Abiturprüfung 2013

INFORMATIK

Arbeitszeit: 180 Minuten

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

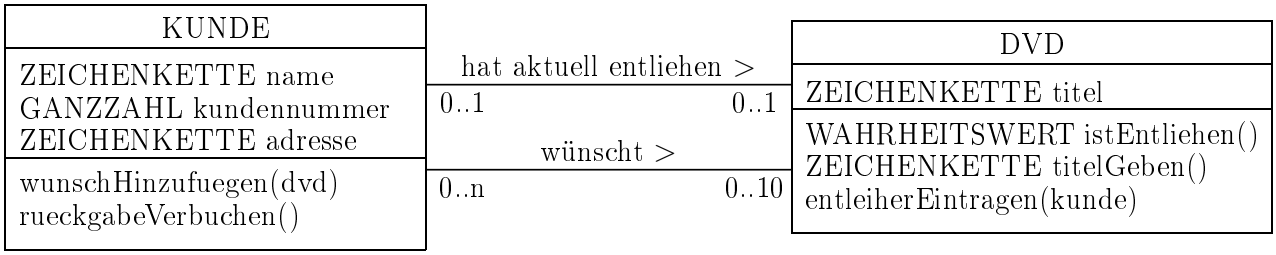
Der Fachausschuss ergänzt im folgenden Feld die erlaubten objektorientierten Programmiersprachen:

Inf1. Modellierung und Programmierung

I.

BE

Beim DVD-Verleih „Wunschkino“ sollen registrierte Kunden aus den DVDs dieses Verleihs online einen Wunschzettel mit bis zu 10 DVDs zusammenstellen und diese Auswahl speichern können. Fordert ein Kunde dann eine DVD an, wird die erste verfügbare DVD seines Wunschzettels ausgewählt und von „Wunschkino“ an ihn verschickt. Erst wenn er diese DVD zurückgesendet hat, kann er wieder eine DVD anfordern.
Die Softwarefirma IT12 ist mit der Erstellung des entsprechenden Systems beauftragt und hat dafür eine Klasse DVD und eine Klasse KUNDE gemäß dem folgenden Klassendiagramm entworfen.



Im Attribut *titel* ist der Titel der DVD gespeichert. Vereinfachend wird davon ausgegangen, dass es zu jedem Titel genau eine DVD gibt. Die Methode *istEntliehen()* gibt genau dann den Wahrheitswert *wahr* zurück, wenn die DVD aktuell entliehen ist. Die Methode *entleiherEintragen(kunde)* trägt das übergebene Kundenobjekt als aktuellen Entleiher ein und veranlasst den Versand der DVD.
Mithilfe der Klasse KUNDE wird von jedem Kunden eine ganzzahlige Kundennummer, sein Name, seine Adresse sowie gegebenenfalls die von ihm entliehene DVD gespeichert. Mit der Methode *wunschHinzufuegen(dvd)* kann der Kunde ein DVD-Objekt seinem Wunschzettel hinten hinzufügen. Ist der Wunschzettel jedoch bereits voll, wird eine entsprechende Meldung ausgegeben. Die Methode *rueckgabeVerbuchen()* aktualisiert den Datenbestand, wenn der Kunde seine entliehene DVD an „Wunschkino“ zurückgesendet hat.

(Fortsetzung nächste Seite)

BE

1. In der Klasse KUNDE wird zur Implementierung des Wunschzettels ein Feld verwendet. Sie hat zusätzlich die folgenden Methoden:

   - Mit der Methode *wunschLoeschen(i)* wird der Wunsch entfernt, der im Wunschzettel an der i-ten Stelle steht; die dahinter stehenden Wünsche werden zudem nach vorne gerückt.
   - Bei Aufruf der Methode *wunschdvdAnfordern()* wird der Wunschzettel von vorne startend durchsucht, bis eine entleihbare DVD gefunden ist. Dann wird der Kunde als Entleiher der DVD und die DVD als von diesem Kunden entliehen eingetragen sowie dieser Wunsch aus dem Wunschzettel entfernt. Ist kein Entleihvorgang möglich, da der Kunde bereits eine DVD entliehen hat oder auf seinem Wunschzettel keine entleihbare DVD gefunden wurde, wird eine entsprechende Meldung ausgegeben.

12 a) Notieren Sie in einer auf dem Deckblatt angegebenen Programmiersprache eine mögliche Implementierung der Methode *wunschdvdAnfordern()*. Alle anderen Methoden der Klasse KUNDE dürfen als implementiert vorausgesetzt werden. Geben Sie auch die Deklarationen der Attribute der Klasse KUNDE an.

10 b) Stellen Sie in einem Sequenzdiagramm die einzelnen Schritte des Methodenablaufs von *wunschdvdAnfordern()* dar. Gehen Sie beispielhaft davon aus, dass die zweite DVD auf dem Wunschzettel entliehen wird.
Geben Sie eine Problemsituation an, die sich ergeben kann, wenn bei einem weiteren Kunden ebenfalls *wunschdvdAnfordern()* aufgerufen wird, und nennen Sie eine Möglichkeit, dieses Problem zu vermeiden.

(Fortsetzung nächste Seite)

BE

2. Von jedem Kunden soll die Historie der von ihm entliehenen DVDs in einer einfach verketteten Liste gespeichert werden. Entleiht ein Kunde eine DVD mehrmals, soll sie auch mehrmals in dieser Liste enthalten sein.
Beachten Sie bei folgender Aufgabe, dass die Klasse DVD wie eingangs beschrieben als vollständig implementiert vorausgesetzt werden darf.

6 a) Zeichnen Sie ein Klassendiagramm, welches als Grundlage zur Implementierung der Historie dienen kann. Verwenden Sie dabei das Softwaremuster Kompositum und berücksichtigen Sie auch das Prinzip der Trennung von Struktur und Inhalt. Auf die Angabe von Attributen und Methoden kann verzichtet werden. Machen Sie deutlich, dass eine DVD mehrfach in der Historie stehen kann.

14 b) Notieren Sie in einer auf dem Deckblatt angegebenen Programmiersprache eine mögliche Implementierung aller betroffenen Klassen der Listenstruktur so, dass in die Liste eine DVD vorne eingefügt werden kann und dass die Titel aller gespeicherten DVDs in der Reihenfolge des Einfügens ausgegeben werden können. Wenden Sie soweit wie möglich das Prinzip der Rekursion an. Auf Attribute und Methoden, die zur Lösung nicht benötigt werden, kann verzichtet werden.

3. „Wunschkino“ will seinen Bestand an DVDs in einem alphabetisch geordneten Binärbaum speichern. Als Schlüssel dient dabei der Titel der DVD. Es darf vorausgesetzt werden, dass die Titel eindeutig sind.

5 a) Zeichnen Sie den Baum, der sich ergibt, wenn in den zunächst leeren Baum nacheinander die DVDs „Batman“, „Monaco Franze“, „High Noon“, „Fluch der Karibik“, „Cars 3“, „König der Löwen“ und „Titanic“ eingefügt werden. Geben Sie außerdem eine Einfüge-Reihenfolge an, die zu einem Baum mit möglichst wenig Ebenen führt.

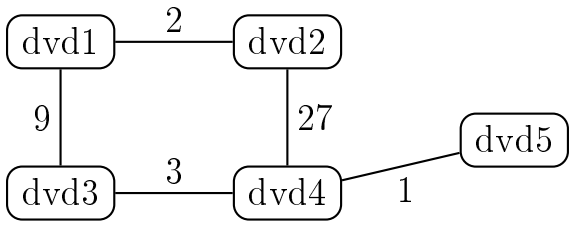
4 b) Nennen und beschreiben Sie kurz einen Algorithmus, der alle im Baum gespeicherten DVD-Titel in alphabetischer Reihenfolge ausgibt.

(Fortsetzung nächste Seite)

BE

7 c) „Wunschkino“ hat in einem solchen Baum 20 000 DVDs gespeichert. Beschreiben Sie, wie der Baum aufgebaut sein sollte, damit die Suche nach einer beliebigen DVD möglichst effizient ist. Schätzen Sie ab, wie lange eine Suche in diesem Fall maximal dauert, wenn pro untersuchtem Baumelement 1 ns nötig ist.

4. Für Marketingzwecke fasst „Wunschkino“ die Vorlieben aller seiner Kunden in einem gemeinsamen Graph mit sämtlichen angebotenen DVDs als Knoten zusammen. Gibt ein Kunde eine DVD zurück, wird dazu die Historie der bislang von diesem Kunden entliehenen DVDs ausgewertet, indem das Gewicht der Kante zwischen der zurückgegebenen DVD und jeder anderen DVD aus der Historie dieses Kunden um 1 erhöht wird. Falls diese Kante noch nicht existiert, wird eine Kante mit Kantengewicht 1 erstellt.
Ein möglicher Graph ist untenstehend abgebildet.



4 a) Stellen Sie diesen Graph als Adjazenzmatrix dar.

3 b) Zeichnen Sie den Graph, der aus dem oben stehenden Graphen entsteht, wenn ein Kunde die DVD dvd2 zurückgibt und seine Historie ansonsten aus den DVDs dvd3 und dvd4 besteht.

4 c) Nennen Sie einen Algorithmus zum Durchlaufen eines Graphen. Beschreiben Sie eine Ausleihsituation, in der bei Abarbeitung dieses Algorithmus nicht alle Knoten des Graphen besucht werden.

(Fortsetzung nächste Seite)

BE
11

d) Eine Klasse GRAPH enthält zur Speicherung der Kanten und Knoten des oben beschriebenen Graphen eine $n \times n$-Matrix a und ein Feld d der Länge n. Der Wert von n ist dabei die (als konstant angenommene) Anzahl aller DVDs von „Wunschkino“. Betrachten Sie die folgenden Methoden *m1*, *m2* und *m3* und beschreiben Sie, was der Aufruf der Methode *m3* bewirkt. Geben Sie auch den Zweck der Methode an.

```
Methode m1(GANZZAHL k)
    GANZZAHL m = 0
    wiederhole für i = 0, 1, 2, ···, n-1
        wenn (a[i][k] > m)
            m = a[i][k]
        endeWenn
    endeWiederhole
    gib m zurück
endeMethode

Methode m2(GANZZAHL k, GANZZAHL w)
    wiederhole für i = 0, 1, 2, ···, n-1
        wenn (a[i][k] ist gleich w)
            gib Titel von d[i] aus
        endeWenn
    endeWiederhole
endeMethode

Methode m3(GANZZAHL k)
    GANZZAHL u = m1(k)
    wenn (u ungleich 0)
        m2(k,u)
    endeWenn
endeMethode
```

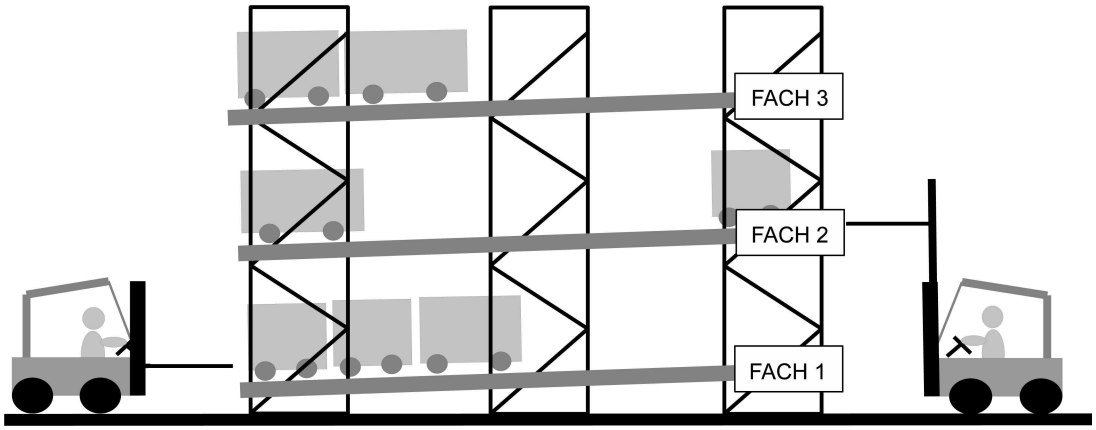
80

Inf1. Modellierung und Programmierung

II.

BE

Die effiziente Lagerung von Waren erfolgt in großen Lagerzentren oft in Hochregallagern. Wird ein Regal prinzipiell immer von der gleichen Seite mit Waren bestückt, während die Entnahme der vorher eingelagerten Waren stets von der anderen Seite erfolgt, spricht man von einem Durchlaufregal.
Das in dieser Aufgabe betrachtete Durchlaufregal besteht aus drei übereinander liegenden, zehn Meter langen Fächern mit geneigten Böden. Darin werden rollfähige Container unterschiedlicher Länge gelagert. Die Fächer sind – beginnend mit 1 – von unten nach oben durchnummeriert.



Um die Lagerhaltung zu optimieren, soll ein Programm zur Simulation des Lagers entwickelt werden.

1. Ein wichtiges Dokument im Rahmen des Softwareentwicklungsprozesses ist das Pflichtenheft.

3 a) Erklären Sie, was man unter einem Pflichtenheft versteht. Geben Sie insbesondere an, in welcher Phase des Entwicklungsprozesses dieses erstellt wird.

Bei der Implementierung wird oft auf Softwaremuster (auch Entwurfsmuster oder Design Patterns genannt) zurückgegriffen.

2 b) Erläutern Sie, was man unter einem Softwaremuster versteht.

7 c) Ein spezielles Softwaremuster ist „Model-View-Controller“ (MVC). Stellen Sie die Grundidee dieses Musters dar und nennen Sie einen konkreten Vorteil beim Einsatz dieses Musters.

(Fortsetzung nächste Seite)

BE

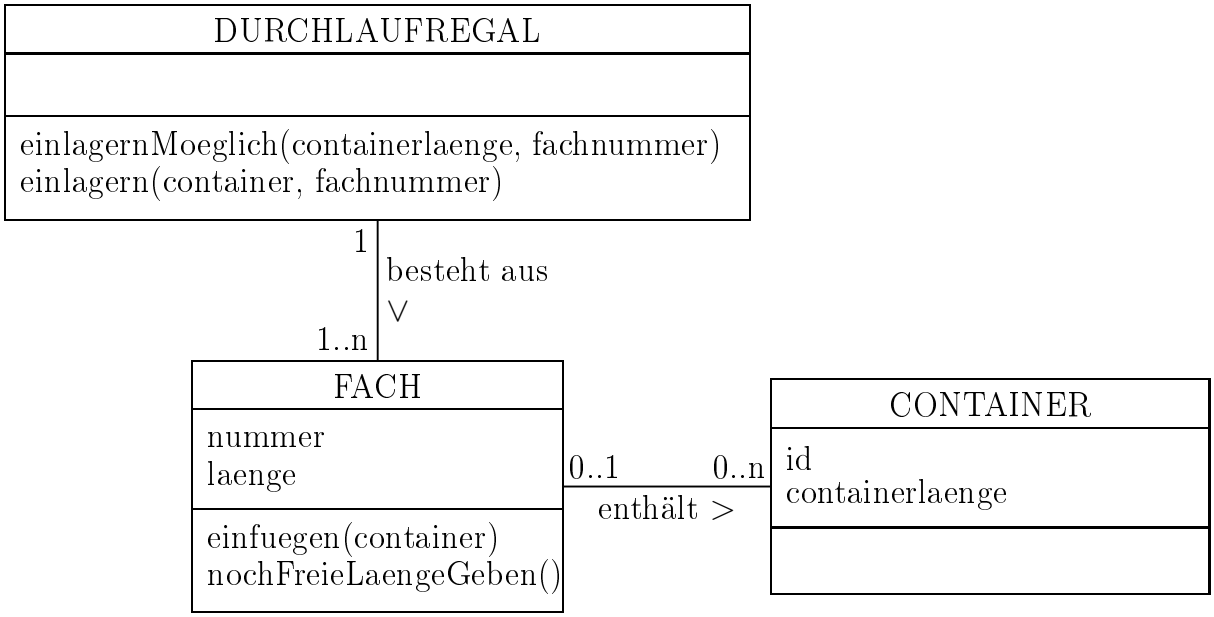
2. In Produktbeschreibungen von Hochregallager-Herstellern kann man lesen: „Die Lagerung in Durchlaufregalen erfolgt nach dem FIFO-Prinzip“. Dieses für die Warenlagerung grundlegende Prinzip geht davon aus, dass Artikel, die zuerst eingelagert werden, auch als erste verbraucht bzw. weiterverarbeitet werden (First In – First Out).

4 a) Geben Sie eine spezielle Datenstruktur an, die sich zur Implementierung eines Faches nach dem FIFO-Prinzip eignet, und beschreiben Sie kurz die grundlegenden Methoden dieser Struktur.
Hinweis: Eine Implementierung ist nicht notwendig.

3 b) Erläutern Sie an einem konkreten Beispiel, warum das FIFO-Prinzip für das Hochregal als Ganzes nicht zwingend gelten muss.

2 c) Eine andere Datenstruktur ist der Stapel. Beschreiben Sie eine Lagervariante, die mithilfe eines Stapels modelliert werden kann.

Für die Implementierung der Simulation wird das folgende Klassendiagramm zugrunde gelegt. Darin sind nur die für die weiteren Aufgaben relevanten Klassen, Attribute und Methoden aufgeführt.



Hinweis: Auf die Angabe der Standardmethoden zum Auslesen bzw. Ändern von Attributwerten wurde verzichtet. Diese können, falls im Folgenden bei der Implementierung notwendig, als bereits vorhanden betrachtet werden.

(Fortsetzung nächste Seite)

BE	
12	3. Notieren Sie in einer auf dem Deckblatt angegebenen Programmiersprache eine mögliche Implementierung für die Klasse DURCHLAUFREGAL. Orientieren Sie sich dabei an den Vorgaben des angegebenen Klassendiagramms und berücksichtigen Sie bei der Umsetzung folgende Gesichtspunkte: • Die Verwaltung der Fächer erfolgt mithilfe eines Feldes. Achten Sie insbesondere darauf, dass die Anzahl der Fächer bei der Instanzierung festgelegt werden kann. • Die Methode *einlagernMoeglich* gibt genau dann den Wahrheitswert *wahr* zurück, wenn das Fach mit der übergebenen Nummer noch einen Container der Länge *containerlaenge* aufnehmen kann. Die Fachnummer muss dabei nicht auf Gültigkeit überprüft werden. • Die Methode *einlagern* fügt das übergebene Container-Objekt in das Fach mit der ebenfalls übergebenen Nummer *fachnummer* ein. Dabei kann davon ausgegangen werden, dass für den Container noch genügend Platz vorhanden ist und *fachnummer* einen gültigen Wert hat.
4	4. Erörtern Sie anhand von zwei Aspekten, ob ein Feld zur Umsetzung der enthält-Beziehung zwischen FACH und CONTAINER geeignet ist.

(Fortsetzung nächste Seite)

BE

5. Zur Umsetzung der enthält-Beziehung zwischen FACH und CONTAINER wird eine einfach verkettete Liste verwendet. Diese basiert auf dem Softwaremuster Kompositum unter Berücksichtigung des Prinzips der Trennung von Struktur und Daten, wobei die Datenobjekte über eine definierte Schnittstelle, beispielsweise mithilfe einer abstrakten Klasse, an die Listenstruktur angekoppelt werden. Die Methoden nutzen – soweit möglich – das Prinzip der Rekursion.

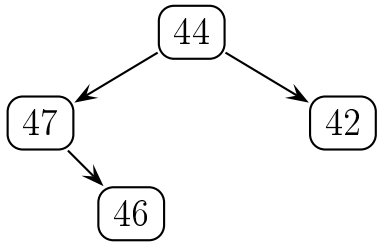
7 a) Geben Sie ein den Vorgaben entsprechendes Klassendiagramm an. Integrieren Sie darin auch die Klassen FACH und CONTAINER. Auf die Angabe von Attributen und Methoden kann verzichtet werden.

Die Methode *anzahlGeben()* soll die Anzahl der momentan in der Liste verwalteten Datenobjekte zurückgeben.

11 b) Notieren Sie – ausgehend von Ihrem Klassendiagramm aus Teilaufgabe 5a – mithilfe einer auf dem Deckblatt angegebenen Programmiersprache eine Implementierung der betroffenen Klassen der Listenstruktur. Beschränken Sie sich dabei auf die Klassen sowie Attribute und Methoden, die zur Umsetzung der Methode *anzahlGeben()* notwendig sind.

6 c) Zeigen Sie in einem Sequenzdiagramm, wie diese Anzahl mithilfe der zur Lösung in Teilaufgabe 5b eingeführten Methoden berechnet wird, wenn die Liste zwei Datenelemente verwaltet.

6. Die Verwaltung der Daten der Container, die sich derzeit im gesamten Lagerbereich befinden, erfolgt mithilfe eines geordneten Binärbaums, der mit dem Entwurfsmuster Kompositum implementiert wurde.
Aktuell werden in dem nachstehend abgebildeten Baum bereits vier Container verwaltet, die vereinfachend nur durch die ID repräsentiert sind.



(Fortsetzung nächste Seite)

BE

2 a) Geben Sie an, welches Ordnungsprinzip diesem geordneten Binärbaum zugrunde liegt.

5 b) In den Baum werden nun zwei Container mit den IDs 50 und 4 eingefügt. Zeichnen Sie den daraufhin entstandenen Binärbaum und erläutern Sie kurz, warum in diesem Fall unabhängig von der Reihenfolge des Einfügens jeweils der gleiche Baum entsteht.

7 c) Aus dem Datenbestand sollen nun – aufsteigend nach der ID sortiert – die Daten der Container aufgelistet werden, die eine Mindestlänge von 1 Meter haben. Dazu ist u. a. eine geeignete Methode *spezialdatenAusgeben()* in der Klasse KNOTEN notwendig, mit der die datenverwaltenden Baumknoten modelliert werden. Geben Sie den Algorithmus dieser Methode an. Kommentieren Sie insbesondere, welche Referenzattribute Sie dabei verwenden.

5 7. Der Bearbeitungsdurchlauf der Waren im Lagerzentrum muss in einer bestimmten Reihenfolge geschehen. Die Beschreibung der Möglichkeiten erfolgt mithilfe eines Graphen:

Knoten: 1: anliefern 2: wiegen 3: in Rollcontainer packen
4: einlagern 5: auslagern 6: umpacken
7: ausliefern

Kanten, dargestellt durch eine Adjazenzmatrix:

	1	2	3	4	5	6	7
1		x					
2			x			x	x
3				x			
4					x		
5						x	x
6							x
7							

Zeichnen Sie den Graphen.

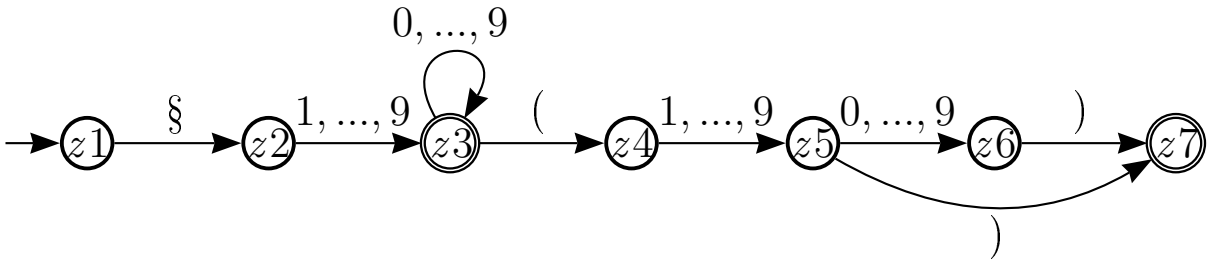
80

Inf2. Theoretische und Technische Informatik

III.

BE

1. Auf Gesetzestexte wird meist durch Angabe des Paragraphen und gegebenenfalls der Absatznummer verwiesen. Der folgende endliche Automat beschreibt eine mögliche vereinfachte Notationsform für derartige Verweise:



2 a) Geben Sie an, welche der folgenden Zeichenketten von diesem Automaten erkannt werden und welche nicht:

§012 §1109(2) §8 §31(180)

6 b) Stellen Sie die von diesem Automaten erkannte Sprache in textueller Notation (z. B. EBNF) oder mithilfe eines Syntaxdiagramms dar.

2. Bei der Suche nach bestimmten Dateien, nach E-Mails mit bestimmten Inhalten oder auch nach Schlüsselwörtern in längeren Texten kommen oft endliche Automaten zur Mustererkennung zum Einsatz.
Eine Behörde speichert alle Vorgänge in Dateien ab, deren Bezeichner nach einem bestimmten Verfahren codiert werden. Jeder Dateiname beginnt mit dem eindeutigen, aus ein bis drei Buchstaben bestehenden Kürzel des Sachbearbeiters, der den Vorgang initiiert hat, und endet mit dem eindeutigen Kürzel desjenigen, der für den Abschluss des Vorgangs verantwortlich ist. Dazwischen können beliebig viele Zeichen stehen.
Nun sollen alle Vorgänge heraus gesucht werden, die vom Sachbearbeiter mit dem Kürzel „SP“ begonnen, aber nicht von ihm abgeschlossen wurden.

8 a) Zeichnen Sie das Zustandsübergangsdiagramm eines endlichen Automaten, der genau diejenigen Zeichenketten erkennt, die mit „SP“ beginnen, aber nicht mit „SP“ enden. Als Alphabet dürfen Sie {S, P, x} verwenden, wobei x vereinfachend für alle übrigen Zeichen steht.

3 b) Zeigen Sie anhand eines Beispiels, dass der in Teilaufgabe 2a beschriebene Automat die obige Suchanfrage möglicherweise nicht korrekt bearbeitet.

(Fortsetzung nächste Seite)

BE

3. Mithilfe seiner sehr bekannten „Fibonacci-Folge“ beschrieb der Mathematiker Leonardo da Pisa 1202 das Wachstum einer Kaninchenpopulation. Die beiden ersten Zahlen der Folge sind jeweils 1. Jede weitere Zahl der Folge berechnet sich dabei als Summe ihrer beiden direkten Vorgänger: 1, 1, 2, 3, 5, 8, 13, ...
Der folgende rekursive Algorithmus liefert die n-te Zahl der Fibonacci-Folge:

```
Methode fib(n)
        wenn n = 1 oder n = 2
        dann gib zurück 1
        sonst gib zurück fib(n−1) + fib(n−2)
EndeMethode
```

3 a) Testen Sie den Algorithmus, indem Sie fib(5) als Folge von Methodenaufrufen angeben.

Wie Teilaufgabe 3a zeigt, ist die Berechnung der n-ten Fibonacci-Zahl mithilfe der rekursiven Methode aufwendig. Um den Laufzeitaufwand bei rekursiven Methoden abzuschätzen, kann man die Anzahl der jeweiligen Aufrufe betrachten. Die abgebildete Tabelle zeigt die Anzahl a(n) der Methodenaufrufe zur Berechnung von fib(n) nach obigem Algorithmus für n = 1, 2, 3, 4.

n	a(n)	$\frac{a(n)}{a(n-1)}$	Erläuterung
1	1	−−	fib(1) = 1, also ein Aufruf von fib
2	1	1,00	fib(2) = 1, also ein Aufruf von fib
3	3	3,00	fib(3) = fib(2) + fib(1) = 1 + 1, also drei Aufrufe von fib
4	5	1,67	fib(4) = fib(3) + fib(2) = ..., also fünf Aufrufe von fib

7 b) Erweitern Sie die Tabelle, indem Sie a(n) und den Quotienten aus a(n) und a(n−1) auf zwei Nachkommastellen gerundet für n = 5, 6, 7 und 8 ergänzen.
Nennen und begründen Sie damit Ihre Vermutung bezüglich des Laufzeitaufwandes des gegebenen Algorithmus zur Berechnung von fib(n).

(Fortsetzung nächste Seite)

BE

4. Die Assembler-Sprache einer Registermaschine umfasst folgende Befehle:

`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 die Speicherzelle x
`add x`	addiert den Wert in Speicherzelle x zum Wert im Akkumulator
`sub x`	subtrahiert den Wert in Speicherzelle x vom Wert im Akkumulator
`mult x`	multipliziert den Wert in Speicherzelle x mit dem Wert im Akkumulator
`div x`	dividiert den Wert im Akkumulator durch den Wert in Speicherzelle x (Ganzzahldivision)
`inc`	vergrößert den Wert im Akkumulator um 1
`dec`	verringert den Wert im Akkumulator um 1
`jump x`	springt zum Befehl in Speicherzelle x
`jge x`	springt zum Befehl in Speicherzelle x, falls der Wert im Akkumulator positiv oder Null ist
`jgt x`	springt zum Befehl in Speicherzelle x, falls der Wert im Akkumulator positiv ist
`jle x`	springt zum Befehl in Speicherzelle x, falls der Wert im Akkumulator negativ oder Null ist
`jlt x`	springt zum Befehl in Speicherzelle x, falls der Wert im Akkumulator negativ ist
`jeq x`	springt zum Befehl in Speicherzelle x, falls der Wert im Akkumulator Null ist
`end`	beendet die Abarbeitung des Programms

(Fortsetzung nächste Seite)

BE

5

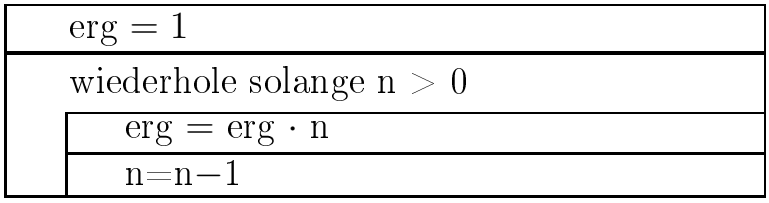
a) Vollziehen Sie das nachfolgende Assembler-Programm schrittweise nach, indem Sie angeben, welche Werte nach jedem Befehl jeweils in den Speicherzellen 101, 102 und im Akkumulator stehen, wenn zu Beginn 101 mit 5 und 102 mit 18 vorbelegt ist.

```
1:  load 102
2:  div 101
3:  mult 101
4:  sub 102
5:  jeq 8
6:  dload 0
7:  jump 9
8:  dload 1
9:  end
```

Erläutern Sie kurz, was dieses Programm allgemein (d. h. abhängig von den Ausgangswerten in den Speicherzellen 101 und 102) leistet.

6

b) Übersetzen Sie das nachfolgende Struktogramm zur Berechnung der Fakultät von n in ein Assembler-Programm. Verwenden Sie für die Variable erg die Speicherzelle 201 und für die Variable n die Speicherzelle 202.



40

Inf2. Theoretische und Technische Informatik

IV.

BE

11 1. Das „Alphabet des Lebens“ besteht aus den Buchstaben A, C, G und U. Mit Ausnahme von UGA, UAG und UAA repräsentiert jede beliebige Dreierkombination aus diesen Buchstaben eine Aminosäure. Die „Sprache des Lebens“ beschreibt Aminosäureketten nach folgenden drei Regeln:

- Jedes gültige Wort beginnt mit dem Tripel AUG (für die Aminosäure Methionin).
- Jedes gültige Wort endet mit UGA, UAG oder UAA, welche als einzige aller möglichen Dreierkombinationen keiner Aminosäure zugeordnet sind.
- Dazwischen dürfen beliebige weitere Tripel für Aminosäuren, also alle Dreierkombinationen außer UGA, UAG und UAA, eingefügt werden.

Beispielsweise sind AUGUGCAUGUCCUGUUAA oder AUGUGA korrekte Wörter.
Erstellen Sie das Zustandsdiagramm eines endlichen Automaten über dem Alphabet {A, C, G, U} zur Erkennung der „Sprache des Lebens“.

2. Mithilfe der Klasse WORTPRUEFER (vgl. nebenstehender Pseudocode) werden Zeichenketten auf Zugehörigkeit zu einer Sprache überprüft.
Die Methoden *laengeGeben()* bzw. *zeichenAnPosition(k)* der Klasse ZEICHENKETTE liefern die Anzahl der Zeichen bzw. das Zeichen an der k-ten Position der Zeichenkette, wobei die Zählung der Position bei 0 beginnt.

6 a) Die Methode *wortPruefen* eines Objekts der Klasse WORTPRUEFER wird mit dem Parameterwert AUGUGGUGGUGA aufgerufen. Geben Sie an, welche Werte das Attribut *zustand* der Reihe nach annimmt, und bestimmen Sie den Text, der von dieser Methode ausgegeben wird.

5 b) Beschreiben Sie kurz die Eigenschaften nichtleerer Zeichenketten, die als Wörter der Sprache akzeptiert werden. Geben Sie an, welche Bedeutung das Attribut x in Bezug auf das Tripel UGG hat.

(Fortsetzung nächste Seite)

BE

```
Klasse WORTPRUEFER
    die Attribute zustand und x sind jeweils vom Typ GANZZAHL

    Methode wortPruefen(ZEICHENKETTE text)
        setze die Werte für zustand und x jeweils auf 0
        wiederhole für k von 0 bis text.laengeGeben()−1
            zustandWechseln(text.zeichenAnPosition(k))
        endeWiederhole
        wenn zustand gleich 0 dann
            gib folgenden Text aus: “Das Wort wurde erkannt: “
            gib den Wert der Variablen x aus
        sonst
            gib folgenden Text aus: “Das Wort wurde nicht erkannt.“
        endeWenn
    endeMethode

    Methode zustandWechseln(ZEICHEN eingabe)
        wenn zustand gleich 0 dann
            falls eingabe gleich ‘U‘: setze zustand auf 1
            falls eingabe gleich ‘A‘, ‘C‘ oder ‘G‘: setze zustand auf 3
            andernfalls setze zustand auf 5
        sonstWenn zustand gleich 1 dann
            falls eingabe gleich ‘G‘: setze zustand auf 2
            falls eingabe gleich ‘A‘ , ‘C‘ oder ‘U‘: setze zustand auf 4
            andernfalls setze zustand auf 5
        sonstWenn zustand gleich 2 dann
            falls eingabe gleich ‘G‘: setze zustand auf 0 und erhöhe x um 1
            falls eingabe gleich ‘A‘, ‘C‘ oder ‘U‘: setze zustand auf 0
            andernfalls setze zustand auf 5
        sonstWenn zustand gleich 3 dann
            falls eingabe gleich ‘A‘, ‘C‘, ‘G‘ oder ‘U‘: setze zustand auf 4
            andernfalls setze zustand auf 5
        sonstWenn zustand gleich 4 dann
            falls eingabe gleich ‘A‘, ‘C‘, ‘G‘ oder ‘U‘: setze zustand auf 0
            andernfalls setze zustand auf 5
        endeWenn
    endeMethode

endeKlasse
```

(Fortsetzung nächste Seite)

BE

7

3. In einer Apotheke werden Aminosäureprodukte in Pulverform verkauft, die in vollständig gefüllten zylinderförmigen Dosen abgepackt sind. Aufgrund der vorgegebenen Regalhöhe haben alle ausgestellten Dosen eine Höhe von 12 cm. Der Radius der Dosengrundfläche richtet sich nach der jeweiligen Verkaufsmenge des Pulvers und wird durch folgenden Algorithmus näherungsweise berechnet:

$y = V$
$z = 1$
wiederhole solange $y > z$
$\quad y = (y + z)/2$
$\quad z = V/y$
Rückgabe $y/19$

Der dabei verwendete Wert 19 für den Divisor ergibt sich aus der vorgegebenen Dosenhöhe in mm und der Kreiszahl.

Schreiben Sie basierend auf nebenstehendem Befehlssatz ein Assembler-Programm zur Berechnung des Dosenradius (in mm) gemäß dem angegebenen Algorithmus, wobei das Volumen V in mm^3 eingegeben wird.
Ergänzen Sie dabei die begonnene Implementierung. Das Ergebnis soll am Ende in Zelle 106 stehen.

```
01:  dload 400000 -- Beispielwert für V
02:  store 101 -- V in Zelle 101
03:  dload 2
04:  store 104 -- Konstante 2 in Zelle 104
05:  dload 19
06:  store 105 -- Konstante 19 in Zelle 105
...
```

(Fortsetzung nächste Seite)

BE

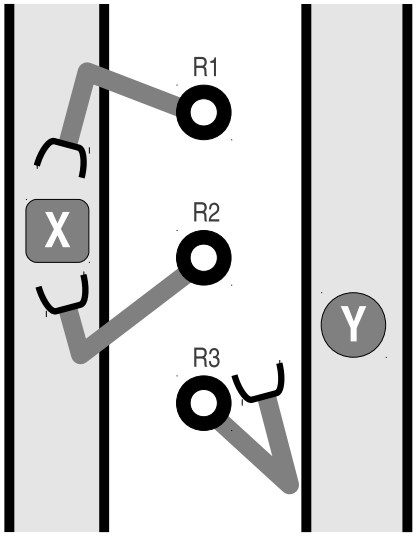
Die verwendete Maschine verfügt über folgenden Befehlssatz:

`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 die Speicherzelle x
`add x`	addiert den Wert in Speicherzelle x zum Wert im Akkumulator
`sub x`	subtrahiert den Wert in Speicherzelle x vom Wert im Akkumulator
`mult x`	multipliziert den Wert in Speicherzelle x mit dem Wert im Akkumulator
`div x`	dividiert den Wert im Akkumulator durch den Wert in Speicherzelle x
`jump x`	springt zum Befehl in Speicherzelle x
`jge x`	springt zum Befehl in Speicherzelle x, falls der Wert im Akkumulator positiv oder Null ist
`jgt x`	springt zum Befehl in Speicherzelle x, falls der Wert im Akkumulator positiv ist
`jle x`	springt zum Befehl in Speicherzelle x, falls der Wert im Akkumulator negativ oder Null ist
`jlt x`	springt zum Befehl in Speicherzelle x, falls der Wert im Akkumulator negativ ist
`jeq x`	springt zum Befehl in Speicherzelle x, falls der Wert im Akkumulator gleich Null ist
`end`	beendet die Abarbeitung des Programms

(Fortsetzung nächste Seite)

BE

4. Ein Pharmakonzern stellt in zwei Fertigungsstraßen ein Aminosäureprodukt in zwei Packungsgrößen X und Y her, wobei in jeder Straße immer nur eine der beiden Packungsgrößen hergestellt wird. Zwischen den beiden Förderbändern sind drei Schwenkroboter R1, R2 und R3 montiert, die für die Ausführung der einzelnen Arbeitsschritte eingesetzt werden.



Auf jeder Fertigungsstraße befindet sich im Bearbeitungsbereich der Schwenkroboter jeweils höchstens eine Packung.
Wenn derselbe Schwenkroboter für die Bearbeitung einer Packung in aufeinander folgenden Schritten benötigt wird, wird er aus Gründen der Effizienz nicht zwischendurch auf der anderen Fertigungsstraße eingesetzt. Die Produktionsstraße darf daher erst vom Roboter verlassen werden, wenn er für die aktuell zu bearbeitende Packung nicht mehr benötigt wird.

Die für die einzelnen Schritte der beiden Prozesse zur Herstellung der Packungsgröße X bzw. Y notwendigen Roboter sind in folgender Tabelle aufgelistet:

Prozess X zur Herstellung von Packungsgröße X:

Arbeitsschritt	X1	X2	X3
Benötigte Roboter	R1	R1 und R2	R3

Prozess Y zur Herstellung von Packungsgröße Y:

Arbeitsschritt	Y1	Y2
Benötigte Roboter	R2	R1, R2 und R3

4 a) Erläutern Sie zunächst allgemein, dann unter Bezugnahme auf das dargestellte Szenario kurz die Begriffe Synchronisation und kritischer Abschnitt.

7 b) Erläutern Sie den Begriff Verklemmung und beschreiben Sie eine Situation im dargestellten Szenario, die zur Verklemmung führen kann. Stellen Sie den sich ergebenden Ablauf in einem Sequenzdiagramm dar.

40