Hinweise zur Bearbeitung der Klausur zum Kurs 1661 Datenstrukturen I

Bitte lesen Sie sich diese Hinweise vollständig und aufmerksam durch, bevor Sie mit der Bearbeitung der Klausur beginnen. Beachten Sie insbesondere die Punkte 7 und 8!

1. Die Klausurdauer beträgt 2 Stunden.
2. Prüfen Sie bitte die Vollständigkeit Ihrer Unterlagen. Die Klausur umfasst:
   - 2 Deckblätter
   - diese Hinweise
   - 1 Formblatt für eine Teilnahmebescheinigung zur Vorlage beim Finanzamt
   - 5 Aufgaben auf den Seiten 2–4
3. Bevor Sie mit der Bearbeitung der Klausuraufgaben beginnen, füllen Sie bitte die folgenden Teile der Klausur aus:
   (a) sämtliche Deckblätter mit Name, Anschrift sowie Matrikelnummer. Markieren Sie vor der Abgabe auf allen Deckblättern die von Ihnen bearbeiteten Aufgaben.
   (b) die Teilnahmebescheinigung, falls Sie diese wünschen.
4. Schreiben Sie Ihre Lösungen auf Ihr eigenes Papier (DIN A4) und nicht auf die Seiten mit den Aufgabenstellungen. Heften Sie vor Abgabe der Klausur die Deckblätter (und evtl. die Teilnahmebescheinigung) an Ihre Bearbeitung.
5. Schreiben Sie bitte auf jedes Blatt oben links Ihren Namen und oben rechts Ihre Matrikelnummer. Numerieren Sie Ihre Seiten bitte durch.
6. Wenden Sie bei der Lösung der Aufgaben, soweit dies möglich ist, die Algorithmen und Notationen aus den Kurseinheiten an.
7. Schreiben Sie, wenn Algorithmen gefordert sind, keine kompletten PASCAL- oder JAVA-Programme, sondern beschränken Sie sich auf die wesentlichen Teile des Algorithmus, die z.T. auch in Prosa formuliert werden können. Formulieren Sie Algorithmen aber so, dass die elementaren Einzelschritte erkennbar werden.

   Sparen Sie bei solchen Aufgaben nicht mit Kommentaren. Wenn Ihre Lösung aufgrund fehlender Kommentare nicht verständlich ist, führt das zu Punktabzug.
8. Vermeiden Sie in jedem Fall bei der Definition von Funktionen in Algebren PASCAL- oder JAVA-Programme sowie Algorithmen. Geben Sie lediglich mathematische Definitionen an wie sie im Kurstext verwandt worden sind!

   Ebenso sind Mengendefinitionen in mathematischer Mengennotation durchzuführen. Geben Sie auf keinen Fall Datentypdefinitionen an, und verwenden Sie keine konkreten Datenstrukturen!
9. Als Hilfsmittel sind nur unbeschriebenes Konzeptpapier und Schreibzeug zugelassen. Die Reinschrift der Klausur darf **nicht mit Bleistift** erfolgen.
10. Es sind maximal 66 Punkte erreichbar. Zur Erlangung eines Übungsscheins bzw. Zertifikats benötigen Sie mindestens 33 Punkte.

16 Punkte

Aufgabe 1 Playlist-Algebra

Von einer Künstleragentur erhalten Sie den Auftrag, eine Software zur Unterstützung von DJ- and Musikerauftritten zu erstellen. Als Komponente dieser Software ist eine sogenannte „Playlistverwaltung" vorgesehen. Eine Playlist entspricht einer Folge von Musikstücken, die während eines Künstlerauftritts dargeboten werden.

Zur Unterstützung der Konzertplanung sollen für ein einzelnes Stück (*Track*) folgende Informationen verwaltet werden: Name des Stückes („*title*"), dessen Länge („*length*", in Millisekunden) sowie der Name des vortragenden Interpreten („*interpret*"). Damit nicht nur Interpreten, sondern auch Komponisten und Texter (resp. die entsprechenden Rechteinhaber) ihren Anteil an den Konzerteinnahmen erhalten, müssen Konzertveranstalter solche Playlisten ggf. an die GEMA (Gesellschaft für musikalische Aufführungs- und mechanische Vervielfältigungsrechte) melden. Daher werden zu jedem Stück zusätzlich benötigt: der Komponist des Stückes („*composer*"), der Autor des Textes („*author*"), eine Angabe, ob das Stück von der GEMA verwaltet wird („*licence*"; mögliche Werte: GEMA, free, special, unspecified). Es ist erlaubt, Felder unspezifiziert zu lassen (leerer String, length = 0 bzw. *licence* = unspecified).

Mögliche Operationen auf einem Track sind

createTrack, um ein Track-Objekt mit Hilfe aller oben genannter Angaben zu erstellen,

getTitle, *getLength*, *getInterpret*, *getComposer*, *getAuthor*, *getLicence* um die oben beschriebenen Felder auszulesen.

Eine *Playlist* ist eine Abfolge von Tracks. Auf diese Tracks kann man über ihre Position in der Abfolge zugreifen. Positionsnummern sollen automatisch verwaltet werden, d.h. beim Einfügen, Löschen und Verschieben von Tracks soll die Numerierung nachfolgender Tracks automatisch aktualisiert werden. Auf einer Playlist sollen folgende Operationen unterstützt werden:

empty erzeugt eine leere Playlist;

getTrackByPos liefert den Track an der gegebenen Position;

getLastPos gibt die Position des letzten enthaltenen Tracks zurück;

getTotalLength liefert die Gesamtspieldauer in Millisekunden;

insertTrackAtPos fügt einen Track so in die Playlist ein, dass er die angebene Position erhält;

deleteTrackAtPos löscht den Track an der angegebenen Position,

editTrackAtPos erlaubt es, einen durch eine Position bezeichneten Track in der Playlist durch einen übergebenen Track zu ersetzen;

findTrack liefert die Position des ersten Tracks, der nach einer übergebenen Position in der Playlist steht und einem übergebenen Track entspricht. Ein Track entspricht einem anderen, wenn alle Felder, die in beiden gleichermaßen spezifiziert sind, übereinstimmen. Einer- oder beiderseits unspezifizierte Felder müssen dazu nicht übereinstimmen.

Formulieren Sie die Spezifikation aller erforderlichen Datentypen und Operationen als Algebra.

3 Punkte

(a) Geben Sie dabei die Sorten und Operatorsignaturen der Algebra an.

(b) Geben Sie dabei die Trägermengen für die eingeführten Sorten an. *5 Punkte*

(c) Geben Sie dabei die Semantik der eingeführten Operationen an. *8 Punkte*

Aufgabe 2 Hashing **9 Punkte**

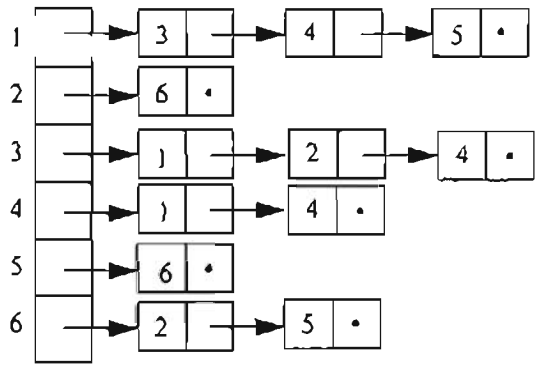
Fügen Sie die Namen Anne, Christian, Frank, Heidrun, Jianqiu, Mahmoud, Markus, Simone und Thomas in dieser Reihenfolge in eine Hashtabelle mit m=10 ein. Benutzen Sie dabei die Divisionsmethode als Hashfunktion und die Grundform des quadratischen Sondierens (nur Addition). Den Namen wird zur Berechnung der Hashfunktion als Wert jeweils die Summe der Buchstabenwerte ihrer ersten 3 Zeichen zugeordnet, wobei A mit 1, B mit 2, ..., Z mit 26 bewertet ist. Geben Sie für jeden eingefügten Namen die Folge der Behälternummern an, die getestet wird. Wie lautet der Inhalt der finalen Hashtabelle?

Aufgabe 3 Graphen **10 Punkte**

Analysieren Sie im folgenden die Graphen G1 und G2. Wenn Sie die Wahl haben, fahren Sie mit der Bearbeitung beim Knoten mit der kleinsten Nummer fort.

	1	2	3	4	5	6
1	∞	9	∞	4	∞	5
2	9	∞	3	∞	∞	∞
3	∞	3	5	∞	6	7
4	4	∞	∞	∞	9	∞
5	∞	∞	6	9	∞	∞
6	5	∞	8	∞	∞	∞

Adjazenzmatrix G1



Adjazenzliste G2

(a) Könnten die Graphen als ungerichtete Graphen aufgefaßt werden? Wenn ja, warum? *2 Punkte*

(b) Enthält G2 Zyklen? Wenn ja, geben Sie diese bitte an. *2 Punkte*

(c) Geben Sie die stark verbundenen Komponenten von G1 und G2 an. *2 Punkte*

(d) Geben Sie folgende Spannbäume an: *4 Punkte*

- Tiefendurchlauf durch G1 ausgehend von Knoten 1.
- Breitendurchlauf durch G1 ausgehend von Knoten 1.
- Tiefendurchlauf durch G2 ausgehend von Knoten 2.
- Breitendurchlauf durch G2 ausgehend von Knoten 2.

16 Punkte

Aufgabe 4 (Kürzeste Wege in Graphen)

Gegeben sei die folgende Kostenmatrix eines Graphen G:

	A	B	C	D	E	F	G	H
A	∞	∞	2	8	∞	∞	∞	∞
B	∞	∞	∞	2	4	∞	∞	∞
C	∞	3	∞	∞	∞	4	∞	∞
D	∞	∞	∞	∞	1	∞	∞	∞
E	∞	∞	∞	∞	∞	∞	3	7
F	∞	∞	∞	∞	∞	∞	6	∞
G	∞	∞	1	∞	∞	∞	∞	1
H	∞	∞	∞	∞	∞	∞	∞	∞

2 Punkte (a) Stellen Sie G graphisch dar.

14 Punkte (b) Finden Sie mit Hilfe des Algorithmus von Dijkstra die kürzesten Wege von Knoten A zu allen anderen Knoten. Geben Sie die Zwischenschritte an. Verwenden Sie dabei eine graphische Darstellung für G. Markieren Sie die verschiedenen Zustände (Farben) von Knoten und Kanten.

15 Punkte

Aufgabe 5 B-Baum

Gegeben sei eine sortierte Folge s_i mit N Schlüsselelementen. Überlegen Sie sich eine Vorgehensweise, mit der Sie einen B-Baum der Ordnung m mit einem vorgegebenen Knotenfüllgrad von p Prozent erzeugen können. Da in einem Knoten mindestens m und höchstens $2m$ Elemente sein dürfen, gilt $50 \leq p \leq 100$. Alle bzw. möglichst viele Knoten unterhalb der Wurzel sollen den Füllgrad p erfüllen und natürlich trotzdem den Struktureigenschaften des B-Baumes genügen.

Hinweis: Sie müssen hier keinen Algorithmus notieren. Es ist vorteilhaft, den Baum von den Blättern ausgehend weiter nach oben aufzubauen (bottom-up).