Hinweise zur Bearbeitung der Klausur zum Kurs 1663 „Datenstrukturen“

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 3 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
   - 6 Aufgaben auf den Seiten 2 – 5

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. Nummerieren 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 MODULA-2-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 aber 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 MODULA-2-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 (kein Bleistift!) zugelassen.

10. Es sind maximal 100 Punkte erreichbar. Zur Erlangung eines Übungsscheins bzw. Zertifikats benötigen Sie etwa 50 Punkte.

16 Punkte

Aufgabe 1 (Rekursionsgleichungen)

Lösen Sie die folgenden Rekursionsgleichungen auf. Geben Sie zu jeder Lösung sowohl einen Induktionsbeweis als auch eine möglichst einfache Darstellung mit Hilfe der O-Notation an.

5 Punkte (a) $T(1) = 0$
$T(n) = T(n/2) + n, n > 1$
Zur Vereinfachung können Sie davon ausgehen, dass n eine Zweierpotenz ist.

5 Punkte (b) $T(1) = 1$
$T(n) = 2\ T(n\ -1) + 1, n > 1$

6 Punkte (c) $T(0) = 0$
$T(1) = 1$
$T(n) = 2\ T(n-1)\ -\ T(n-2) + 1, n > 1$

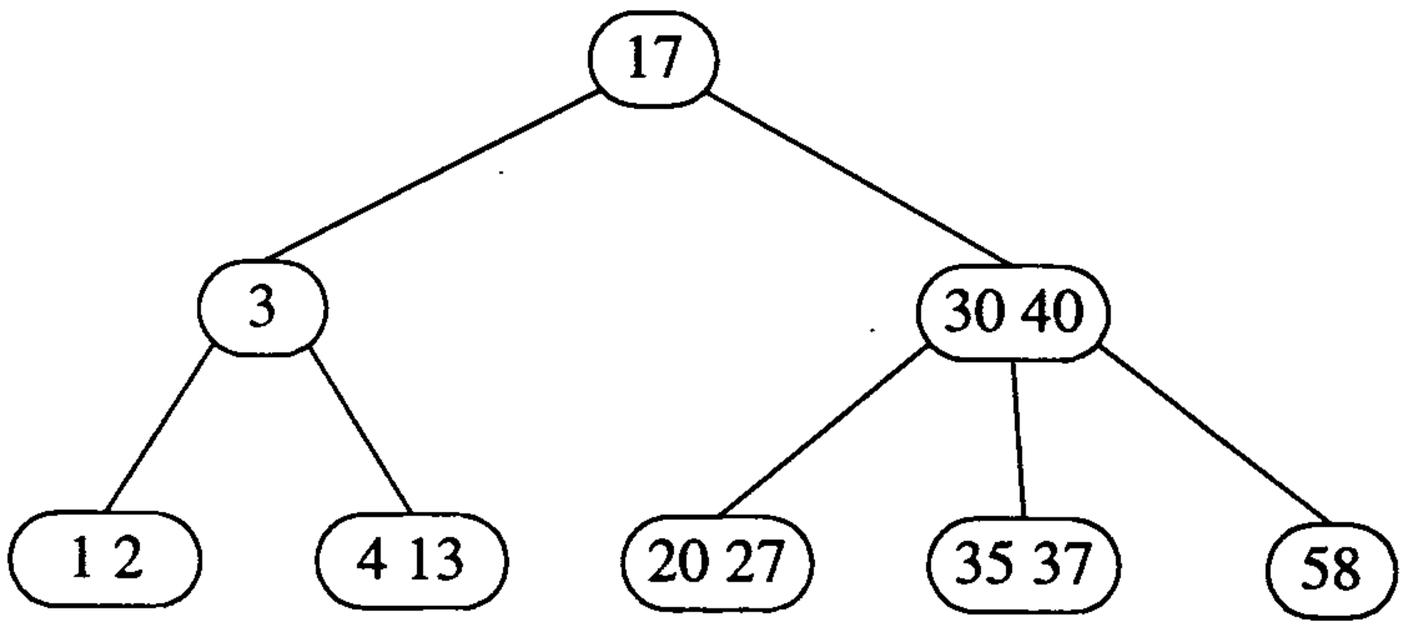
18 Punkte

Aufgabe 2 (B-Bäume)

9 Punkte (a) Erzeugen Sie einen B-Baum der Ordnung 1, indem Sie in den unten angegebenen Baum nacheinander die Schlüsselelemente

70, 8, 16, 25 und 19

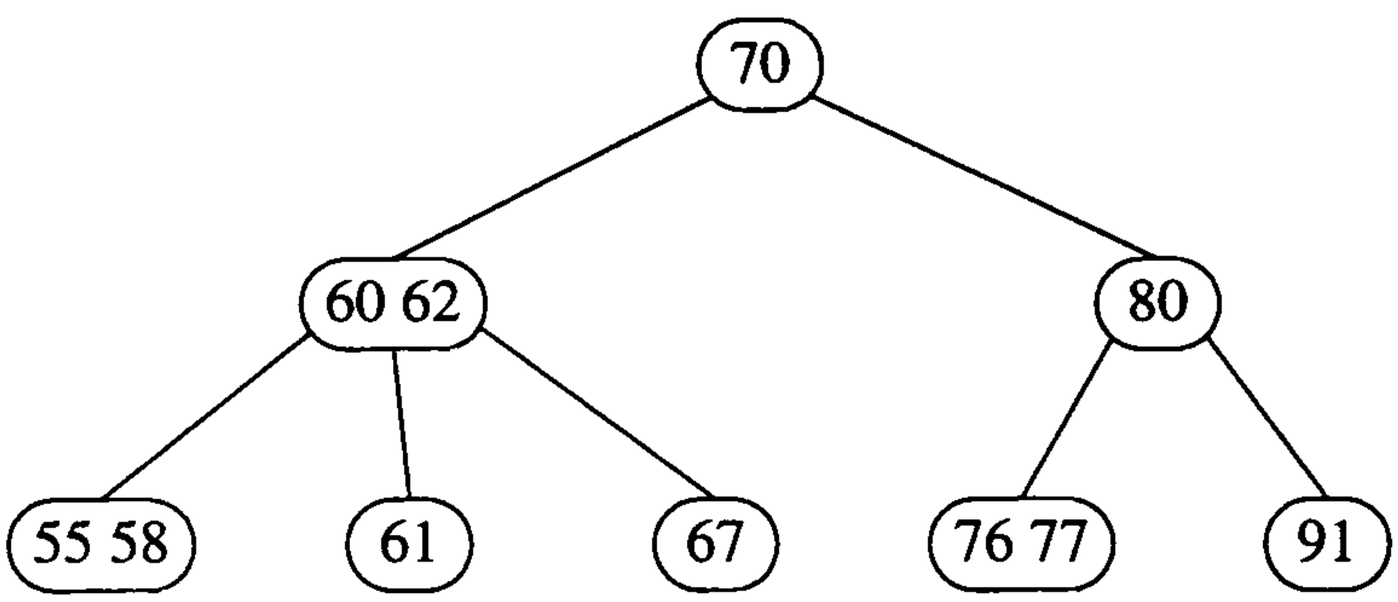
einfügen. Stellen Sie jeweils die Bäume dar, für die eine Overflow-Operation durchgeführt werden muss. Markieren Sie das entsprechende Blatt und stellen Sie außerdem das Ergebnis der Overflow-Operation dar. Zeichnen Sie abschließend den Baum, der als Ergebnis aller Einfügeoperationen entsteht.



(b) Löschen Sie aus dem unten gezeigten B-Baum der Ordnung 1 nacheinander die Schlüsselelemente *9 Punkte*

61, 76, 55, 58 und 77.

Stellen Sie die Bäume dar, in denen eine Underflow-Behandlung nötig ist, und markieren Sie den entsprechenden Knoten im Baum. Geben Sie an, mit welcher Operation zu reagieren ist und stellen Sie den Ergebnisbaum dieser Operation dar.



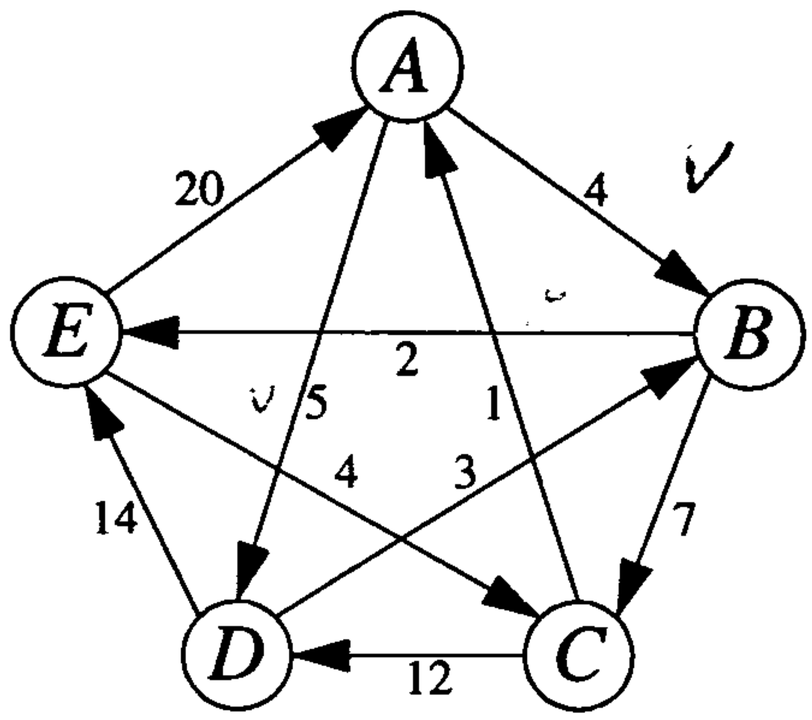
Aufgabe 3 (Insertion Sort) *14 Punkte*

Sortieren Sie die Zahlenfolge 35, 62, 28, 50, 11, 45 mittels *Insertion Sort*. Zeigen Sie jeweils für jeden Wert der Laufvariable *i* das komplette Array, so wie es am Ende eines jeden Schleifendurchlaufs aussieht. Markieren Sie jeweils durch Unterstreichen das bis dahin sortierte Teil-Array.

18 Punkte Aufgabe 4 (Dijkstras Algorithmus)

14 Punkte (a) Berechnen Sie mit Dijkstras Algorithmus zu dem nachstehend gezeigten Graphen vom Knoten *A* alle kürzesten Wege zu den anderen Knoten. Stellen Sie alle Zwischenergebnisse (nach der Bearbeitung eines neuen Knotens) als Graph folgendermaßen dar:

1. Notieren Sie die jeweils aktuellen Kosten in den Knoten.
2. Stellen Sie expandierte Knoten grün oder durch Schattierung dar.
3. Zeichnen Sie die Kanten des *shortest path tree* rot oder fett.
4. Markieren Sie den Knoten, der im nächsten Schritt bearbeitet werden soll, mit einem *.



4 Punkte (b) Welchen Gesamtaufwand und welchen Platzbedarf hat Dijkstras Algorithmus bei einer Implementierung mit Adjazenzlisten und als Heap dargestellter Priority Queue?

18 Punkte Aufgabe 5 (Geometrische Algorithmen)

7 Punkte (a) Beschreiben Sie kurz die drei Schritte des Divide and Conquer-Algorithmus für orthogonale Objekte für die Lösung des Segmentschnitt-Problems.

7 Punkte (b) Welche unterschiedlichen Fälle für die Lage eines horizontalen Segments unterscheidet der Algorithmus? Stellen Sie diese Fälle dar.

4 Punkte (c) Welche Laufzeit und welchen Platzbedarf hat der Algorithmus?

Aufgabe 6 (Listenumkehrung) *16 Punkte*

Geben Sie eine Prozedur *Reverse*(*head*: *listelem*) an, die die Reihenfolge der Elemente in einer einfach verketteten Liste umkehrt. Wie im Kurs beschrieben, bestehe die Liste dabei aus Elementen vom Typ

```
listelem = record
             value: elemtype;
             succ: ↑ listelem
           end
```

Dabei dient das erste physikalische Listenelement als Kopfelement, dessen *value*-Feld ungenutzt bleibt und dessen *succ*-Feld auf das erste logische Listenelement zeigt.

Die Prozedur *Reverse* soll nicht rekursiv implementiert sein, ihre Aufgabe in $O(n)$ erfüllen, mit einem konstanten Hilfsspeicher auskommen und keine neuen Listenelemente allokieren.

Zur Vereinfachung können Sie voraussetzen, dass *Reverse* nur für Listen mit mehr als 2 Elementen aufgerufen wird.