Abiturprüfung 2017

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

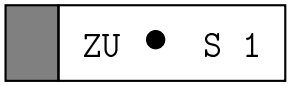
I.

BE

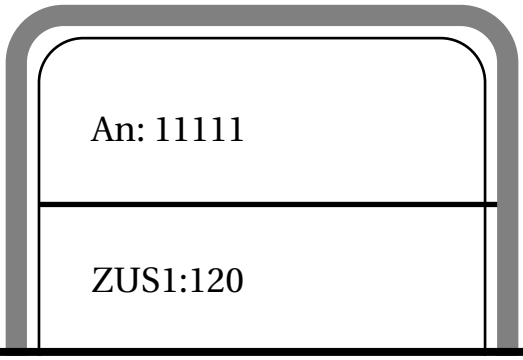
Die Stadt Zuseburg will auf dem städtischen, rund um die Uhr gebührenpflichtigen Parkplatz am Rathaus die Parkgebühren per SMS einziehen. Die Vorgehensweise wird auf einem Schild folgendermaßen erklärt:

SMS-Parken – so einfach geht's!

Erstellen Sie eine SMS mit folgendem Inhalt: KFZ-Kennzeichen ohne Leerzeichen und Bindestrich, gefolgt von einem Doppelpunkt und der Parkdauer in Minuten. Schicken Sie diese SMS an die Kurzwahlnummer 11111. Die fällige Parkgebühr wird Ihnen automatisch durch Ihren Mobilfunk-Anbieter abgebucht.



120 min



Zur Vereinfachung werden wie im Schild beschrieben Leerzeichen und Bindestrich bei den Kennzeichen weggelassen. Es kann davon ausgegangen werden, dass die so übermittelten Kennzeichen dennoch eindeutig sind und dass für jedes Kennzeichen nur ein Ticket gelöst wird.
Die erzeugten Parktickets werden innerhalb des Abrechnungssystems in einer einfach verketteten Liste verwaltet.
Die Software-Entwickler haben zur Umsetzung dieses Abrechnungssystems die Klassen PARKSYSTEM, UHR, TICKET und TICKETLISTE erstellt. Systemintern werden Datum und Uhrzeit durch die Anzahl der seit dem letzten Systemstart verstrichenen Minuten repräsentiert.

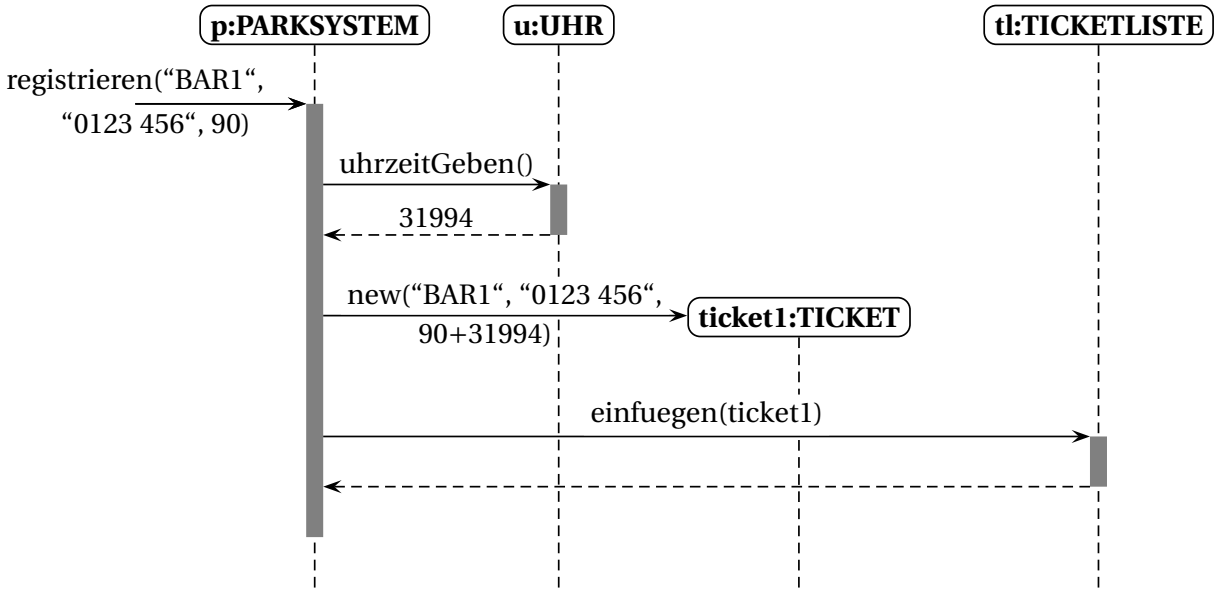
(Fortsetzung nächste Seite)

BE

5

1. Von einem Ticket sollen mithilfe der Klasse TICKET das übermittelte Kennzeichen, die Telefonnummer, mit der das Parkticket angefordert wurde, sowie das gebuchte Parkzeitende als Uhrzeit in Minuten gespeichert werden. Einem Konstruktor sollen diese drei Daten übergeben werden können.
   Außerdem soll die Klasse TICKET die Methode *restzeitGeben(uhrzeit)* besitzen, der beim Aufruf die aktuelle Uhrzeit übergeben wird. Diese Methode gibt die verbleibende Parkzeit in Minuten zurück. Falls die Parkzeit überschritten wurde, ist der Rückgabewert negativ.
   Notieren Sie in einer auf dem Deckblatt angegebenen Programmiersprache eine mögliche Implementierung der Klasse TICKET.

Der Fahrer des Fahrzeugs mit dem Kennzeichen „BA-R 1“ will 90 Minuten parken und schickt von seinem Handy mit der Telefonnummer 0123 456 eine SMS. Das folgende Sequenzdiagramm zeigt, wie das Parksystem die Daten der SMS verarbeitet.



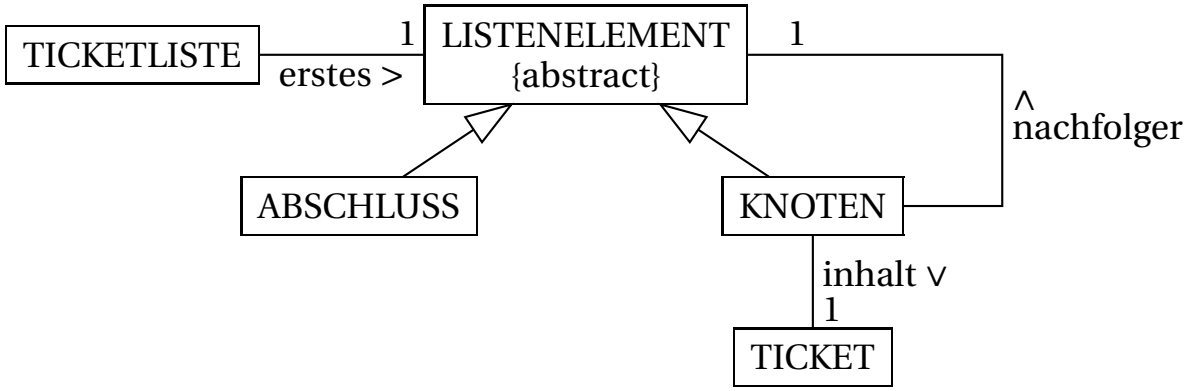
4

2. Beschreiben Sie den dargestellten Ablauf in eigenen Worten.

(Fortsetzung nächste Seite)

BE

3. Die Ticketliste ist als einfach verkettete Liste unter Verwendung des Softwaremusters Kompositum gemäß dem folgenden Klassendiagramm realisiert.



5 a) Zeichnen Sie für eine Ticketliste *tl1* ein Objektdiagramm, das drei Objekte der Klasse TICKET enthält. Geben Sie dabei an, von welcher Klasse die jeweiligen Objekte sind. Auf eine Angabe von Attributen in Objekten der Klasse TICKET darf verzichtet werden.

11 b) Die Klasse TICKETLISTE soll folgende Methoden besitzen:

- *einfuegen(ticket)* fügt das übergebene Ticket am Anfang der Liste ein.
- *suchen(kennzeichen)* gibt das TICKET-Objekt mit dem übergebenen Kennzeichen zurück; falls ein solches nicht in der Liste vorhanden ist, wird die leere Referenz zurückgegeben.

Notieren Sie in einer auf dem Deckblatt angegebenen Programmiersprache eine mögliche Implementierung dieser Methoden und aller dazu nötigen Methoden in der Listenstruktur.
Als bereits implementiert vorausgesetzt werden darf eine Methode *kennzeichenVergleichen(kennzeichen)* der Klasse TICKET, die in einem Wahrheitswert zurückgibt, ob das übergebene Kennzeichen mit dem gespeicherten übereinstimmt.

(Fortsetzung nächste Seite)

BE

Die Methode *gueltigkeitPruefen* der Klasse PARKSYSTEM ist durch folgenden Algorithmus gegeben:

```
Methode gueltigkeitPruefen(kennzeichen)
    uz = u.uhrzeitGeben()
    t = tl.suchen(kennzeichen)
    gib t.restzeitGeben(uz) zurück
endeMethode
```

Hinweis: Mit *u* bzw. *tl* wird ein Objekt der Klasse UHR bzw. TICKETLISTE referenziert. Die Methoden *suchen* und *restzeitGeben* werden in Teilaufgabe 3b bzw. Aufgabe 1 beschrieben.

8 c) Ein Kontrolleur will prüfen, ob zu dem Fahrzeug mit dem Kennzeichen „M-AD 12“ ein gültiges Ticket vorliegt. Er ist per Funk mit dem Parksystem verbunden und ruft die Methode *gueltigkeitPruefen* des Parksystems auf, wobei er die Zeichenkette MAD12 als Eingabewert übergibt. Stellen Sie den Ablauf der Gültigkeitsprüfung in einem Sequenzdiagramm dar. Gehen Sie dabei davon aus, dass das Ticket für das Kennzeichen „M-AD 12“ bereits im ersten Knoten enthalten ist.

3 d) Der oben dargestellte Algorithmus für die Methode *gueltigkeitPruefen* berücksichtigt nicht den Fall, dass für ein übergebenes Kennzeichen kein Objekt der Klasse TICKET gefunden wird. Geben Sie passende Änderungen in dieser Methode an, so dass dieser Fall sinnvoll behandelt wird.

4. In der Version 2.0 des Abrechnungssystems wird die Suche nach einem bestimmten Kennzeichen effizienter gestaltet, indem die Tickets statt in einer Liste in einem geordneten Binärbaum verwaltet werden. Geordnet wird dabei lexikographisch nach den übermittelten Kennzeichen.

4 a) Zeichnen Sie den Baum, der entsteht, wenn die folgenden Kennzeichen in der angegebenen Reihenfolge einsortiert werden:

MAD12, MZ7, HL33, KAH3, PAA1, AZ5, ROM3.

Geben Sie an, wie viele Vergleiche beim Suchen nach einem beliebigen Kennzeichen in dem entstandenen Baum höchstens nötig sind.

(Fortsetzung nächste Seite)

BE

6 b) In einem geordneten Binärbaum sind 250 Elemente gespeichert. Geben Sie begründet eine Ober- und eine Untergrenze für die Anzahl der maximal nötigen Vergleiche an, die zum Suchen eines Elements nötig sind.

2 c) Wie die Liste kann auch ein geordneter Binärbaum mithilfe des Kompositums modelliert werden. Geben Sie an, worin sich das Klassendiagramm des Baums von dem der Liste aus Aufgabe 3 wesentlich unterscheidet.

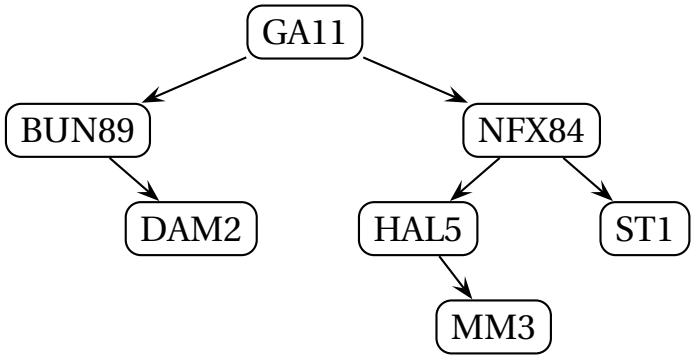
4 d) Beschreiben Sie einen Algorithmus, der die im Binärbaum gespeicherten abgelaufenen Tickets unter Verwendung des Inorder-Durchlaufs ausgibt.

5 e) Um Tickets entfernen zu können, wird in der Klasse TICKETBAUM folgender Algorithmus implementiert:

löschen(Knoten k)

Anzahl der Kindknoten von k: 0	1	2
Entferne den Knoten k.	Ersetze den Knoten k durch seinen Kindknoten.	Suche im rechten Teilbaum von k denjenigen Knoten k1, dessen Inhalt den kleinsten Schlüsselwert besitzt. Ersetze den Inhalt des Knotens k durch den Inhalt des Knotens k1. Führe den Algorithmus löschen für den Knoten k1 aus.

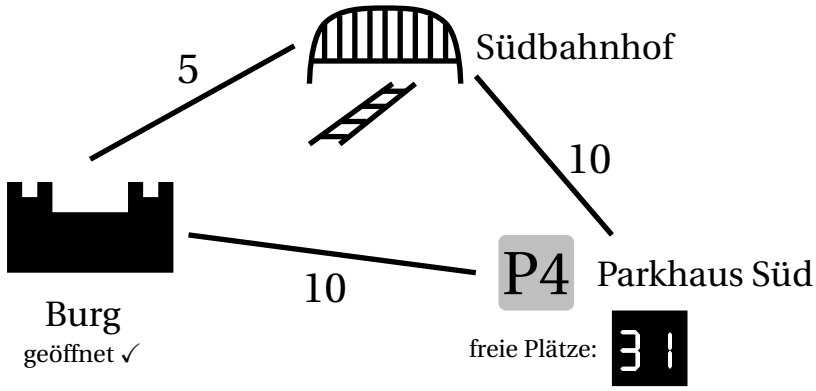
Zeichnen Sie den Baum, der sich gemäß dem oben angegebenen Algorithmus ergibt, wenn bei folgendem Baum das Ticket zum Kennzeichen GA11 gelöscht wird:



(Fortsetzung nächste Seite)

BE

5. Um möglichst viele Autofahrer in Zuseburg zum Umstieg auf Busse zu bewegen, wird an allen Parkhäusern, Bahnhöfen und Sehenswürdigkeiten eine Infotafel aufgestellt, aus der ersichtlich ist, zwischen welchen dieser Orte direkte Busverbindungen bestehen. Für jede dieser Verbindungen wird außerdem noch die Fahrzeit in Minuten angezeigt.
   Bei den Parkhäusern wird zusätzlich angegeben, wie viele freie Plätze sie im Moment haben, bei den Sehenswürdigkeiten, ob sie aktuell geöffnet sind. Die Informationen über die Fahrzeiten, die Öffnung der Sehenswürdigkeiten sowie die Zahl der freien Parkplätze werden elektronisch immer aktuell gehalten. Einen Ausschnitt einer solchen Infotafel zeigt beispielhaft nachstehende Grafik:



   Für die Fahrzeiten gilt: Vom Hauptbahnhof gelangt man in 10 Minuten zum Parkhaus West sowie in ebenfalls 10 Minuten zum Stadtmuseum. Ein weiterer Bus vom Hauptbahnhof erreicht in 3 Minuten das Rathaus. Vom Rathaus fährt ein Bus zur Burg und braucht dafür 10 Minuten. Vom Südbahnhof aus gelangt man in 5 Minuten zur Burg. Vom Parkhaus Süd erreicht man die Burg und den Südbahnhof jeweils in 10 sowie das Parkhaus West in 15 Minuten. Für die Rückfahrten gelten jeweils die gleichen Zeiten.

9 a) Stellen Sie die Informationen über diese Verbindungen als Graph dar und geben Sie die zugehörige Adjazenzmatrix an. Nennen Sie zudem zwei wesentliche Eigenschaften des Graphen.

5 b) Mit der Datenstruktur Graph sollen die Informationen über Parkhäuser, Bahnhöfe und Sehenswürdigkeiten verwaltet und angezeigt werden können. Geben Sie hierfür unter Verwendung des Prinzips von Generalisierung und Spezialisierung ein geeignetes Klassendiagramm an.

(Fortsetzung nächste Seite)

BE
9
80

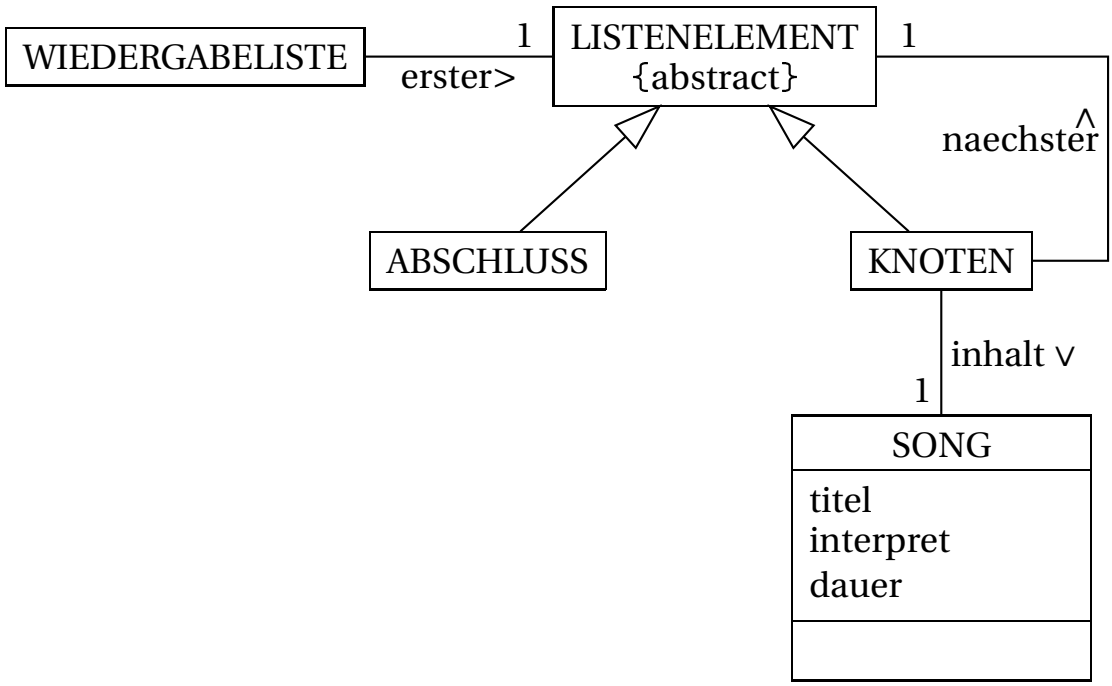
c) Im Internet sollen sich Touristen für jeden Infotafel-Standort anzeigen lassen können, welche anderen Infotafel-Standorte mit maximal einem Umsteigen erreichbar sind und wie lange die Fahrt dorthin dauert. Für das an jedem Infotafel-Standort nötige Umsteigen werden dabei fünf Minuten veranschlagt. Formulieren Sie einen Algorithmus, der zu einem Standort alle derartig erreichbaren Standorte mit den jeweiligen Fahrzeiten ausgibt.

INF1. MODELLIERUNG UND PROGRAMMIERUNG

II.

BE

Das Online-Portal Music17 bietet seinen Nutzern die Möglichkeit, über das Internet Musik anzuhören.
Für bestimmte Musikrichtungen sind dazu Wiedergabelisten angelegt. Die Songs einer Wiedergabeliste werden in einer einfach verketteten Liste gemäß dem abgebildeten Klassendiagramm verwaltet:



1. In der Wiedergabeliste *liste1* sind momentan die drei Objekte *sg1*, *sg2*, *sg3* der Klasse SONG enthalten:

	titel	interpret	dauer
sg1	Bluescreen, Bluescreen	Detlef	155
sg2	Keiner hat mich lieb	Luigi & The Kumitments	199
sg3	Hallo Welt	Detlef	230

Das Attribut *dauer* gibt die Länge des Songs in Sekunden an.

5 a) Zeichnen Sie für die Wiedergabeliste *liste1* ein Objektdiagramm gemäß obigem Klassendiagramm. Geben Sie dabei an, von welcher Klasse die jeweiligen Objekte sind. Auf eine Angabe der Attribute in Objekten der Klasse SONG kann verzichtet werden.

(Fortsetzung nächste Seite)

BE
9

b) In der Klassenstruktur der Liste werden zusätzlich die Methoden *m1* und *m2* definiert.

```
In der Klasse WIEDERGABELISTE:
    Methode m1(name)
        erster = erster.m2(name)
    endeMethode

In der Klasse LISTENELEMENT:
    abstrakte Methode LISTENELEMENT m2(name)

In der Klasse KNOTEN:
    Methode LISTENELEMENT m2(name)
        wenn inhalt.interpretGeben() ist gleich name
            gib naechster.m2(name) zurück
        sonst
            naechster = naechster.m2(name)
            gib eigene Referenz zurück
        endeWenn
    endeMethode

In der Klasse ABSCHLUSS:
    Methode LISTENELEMENT m2(name)
        gib eigene Referenz zurück
    endeMethode
```

Stellen Sie für die Situation aus Teilaufgabe 1a den durch den Methodenaufruf *liste1.m1("Detlef")* bewirkten Ablauf in einem Sequenzdiagramm dar. Berücksichtigen Sie hierbei lediglich Objekte der Klassen WIEDERGABELISTE, KNOTEN und ABSCHLUSS. Geben Sie zudem an, was dieser Aufruf bewirkt und wie die ursprüngliche Liste dadurch verändert wird.

(Fortsetzung nächste Seite)

BE

7 c) Die Methode *gesamtdauerGeben()* gibt die Gesamtdauer der in einer Wiedergabeliste enthaltenen Songs in Sekunden zurück. Notieren Sie in einer auf dem Deckblatt angegebenen Programmiersprache eine Implementierung dieser Methode in der Klasse WIEDERGABELISTE und aller dafür benötigten Methoden in den Klassen LISTENELEMENT, ABSCHLUSS und KNOTEN. Verwenden Sie so weit wie möglich das Prinzip der Rekursion.
Hinweis: Die Klasse SONG kann als bereits implementiert vorausgesetzt werden.

2. Music17 möchte seinen Nutzern ab sofort auch diverse Hitlisten anbieten. Das sind Wiedergabelisten, in denen die meistgespielten Songs sortiert abgelegt werden. Je häufiger ein Song gespielt wurde, desto weiter vorne in der Liste erscheint er. Die Klasse SONG erhält für diesen Zweck zusätzlich das Attribut *gespielt*, das angibt, wie oft der Song bereits abgespielt wurde.

15 a) Geben Sie in einer auf dem Deckblatt angegebenen Programmiersprache eine Implementierung der folgenden Methoden sowie aller dafür benötigten Methoden in den Klassen der Listenstruktur an:

- für die Klasse SONG eine Methode *istHaeufigerAls(andererSong)*, die genau dann *wahr* zurückgibt, wenn der Song, dessen Methode aufgerufen wurde, öfter gespielt wurde als der übergebene Song;
- für die Klasse WIEDERGABELISTE eine Methode *sortiertEinfuegen(song)*, die den übergebenen Song gemäß Abspielhäufigkeit in der Hitliste passend einfügt. Verwenden Sie so weit wie möglich das Prinzip der Rekursion.

(Fortsetzung nächste Seite)

BE

8 b) Um eine Top-20-Hitliste aufzubauen, startet die Anwendung mit einer leeren Liste. Sobald ein Song gespielt wurde, wird sein Attribut *gespielt* um eins erhöht und die Methode *listeAktualisieren(song)* der Hitliste aufgerufen, um sie auf den neuesten Stand zu bringen. Die Hitliste umfasst maximal 20 Songs, wobei jeder Song höchstens einmal vorkommen darf.
Neben *sortiertEinfuegen(song)* aus Teilaufgabe 2a enthält die Klasse WIEDERGABELISTE noch folgende Methoden:

anzahlGeben()	gibt die Anzahl der Songs der Wiedergabeliste zurück.
entfernen(song)	entfernt das Objekt *song* aus der Wiedergabeliste.
entfernen(i)	entfernt den Song an Position i ($i \geq 0$) aus der Wiedergabeliste.
istEnthalten(song)	gibt *wahr* zurück, falls das Objekt *song* in der Liste enthalten ist, ansonsten *falsch*.

Beschreiben Sie, wie man *listeAktualisieren(song)* nur mithilfe dieser fünf Methoden realisieren kann.

4 c) Eine Hitliste könnte auch mithilfe eines Feldes implementiert werden. Erläutern Sie anhand je eines Arguments, was für bzw. gegen die Verwendung eines Feldes spricht.

8 3. Mit einer Handy-App kann ebenfalls auf das Online-Portal zugegriffen und Musik abgespielt werden. Vereinfachend dürfen Sie annehmen, dass die App lediglich folgenden Funktionsumfang besitzt: Nach dem Aufruf ist die App bereit für Benutzereingaben. Ist bereits eine Wiedergabeliste ausgewählt, so werden nach Drücken der Starttaste alle Songs der Wiedergabeliste der Reihe nach abgespielt; andernfalls muss zuerst eine entsprechende Liste ausgewählt werden. Der Benutzer kann durch Drücken der Pause-Taste die Wiedergabe anhalten und auch wieder fortsetzen. Mit der Stopp-Taste wird ebenfalls die Wiedergabe angehalten, beim anschließenden Drücken der Start-Taste beginnt der aktuelle Song von vorne.
Stellen Sie die beschriebenen Abläufe in einem Zustandsübergangsdiagramm dar.

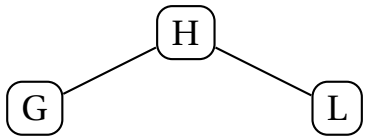
(Fortsetzung nächste Seite)

BE

4. Bei Music17 kann auch gezielt durch Eingabe des Titels nach bestimmten Songs gesucht werden. Der gesamte Songbestand wird dazu in einem nach den Titeln lexikographisch geordneten Binärbaum verwaltet.

3 a) Beschreiben Sie kurz die Eigenschaften eines derart geordneten Binärbaums.

6 b) Im vorliegenden geordneten Binärbaum sind bereits die drei Songs mit den Titeln „Heute ist nicht morgen“, „Gestern ist vorbei“ und „Lieber doch übermorgen“ gespeichert.
Hinweis: Die Titel werden zur Vereinfachung mit dem jeweiligen Anfangsbuchstaben abgekürzt.



In diesen Baum sollen weitere Songs mit folgenden Titeln in der angegebenen Reihenfolge eingefügt werden:

- Chaos im Kopf
- Am Ende wird alles gut
- Immer wieder geht‘s bergauf
- Beim nächsten Mal sehen wir uns wieder

Ergänzen Sie den abgebildeten Baum entsprechend.
Begründen Sie, warum der entstandene Baum hinsichtlich der Suche nicht optimal ist und geben Sie den entsprechend optimierten Baum an.

(Fortsetzung nächste Seite)

BE

5. Music17 wertet das Hörverhalten seiner Kunden aus und erzeugt daraus Listen, die unter der Rubrik „Andere hörten auch“ bei jedem Song abgerufen werden können. Wenn ein Hörer von einem Titel zu einem weiteren wechselt, wird dies registriert.
Die Anzahl der erfolgten Wechsel wird in einer Adjazenzmatrix gespeichert. Die Songtitel werden in der Matrix zur Vereinfachung wieder durch ihre Anfangsbuchstaben repräsentiert:

	A	B	C	G	H	I	L
A		15	25				
B	10		2	12			
C	6			5			
G						30	
H				10			
I				30			3
L				4			

Hinweis: Hier wurde z. B. zweimal von Song B zu Song C gewechselt.

6 a) Begründen Sie anhand der Adjazenzmatrix, warum der zugehörige Graph gerichtet ist, und zeichnen Sie diesen.

9 b) Für den Song „Am Ende wird alles gut“ sollen alle Titel ermittelt werden, die von ihm aus über eine oder mehrere Kanten, deren Kantengewicht jeweils mindestens 10 beträgt, erreichbar sind.
Formulieren Sie einen rekursiven Algorithmus, der dies leistet.
Geben Sie für den Startknoten A die ermittelten Titel in der durch Ihren Algorithmus festgelegten Reihenfolge an.

80

INF2. THEORETISCHE UND TECHNISCHE INFORMATIK

III.

BE

Ein auf Paketversand spezialisiertes Unternehmen hat flächendeckend an verschiedenen Orten Paketstationen aufgestellt. Die Mitarbeiter dieses Paketdienstes befüllen die Paketstationen mit quaderförmigen Paketen, die vom Empfänger rund um die Uhr abgeholt werden können.

1. Das Unternehmen gibt an, dass Sendungen mit einer maximalen Größe von 60 cm × 35 cm × 35 cm für die Paketstationen geeignet sind. Um die gültigen Größenangaben formal zu beschreiben, wird im Folgenden die Sprache L mit dem Alphabet $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, \times\}$ betrachtet, die aus Wörtern gültiger Größenangaben der Form ZZ×ZZ×ZZ besteht. Die Maße werden mit nicht negativen ganzen Zahlen in cm angegeben. Die Ziffern Z müssen dabei so gewählt sein, dass die entsprechenden Höchstmaße für Länge, Breite und Höhe des Pakets nicht überschritten werden.
   Hinweis: Zur Vereinfachung ist also beispielsweise 00×03×00 eine gültige Größenangabe.

5 a) Geben Sie eine Grammatik der Sprache L über dem Alphabet Σ an.

4 b) Geben Sie das Zustandsdiagramm eines erkennenden Automaten an, der genau die Wörter der Sprache L akzeptiert.

2 c) Weisen Sie anhand des Zustandsdiagramms aus Teilaufgabe 1b durch Angabe der durchlaufenen Zustände nach, dass die Größenangabe 47×08×15 zur Sprache L gehört.

2 d) Beschreiben Sie anhand eines konkreten Wortes der Sprache L den Unterschied zwischen Syntax und Semantik.

(Fortsetzung nächste Seite)

BE

2. Bei der Annahme eines Pakets wird dieses mit einer Auftragsnummer und einer Prüfziffer versehen. Bei der Berechnung der Prüfziffer kommt eine Registermaschine mit folgendem Befehlssatz zum Einsatz:

`dload n`	lädt die ganze Zahl n in den Akkumulator
`load x`	kopiert den Wert aus der Speicherzelle x 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
`sub x`	subtrahiert den Wert aus der Speicherzelle x vom Wert im Akkumulator
`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

Das nachstehende Programm beschreibt eine Methode zur Berechnung der Prüfziffer. Die Auftragsnummer steht zu Beginn in Speicherzelle 202, eine vorerst beliebige positive Zahl in Speicherzelle 201; der Rückgabewert steht am Ende im Akkumulator.

```
10:  load 202
11:  sub 201
12:  jlt 15
13:  store 202
14:  jump 11
15:  load 202
16:  end
```

(Fortsetzung nächste Seite)

BE

3 a) In Speicherzelle 201 steht der Wert 6, in Speicherzelle 202 der Wert 17. Notieren Sie, welche Belegungen die betroffenen Speicherzellen und der Akkumulator beim Ablauf dieses Programms schrittweise annehmen. Geben Sie jeweils auch den Befehl an, der zur jeweiligen Speicherbelegung führt.

2 b) Geben Sie allgemein die mathematische Bedeutung des Rückgabewerts der Methode in Abhängigkeit von beliebigen positiven Eingangswerten in den Speicherzellen 201 und 202 an. Nennen Sie die möglichen Prüfziffern, wenn der Eingangswert in Speicherzelle 201 den Wert 6 hat.

4 c) Beschreiben Sie die Methode durch eine geeignete graphische Darstellung, z. B. durch ein Struktogramm.

3 d) Nicht positive Eingangswerte in Speicherzelle 201 sind für das Programm ungeeignet. Ergänzen Sie ohne Veränderung der Befehle in den Zellen 10 bis 16 das vorliegende Programm so, dass in diesen Fällen am Ende des Programms der Fehlerwert „−1“ im Akkumulator steht, während das Programm in allen anderen Fällen weiterhin wie vorgegeben arbeitet.

3. Ein Online-Shop für Bücher nutzt zur Auslieferung der bestellten Ware an seine Kunden den Paketdienst: Der Kunde bestellt ein Buch, der Online-Shop verpackt das Buch in ein Paket zulässiger Größe und übergibt es einem Mitarbeiter des Paketdienstes, der es in das nächstgelegene Verteilerzentrum bringt. Dort werden ankommende Pakete entsprechend ihrer Zielbestimmung sortiert und schließlich an das dem Zielort nächstgelegene Verteilerzentrum transportiert. Von dort wird das Paket durch einen weiteren Mitarbeiter des Paketdienstes in der betreffenden Paketstation deponiert, von der es der Kunde abholen kann.

5 a) Stellen Sie das gegebene Szenario in einem Schichtenmodell mit wenigstens drei Schichten dar. Machen Sie für jede Schicht deutlich, welche Aufgabe sie hat.

2 b) Beschreiben Sie einen Vorteil der Aufteilung eines Vorgangs in Schichten.

(Fortsetzung nächste Seite)

BE	
4	4. Der Paketdienst bietet die Möglichkeit, dass auch ein Privatkunde fertig adressierte und frankierte Pakete zulässiger Größe in der Paketstation ablegen kann, die dann von einem Mitarbeiter abgeholt werden. Hierzu muss der Kunde im Voraus ein Ablagefach der Paketstation online reservieren. Auf dem hierfür verwendeten Server des Paketdienstes wird dies durch folgenden Algorithmus abgewickelt: falls in der Paketstation noch ein Ablagefach frei ist nimm die Reservierung an verringere die Zahl der freien Ablagefächer um eins weise ein freies Ablagefach zu endefalls Die Anzahl der reservierbaren Ablagefächer in einer konkreten Paketstation beträgt 15. Erläutern Sie, wie es geschehen kann, dass der Server mehr als 15 Reservierungen gleichzeitig gespeichert hat, und stellen Sie eine Maßnahme dar, dies zu verhindern.
4	5. Zur Nutzung der Paketstation muss sich ein Kunde registrieren und dabei ein aus fünf Ziffern bestehendes Passwort festlegen. Ein Unbefugter verwendet ein Brute-Force-Verfahren, um sich Zugang zum Konto eines Kunden zu verschaffen. Dabei benötigt er pro Anmeldeversuch 0,2 Millisekunden. Schätzen Sie in diesem Zusammenhang die Zeit ab, die der Unbefugte benötigt, um das Passwort herauszufinden. Nennen Sie zwei Maßnahmen, die der Paketdienst ergreifen könnte, um das Herausfinden eines Passwortes durch einen Brute-Force-Angriff zu erschweren.
40	

INF2. THEORETISCHE UND TECHNISCHE INFORMATIK

IV.

BE

1. Eine einfache Möglichkeit zur Erzeugung von Grafiken ist die sogenannte Turtle-Grafik, bei der man sich vorstellen kann, dass sich eine Turtle (engl. für Schildkröte), die mit einem Stift ausgestattet ist, auf einer Zeichenfläche bewegt; dabei hinterlässt sie eine Spur, falls der Stift abgesenkt ist. Die Turtle kann mit folgenden einfachen Befehlen gesteuert werden:

Vx	bewegt die Turtle um x Einheiten nach vorne
Zx	bewegt die Turtle um x Einheiten zurück
Rx	dreht die Turtle um x Grad im Uhrzeigersinn
Lx	dreht die Turtle um x Grad gegen den Uhrzeigersinn
U	senkt den Stift der Turtle ab (nach unten)
O	hebt den Stift der Turtle an (nach oben)
Wx(Befehle)	wiederholt x-mal den angegebenen Befehl bzw. die durch Komma getrennten Befehle in der Klammer

Dabei ist x eine ganze Zahl größer 0.

Ein Programm für die Turtle besteht aus mindestens einem Befehl; aufeinanderfolgende Befehle werden durch Komma getrennt. Ein Programm endet mit einem Ausrufezeichen.

3 a) Fertigen Sie eine Zeichnung für eine grafische Ausgabe an, die durch das nachfolgende Programm erzeugt wird. Die Einheit soll dabei 1 mm sein. Kennzeichnen Sie in Ihrer Zeichnung den Startpunkt und die Startrichtung der Turtle.

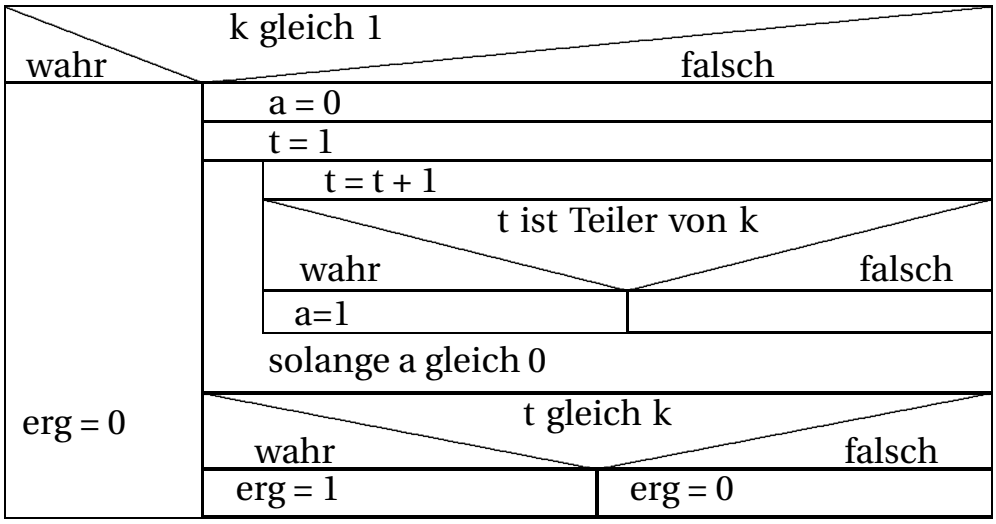
W4(U,V30,L45,O,V30,L45)!

7 b) Geben Sie ein geeignetes Alphabet und die Produktionsregeln für die Turtle-Sprache in einfacher Textnotation (z. B. EBNF) an.

(Fortsetzung nächste Seite)

BE

2. In dem folgenden Struktogramm wird ein Algorithmus dargestellt, der erkennt, ob eine natürliche Zahl k eine Primzahl ist. In diesem Fall wird in die Speicherzelle *erg* die Zahl 1 abgelegt, sonst 0.



3 a) Stellen Sie die Veränderung der Variablenwerte bei Ablauf dieses Algorithmus jeweils für die Startwerte k = 5 und k = 15 durch zwei Speicherbelegungstabellen wie nachfolgend gezeigt dar.

Anweisung	k	a	t	erg
	5			
a = 0		0		
t = 1			1	
t = t + 1			2	
...	...	...	...	...

(Fortsetzung nächste Seite)

BE

Gegeben ist eine Registermaschine mit folgendem Befehlssatz:

`ldai n`	lädt die ganze Zahl n in den Akkumulator
`lda x`	kopiert den Wert aus der Speicherzelle x in den Akkumulator
`sta x`	kopiert den Wert aus dem Akkumulator in die Speicherzelle x
`inc`	erhöht den Wert im Akkumulator um eins
`dec`	erniedrigt den Wert im Akkumulator um eins
`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
`mod x`	speichert den Rest bei der Ganzzahldivision des Akkumulatorwerts durch den Wert aus der Speicherzelle x in den Akkumulator
`jmp x`	springt zum Befehl in Speicherzelle x
`jaz x`	springt zum Befehl in Speicherzelle x, falls der Wert im Akkumulator Null ist
`jap x`	springt zum Befehl in Speicherzelle x, falls der Wert im Akkumulator positiv ist
`jan x`	springt zum Befehl in Speicherzelle x, falls der Wert im Akkumulator negativ ist
`end`	beendet die Abarbeitung des Programms

Im Folgenden soll ein Programm für diese Maschine erstellt werden, das den dargestellten Algorithmus umsetzt. Der Wert von k soll in Speicherzelle 101, der von a in 102, der von t in 103 und der von *erg* in 104 gespeichert werden.

2

b) Betrachten Sie die folgende kurze Sequenz; *xx* steht dabei für ein geeignetes Sprungziel.

```
lda   101
mod   103
jap   xx
ldai  1
sta   102
```

Geben Sie an, welcher Teil des Algorithmus damit umgesetzt wird.

10

c) Setzen Sie unter Verwendung der Sequenz aus Teilaufgabe 2b den gesamten Algorithmus in ein Programm für die gegebene Registermaschine um.

(Fortsetzung nächste Seite)

BE

3. Die sogenannte russische Multiplikation ist eine Möglichkeit, das Produkt zweier beliebiger positiver ganzer Zahlen zu berechnen.
Der folgende rekursive Algorithmus zeigt eine mögliche Implementierung dieser Multiplikationsvariante:

```
Methode mult(a,b)
    wenn b gleich 1
        gib a zurück
    sonst
        wenn b gerade
            gib mult(a∗2, b/2) zurück
        sonst
            gib a + mult(a∗2, b/2) zurück
        endeWenn
    endeWenn
endeMethode
```

Hinweise: Der Operator „/“ steht für die ganzzahlige Division. 8/3 ergibt also beispielsweise 2. Der Operator „∗“ steht für die Multiplikation.

3 a) Zeigen Sie, dass der Algorithmus für a = 4 und b = 10 das richtige Ergebnis liefert, indem Sie mult(4,10) als Folge von Methodenaufrufen angeben.

2 b) Erläutern Sie kurz, welche Folge der eigentlich nicht vorgesehene Aufruf von mult(2017,0) hätte.

(Fortsetzung nächste Seite)

BE

7

c) Die folgende Tabelle zeigt die Anzahl N der nach dem Aufruf von mult(2,b) zusätzlich erforderlichen rekursiven Aufrufe:

b	1	2	4	5	6	8	16	32
N	0	1	2	2	2	3	4	5

Beim Aufruf von mult(2,2) ist also neben dem ersten Aufruf der Methode noch ein rekursiver Aufruf nötig, beim Aufruf von mult(2,4) sind es zwei.

i) Geben Sie die Anzahl der rekursiven Aufrufe für mult(2,3) und mult(2,1024) an.

ii) Schließen Sie anhand der Wertetabelle auf das Laufzeitverhalten für mult(2,b). Begründen Sie dieses Laufzeitverhalten anschließend anhand des gegebenen Algorithmus. Erläutern Sie zudem, warum es beim Aufruf von mult(a,b) auch für a > 2 zum gleichen Laufzeitverhalten kommt.

3

d) Erklären Sie, warum der Aufruf der Methode mult(a,b) für bestimmte Anfangswerte wie a = 2, b = 2017 zu unnötig langen Aufrufsequenzen führt. Beschreiben Sie eine Möglichkeit, wie man die Methode verbessern könnte, damit sie in solchen Fällen effizienter arbeitet.

40