Hinweise zur Bearbeitung der Klausur zum Kurs 1662/1663 „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. 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 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.

12 Punkte

Aufgabe 1 (Speicherbedarf - Speicherlöcher)

Gegeben Sei die folgende Pascal-Implementierung von Merge-Sort:

```
CONST MAX = 100;
TYPE TField = ARRAY[0..MAX] of integer;
     PField = ^TField;

function merge(Part1,Part2 : PField; left,mid,right : integer) : PField;
{verschmilzt zwei sortierte Felder zu einem sortiertem Feld}
var Res  : PField;
    {... weitere Variablendeklarationen}
begin
   new(Res);  {erzeuge Ergebnis}
   {... verschmelze die gegebenen Felder,
        speichere das Ergebnis  in Res}
   merge := res;
end;


function mergesort(Field : PField;left,right:integer) : PField;
var Part1,Part2 : PField;
    Res : PField;
    mid : integer;
begin
  if(left=right) then {Abbruch, nur ein Element}
  begin
    new(Res);
    Res^[left] := Field^[left];
    MergeSort := Res
  end
  else
  begin
     mid := (left+right) div 2;
     Part1 := mergesort(Field,left,mid);
     Part2 := mergesort(Field,mid+1,right);
     Res := merge(Part1,Part2,left,mid,right);
     dispose(Part1);
     dispose(Part2);
     MergeSort := Res;
  end;
end;
```

7 Punkte

(a) Geben Sie den Speicherplatzbedarf dieser Funktion in Abhängigkeit von MAX an. Verwenden Sie hierzu die O-Notation und begründen Sie Ihr Ergebnis (eine informelle Beschreibung genügt).

5 Punkte

(b) Wird in Programmen belegter Speicher nicht wieder freigegeben, spricht man von sogenannten Speicherlöchern (memory leaks). Geben Sie an, wieviel Speicher von der oben angebenen Funktion verschwendet wird, wenn der reservierte Speicher nicht wieder freigegeben wird (also die beiden `dispose`-Anweisungen im Programm fehlen). Geben Sie Ihr Ergebnis in O-Notation an und begründen Sie es.

Aufgabe 2 (B-Bäume) *16 Punkte*

(a) Fügen Sie die folgenden Elemente in einen B-Baum der Ordnung 2 ein. Zeichnen Sie den Baum immer dann, wenn eine Overflow-Operation nötig ist und markieren Sie den entsprechenden Knoten. *8 Punkte*

7, 12, 55, 103, 1, 5, 97, 22, 13, 11, 2

(b) Löschen Sie aus dem Ergebnisbaum aus Aufgabenteil a) nacheinander die Schlüssel *8 Punkte*

12, 13, 55

und zeichnen Sie die Ergebnisbäume nach jeder Löschoperation.

Aufgabe 3 (HeapSort) *18 Punkte*

(a) Beschreiben Sie die Idee des Bottom-Up-Heapsort (im Unterschied zum Standard-Heapsort). *6 Punkte*

(b) Wenden Sie Heapsort an, um eine aufsteigend sortierte Folge zu erhalten. Fügen Sie dazu zunächst die Elemente *12 Punkte*

50, 17, 3, 8, 25, 13

der Reihe nach in einen Heap ein. Geben Sie den Ausgangsheap und seine Array-Einbettung an.

Führen Sie nun Heapsort durch. Zeichnen Sie den Heap nach jedem Schritt und markieren den Einsinkpfad und die gefundene Position gemäß der Idee des Bottom-Up-Heapsort. Markieren Sie im Heap außerdem die bereits sortierte Folge besonders.

20 Punkte

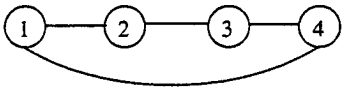
Aufgabe 4 (Berechnung der Transitiven Hülle)

Nachfolgend ist der Algorithmus von Floyd, der aus der Kostenmatrix C eines Graphen $G=(V,E)$ die Kosten der kürzesten Wege zwischen allen Knoten ermittelt und diese in der Kostenmatrix A speichert, dargestellt. Die Matrix P wird zur Rekonstruktion der kürzesten Pfade benötigt.

```
algorithm Floyd ( out A: |V| × |V| Kostenmatrix der kürzesten Wege,
                  in  C: |V| × |V| Kostenmatrix von G )
begin
  kopiere die Werte aus C nach A.
  for i := 1 to n do{dies sind die Haupt-"Schritte" des Algorithmus}
    for j := 1 to n do
      for k := 1 to n do
        if A_ji + A_ik < A_jk then do
          A_jk := A_ji + A_ik
          P_jk=i
        end
      end
    end
  end
end Floyd;
```

12 Punkte (a) Modifizieren Sie Floyds-Algorithmus, um damit die Transitive Hülle von ungerichteten Graphen berechnen zu können, gehen Sie dabei möglichst effizient vor.

8 Punkte (b) Berechnen Sie die Transitive Hülle des unten dargestellten Graphen. Stellen Sie den Graphen als Adjazenzmatrix dar, und listen Sie nur alle ***notwendigen*** Rechenschritte auf, zeichnen Sie auch den resultierenden Graphen.



14 Punkte

Aufgabe 5 (Range-Baum)

7 Punkte (a) Fügen Sie die Zahlenfolge {7,2,6,1,8,3} in einen Range-Baum über dem Raster {1...8} ein. Erzeugen Sie zunächst einen leeren Baum über diesem Raster, und fügen Sie dann die Elemente ein.

7 Punkte (b) Welchen Platzbedarf hat ein so erzeugter Range-Baum über dem Raster {1...N}, der n Koordinaten enthält? Begründen Sie Ihre Antwort.

Aufgabe 6 (k-kleinstes Element) *20 Punkte*

Manchmal ist es notwendig, den minimalen oder den maximalen Wert einer Folge zu finden. Eine Verallgemeinerung hiervon ist es, das k-kleinste Element einer Menge zu finden. Eine Möglichkeit, dies zu erreichen ist es, die Folge zunächst zu sortieren und das k-te Element auszuwählen. Der dazu notwendige Aufwand beträgt $O(n\log(n))$, wobei n'die Folgenlänge darstellt. Es ist jedoch nicht notwendig, die gesamte Folge zu sortieren, um das gesuchte Element zu finden.

(a) Entwickeln Sie einen DAC-Algorithmus, der das k-kleinste Element einer gegebenen Folge berechnet. Ihr Algorithmus sollte im durchschnittlichen Fall in $O(n)$ Zeit auskommen. Sie können davon ausgehen, daß alle Werte in der Eingabefolge paarweise verschieden sind. *11 Punkte*

(b) Beweisen Sie die durchschnittliche Zeitschranke Ihres Algorithmus. Stellen Sie hierfür zunächst die Rekursionsgleichung auf und lösen Sie diese anschließend auf. Zeigen Sie die Korrektheit der Auflösung mittels vollständiger Induktion. *9 Punkte*

Hinweis: Zur Vereinfachung können Sie in Ihrer Berechnung $O(n)$ durch n und konstante Werte durch 1 ersetzen. Weiterhin dürfen Sie annehmen, daß n eine Zweierpotenz ist.