Fachbereich
Informatik

Klausur Einführung in die Informatik für Elektrotechniker II
17. Juli 1999

Name: .....

Matr.-Nr. .....

Bearbeitungszeit: 120 Minuten

Bewertung

(bitte offenlassen :–)

Aufgabe	Punkte	Erreichte Punkte
1	8	
2	4	
3	12	
4	14	
5	6	
Summe	44	

Spielregeln (**Jetzt lesen!**):

- Benutzen Sie für die Lösung der Aufgaben **nur** das mit diesem Deckblatt ausgeteilte Papier. Lösungen, die auf anderem Papier geschrieben werden, können **nicht** bewertet werden. Schreiben Sie ihre Lösung auch auf die Rückseiten der Blätter; benötigen Sie für eine Lösung mehr als ein Blatt, finden Sie am Ende der Klausur Leerblätter. Zusätzliches Papier können Sie von den Tutoren bekommen.
- Tragen Sie jetzt (vor Beginn der eigentlichen Bearbeitungszeit !!!) auf *allen* Blättern ihren Namen und ihre Matrikelnummer ein.
- Schreiben Sie deutlich! Unleserliche oder zweideutige Lösungen können nicht gewertet werden.

- Schreiben Sie **nicht** mit Bleistift.
- Bitte schreiben Sie nicht mit rotem oder grünem Stift (das sind die Farben für die Korrektur).
- Lesen Sie die Aufgaben jeweils bis zum Ende durch; oft gibt es hilfreiche Hinweise! Beachten Sie, daß die Aufgaben mit unterschiedlichen Punktezahlen bewertet werden und teilen Sie Ihre Zeit danach ein.
- Wir weisen noch einmal darauf hin, daß die Benutzung von Taschenrechnern nicht gestattet ist.

Viel Erfolg!

• **AUFGABE 1 (8 Punkte) Theorie.**

1. (2 Punkte) Nennen Sie fünf Sortieralgorithmen. Wie verhalten sich bei den einzelnen Algorithmen die Phasen von Zerlegung und Aufbau bezüglich des Aufwands zueinander?

2. (1 Punkt) Was bedeuten die Worte *stabil* und *in situ* im Zusammenhang mit Sortieralgorithmen?

3. (1 Punkt) Wie definiert man formal einen endlichen Automaten?

4. (2 Punkte) Was ist eine Referenz in *JAVA*? Erläutern Sie die beiden verschiedenen Arten von *Gleichheit*, die für Referenzen definiert sind.

5. (2 Punkte) Erläutern Sie anhand eines Beispiels, welche Vorteile dynamische (verkettete) Listen gegenüber (statischen) Arrays haben.

• **AUFGABE 2 (4 Punkte) Numerik.**

1. (4 Punkte) Die *Eulersche Konstante* C ergibt sich als Grenzwert der Folge der reellen Zahlen:

$$c_n = \sum_{i=1}^{n} \frac{1}{i} - \ln n$$

Schreiben Sie eine *JAVA*-Methode, die die Eulersche Konstante per Approximation ermittelt. Beenden Sie den Approximationsprozeß, wenn eine Genauigkeit von 10^{-10} erreicht ist. Formulieren Sie diese Bedingung mit Hilfe einer Funktion `boolean close( double x, double y )`.

`double euler()`

Benutzen Sie zur Berechnung der Summe in jedem Schritt die Gleichung

$$S_n = S_{n-1} + \frac{1}{n}$$

und zur Ermittlung des natürlichen Logarithmus die Methode `double Math.log( double a )`.

Diese Seite wurde absichtlich leergelassen

• **AUFGABE 3 (12 Punkte) Endliche Automaten.** Eine beliebte Anwendung von endlichen Automaten ist die Untersuchung von Zeichenketten. In dieser Aufgabe sollen Sie einen einfachen Automaten entwerfen und implementieren, der in einer Kette von Ziffern die Zeichenfolgen *000* und *11* erkennt (Sie können davon ausgehen, daß nur die Ziffern 0, 1 und 2 vorkommen).

Gehen Sie dabei vor wie folgt:

1. (3 Punkte) Zeichnen Sie das Zustandsdiagramm des Automaten. Wenn eine der beiden Zeichenfolgen erkannt wurde, soll der Automat in den Zielzustand übergehen.
2. (3 Punkte) Schreiben Sie eine Klasse `NumberTester`, die als Attribut die Zustandsübergangsmatrix enthält, die Ihrem Diagramm aus der ersten Teilaufgabe entspricht

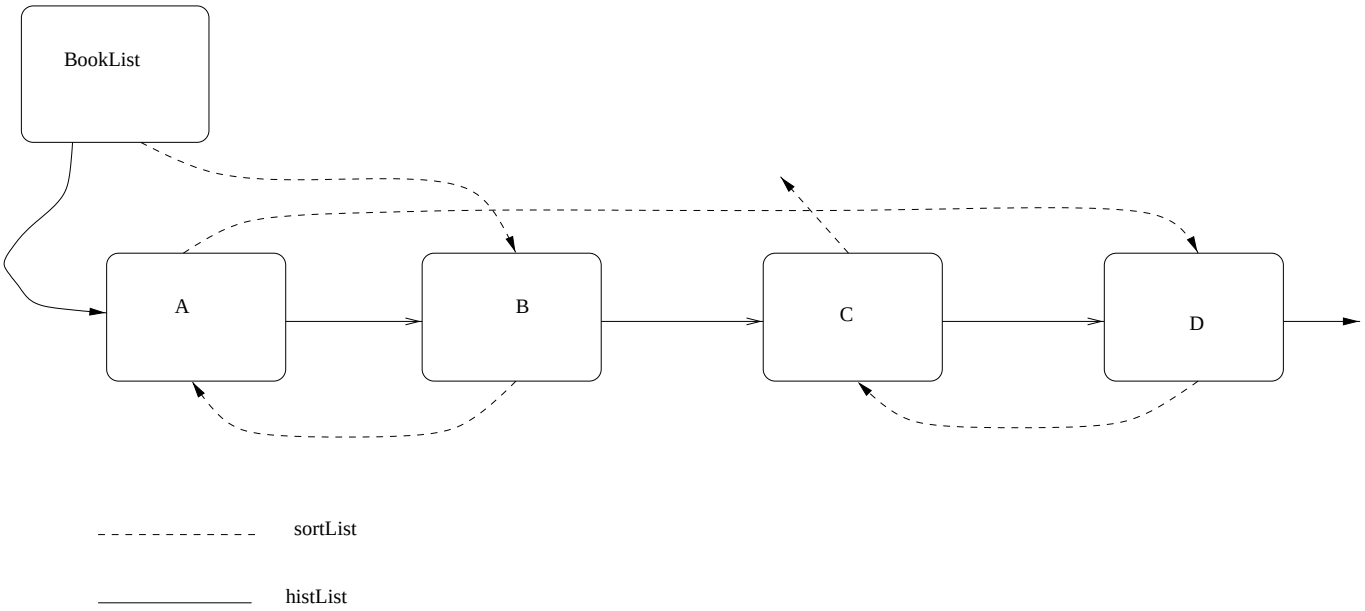
   ```
   private int[][] states = { ... };
   ```

3. (2 Punkte) Des weiteren soll Ihre Klasse eine Methode `public int trans( int state, int input )` besitzen, die anhand der Matrix den neuen Zustand berechnet. Außerdem benötigen Sie noch eine Methode `public int init( int input )`, die das erste Eingabezeichen auswertet.
4. (1 Punkt) Schließlich brauchen Sie noch die Methode `public boolean goal( int state )`, die prüft, ob sich der Automat im Zielzustand befindet.
5. (1 Punkt) Erstellen Sie eine Klasse `Automaton`, die ein Attribut vom Typ `NumberTester` besitzt. Erzeugen Sie ein Objekt der Klasse `NumberTester` und weisen Sie es diesem Attribut zu.
6. (2 Punkte) Schreiben Sie eine Methode `public boolean test( int[] in )` die `true` liefert, wenn sich eine der gesuchten Zeichenfolgen im Array `in` befindet (Hinweis: Brechen Sie die Suche ab, wenn der Automat in einen Zielzustand eintritt oder das letzte Eingabezeichen behandelt wurde).

Diese Seite wurde absichtlich leergelassen

• **AUFGABE 4 (14 Punkte) Simultane Verkettung nach zwei Kriterien.** Mit Hilfe der Verzeigerung von Elementen einer Liste kann man auch verschiedene Sortierreihenfolgen definieren. So sollen z.B. in einer Bibliothek Bücher katalogisiert werden. Jedes Buch wird durch eine eindeutige positive ganze Zahl repräsentiert. Um Bücher leicht im Bestandskatalog finden zu können, sollen sie nach ihren Nummern aufsteigend sortiert abgelegt werden. Andererseits ist es notwendig, die Reihenfolge des Kaufs zu speichern.

Zur Lösung dieser Aufgabe soll eine verkettete Liste verwendet werden, deren Elemente nach zwei Kriterien verkettet sind. Eine Klasse `BookList` soll die Liste verwalten und die Zugriffsmethoden zur Verfügung stellen.



Gehen Sie bei der Lösung der Aufgabe vor wie folgt:

1. (4 Punkte) Schreiben Sie eine Klasse `Book`, die über die Attribute `title` und `number` verfügt. Neben einem Konstruktor, der diese Attribute initialisiert, gibt es in der Klasse noch die Methode `boolean lower( Book other )`, die angibt, ob die Nummer des aktuellen Buches niedriger ist als die eines anderen. Außerdem soll es möglich sein, mit `void print()` Titel und Nummer des Buches auf dem Terminal auszugeben.
2. (2 Punkte) Schreiben Sie eine Klasse `ListCell`, die einen Listeneintrag definert. Sie enthält die Buchdaten und zwei Referenzen, die die Verkettung leisten (vgl. Abbildung).
3. (7 Punkte) Die Klasse `BookList` soll folgendes enthalten:
   - Zwei Attribute `histList` und `sortList`, die jeweils auf den Anfang der Listen zeigen.
   - *(Vorsicht: schwierig!)* Eine Methode `insert( Book b )` nimmt ein gegebenes Buch in die Liste auf, so daß es in der `histList` an erster Stelle steht und die Ordnung der `sortList` erhalten bleibt.
   - Die Methode `printSort()` gibt die Daten aller Bücher in der Liste entsprechend der Reihenfolge von `sortList` aus.

Diese Seite wurde absichtlich leergelassen

• AUFGABE 5 (6 Punkte) Binärbäume.

Bei der Bearbeitung dieser Aufgabe können Sie voraussetzen, daß Ihnen die Klasse `BinTree` zur Verfügung steht. Sie stellt Ihnen folgende Methoden zur Verfügung.

- Die Konstruktoren `BinTree(), BinTree( Object o )` und `BinTree( Object o, BinTree l, BinTree r )`.
- Die Zugriffsmethoden `Object value(), BinTree left()` und `BinTree right()`.
- Die Hilfsfunktionen `boolean isEmpty(), boolean isLeaf()` und `boolean isNode()`.

Im Folgenden können Sie davon ausgehen, daß die Werte der Knoten des Baumes vom Typ `Double` sind. Um also einen Wert vom Typ `double` in den Baum einzufügen, müssen Sie ihn mit `Double( double d )` zu einem Objekt der Klasse `Double` machen. Umgekehrt müssen Sie die Zahlen mit `double Double.doubleValue()` auspacken, bevor Sie mit ihnen rechnen.

1. (3 Punkte) Fügen Sie nun eine Methode `double getNumber()` hinzu, die die Anzahl der Knoten des Baumes berechnet. Verwenden Sie dabei das Verfahren der *postorder*-Traversierung.
2. (3 Punkte) Schreiben Sie eine weitere Methode `double getSum()`, die die Summe der Werte aller im Baum enthaltenen Konten berechnet. Benutzen Sie *inorder*-Traversierung.