleicht den Wert im Akkumulator mit der Zahl n. Dafür wird vom Wert im Akkumulator die Zahl n subtrahiert. Dabei werden die Flags angepasst; der Wert im Akkumulator bleibt unverändert.
jmp x	springt zum Befehl in Speicherzelle x
jmpz x	springt zum Befehl in Speicherzelle x, falls das Z-Flag gesetzt ist
jmpnz x	springt zum Befehl in Speicherzelle x, falls das Z-Flag nicht gesetzt ist
jmpp x	springt zum Befehl in Speicherzelle x, falls weder das N- noch das Z-Flag gesetzt ist
jmpnp x	springt zum Befehl in Speicherzelle x, falls das N- oder Z-Flag gesetzt ist
jmpn x	springt zum Befehl in Speicherzelle x, falls das N-Flag gesetzt ist
jmpnn x	springt zum Befehl in Speicherzelle x, falls das N-Flag nicht gesetzt ist
hold	beendet die Abarbeitung des Programms

3 a) Beschreiben Sie einen möglichen Befehlszyklus einer Registermaschine am Beispiel des Maschinenbefehls sub 100.

1 b) Nennen Sie die Besonderheit des Speichers in der Von-Neumann-Architektur.

(Fortsetzung nächste Seite)

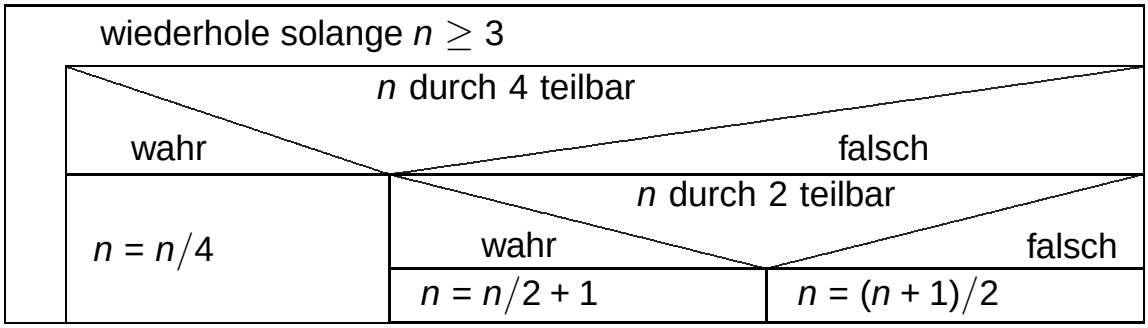
BE

4 c) Gegeben ist folgendes Programm:

```
 0:  load 100
 2:  cmp 101
 4:  jmpnn 12
 6:  load 101
 8:  store 102
10:  jmp 14
12:  store 102
14:  hold
```

Der Zustand der Registermaschine wird im Folgenden durch die Inhalte des Akkumulators A, des Befehlszählers BZ, des Statusregisters SR sowie der Speicherzellen 100 bis 102 beschrieben.
Veranschaulichen Sie die durchlaufenen Zustände bei der Ausführung des Programms anhand einer geeigneten Tabelle. Gehen Sie von folgendem Anfangszustand aus: Der Befehlszähler BZ enthält den Wert 0, die Speicherzelle 100 den Wert 4 und die Speicherzelle 101 den Wert 5.
Geben Sie an, was das Programm in Abhängigkeit von den Startwerten in den Speicherzellen 100 und 101 leistet.

Gegeben ist folgendes Struktogramm für die Methode $c(n)$ für natürliche Zahlen $n \geq 3$.

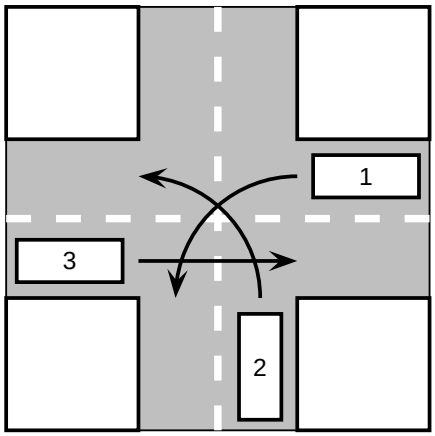


6 d) Übertragen Sie diesen Algorithmus der Methode c in ein Programm für die gegebene Registermaschine. Machen Sie auch die Speicherzelle deutlich, in der der Wert der Variablen n zu Beginn und am Ende des Programms steht.

2 e) Versehentlich wurde die Bedingung $n \geq 3$ bei der Implementierung des Algorithmus durch $n \geq 2$ ersetzt. Erläutern Sie kurz, welches Problem bei der Ausführung des Programms auftreten kann.

(Fortsetzung nächste Seite)

BE	
	5. An einer Kreuzung gleichberechtigter Straßen treffen zeitgleich drei Autos ein, die in die durch Pfeile gekennzeichneten Richtungen weiter fahren wollen.



3	a) Erläutern Sie, warum es bei genauer Befolgung der Verkehrsregeln „Rechts vor links“ und „Linksabbieger müssen den Gegenverkehr abwarten (d. h. Gegenverkehr hat Vorrang)“ zu einer Verklemmung kommt, und nennen Sie eine Möglichkeit, diese zu beheben.
2	b) Erklären Sie allgemein den Begriff „kritischer Abschnitt“ und beschreiben Sie kurz ein Konzept der Informatik, mit dem nebenläufige Prozesse synchronisiert werden können.
40	

IV.

BE

1. Die Software einer Arztpraxis ermöglicht unter anderem die Erstellung von Rechnungen für die Patienten.
   Dabei muss eine Rechnungskennzahl angegeben werden, die wie folgt aufgebaut ist:

   - zwei Buchstaben für die Initialen des Patienten (erster Buchstabe des Nachnamens gefolgt vom ersten Buchstaben des Vornamens)
   - Bindestrich
   - Patientennummer beliebiger Länge, die aus den Ziffern 0 bis 9 besteht, aber nicht mit 0 beginnt
   - Versicherungsart: P bei Privatpatienten, G bei gesetzlich Versicherten
   - nur bei gesetzlich Versicherten: zweistellige Versicherungskennzahl (Nummern im Bereich von 01 bis 12)
   - Bindestrich
   - fortlaufende Rechnungsnummer, die aus beliebig vielen Ziffern von 0 bis 9 besteht, aber nicht mit 0 beginnt

   Beispiele:
   Privatpatient Ingo Matik mit der Patientennummer 32 erhält seine 9. Rechnung. Die zugehörige Rechnungskennzahl ist: MI−32P−9.
   Seine Frau Martha Matik mit der Patientennummer 1234, die gesetzlich bei einer Versicherung mit der Kennzahl 07 versichert ist, erhält ihre 12. Rechnung. Die zugehörige Rechnungskennzahl ist: MM−1234G07−12.

   Für die Darstellung der Rechnungskennzahl stehen das lateinische Alphabet der Großbuchstaben, die Ziffern 0 bis 9 und der Bindestrich zur Verfügung.

5 a) Beschreiben Sie die syntaktisch korrekten Rechnungskennzahlen in einer einfachen Textnotation (z. B. EBNF).

6 b) Zeichnen Sie das Zustandsübergangsdiagramm eines endlichen erkennenden Automaten, der überprüft, ob eine gegebene Zeichenkette tatsächlich eine syntaktisch korrekte Rechnungskennzahl ist.

(Fortsetzung nächste Seite)

BE

2 c) Weisen Sie mithilfe Ihrer Lösung aus Teilaufgabe 1a oder 1b nach, dass die Zeichenkette WA−27P−92 eine syntaktisch korrekte Rechnungskennzahl ist.

6 d) Notieren Sie in einer auf dem Deckblatt angegebenen Programmiersprache eine Klasse KENNZAHLTESTER, die u. a. eine Methode *kennzahlTesten(kennzahl)* enthält.
Diese Methode soll gemäß Ihrem Automaten aus Teilaufgabe 1b arbeiten, d. h. insbesondere überprüfen, ob die als Zeichenkette übergebene Kennzahl eine syntaktisch korrekte Rechnungskennzahl ist, und einen entsprechenden Wahrheitswert zurückgeben.
Beschränken Sie sich bei der Behandlung der Zustandsübergänge vereinfachend auf einen Zustand, von dem aus mindestens zwei Übergänge möglich sind.
Hinweis: Sie dürfen folgende Methoden einer Klasse ZEICHENKETTE verwenden:

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

2. In vielen Ortschaften oder Siedlungen trifft man auf Rechts-vor-links-Kreuzungen.

4 a) Erklären Sie allgemein, was man in der Informatik unter einer Verklemmung versteht.
Beschreiben Sie zudem eine Verkehrssituation, die bei einer solchen Kreuzung zu einer Verklemmung führen kann.

3 b) Erklären Sie, warum eine an der Kreuzung installierte Ampelanlage die in Teilaufgabe 2a beschriebene Verklemmung verhindern kann.

(Fortsetzung nächste Seite)

BE

3. Gegeben sei eine Registermaschine mit folgendem Befehlssatz:

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

3

a) Beschreiben Sie einen möglichen Befehlszyklus einer Registermaschine am Beispiel des Maschinenbefehls sub 102.

(Fortsetzung nächste Seite)

BE
5

b) Ermitteln Sie, welche Zahl nach Durchlaufen des nachstehenden Assemblerprogramms als Ausgabe in Speicherzelle 104 steht, wenn zu Beginn in Speicherzelle 101 die Zahl 15, in Speicherzelle 102 die Zahl 23 und in Speicherzelle 103 die Zahl 7 gespeichert ist. Geben Sie dabei die nacheinander auftretenden Werte des Befehlszählers an, wenn dieser zu Beginn den Wert 1 hat.

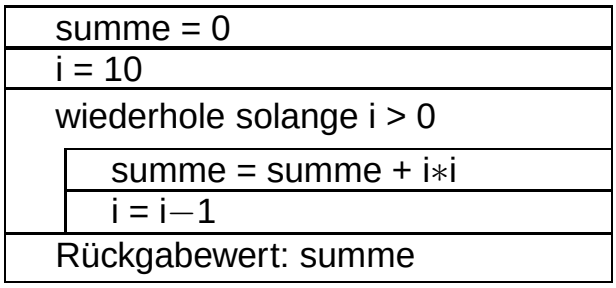
```
 1:  load 101
 2:  sub 102
 3:  jlt 10
 4:  load 101
 5:  sub 103
 6:  jlt 16
 7:  load 101
 8:  store 104
 9:  jmp 18
10:  load 102
11:  sub 103
12:  jlt 16
13:  load 102
14:  store 104
15:  jmp 18
16:  load 103
17:  store 104
18:  end
```

Beschreiben Sie zudem allgemein, welche Zahl das Programm in Speicherzelle 104 ausgibt.

(Fortsetzung nächste Seite)

BE
6
40

c) Betrachten Sie folgendes Struktogramm einer Methode *funkyFunction*:



Beschreiben Sie kurz, was diese Methode berechnet.
Schreiben Sie ein Programm für die gegebene Registermaschine, das den Algorithmus der Methode *funkyFunction* umsetzt. Geben Sie an, in welcher Speicherzelle der Rückgabewert steht.