Hinweise zur Bearbeitung der Aufgaben D1 – D7:

- Wenn Sie aufgefordert werden einen *Algorithmus* zu *formulieren*, so tun Sie dies bitte in Modula-3 oder einer ähnlichen Sprache (Pascal, C). In jedem Fall dürfen Sie ausschließlich Konstrukte verwenden, die auch in Modula-3 zur Verfügung stehen. Verwenden Sie Kommentare um die Idee Ihres Algorithmus zu verdeutlichen.

  Es kommt in erster Linie auf semantische Korrektheit und Verständlichkeit an. Syntaktische Feinheiten bis auf das letzte Semikolon sind nicht so wichtig. Konzentrieren Sie sich auf den Kern des Algorithmus; lassen Sie triviale Ein- und Ausgabeprozeduren weg.

- Verwenden Sie bei Bedarf das folgende

  Master-Theorem

  Sei $f(n)$ eine Funktion und $T(n)$ definiert durch die Rekursion

  $$T(n) = aT(n/b) + f(n),$$

  wobei $a \geq 1$ und $b > 1$ positive reelle Konstanten sind. Dann läßt sich $T(n)$ in asymptotischer Schreibweise wie folgt ausdrücken:

  - Falls $f(n) = O(n^{\log_b a - \varepsilon})$, $\varepsilon > 0$, dann ist $T(n) = \Theta(n^{\log_b a})$.
  - Falls $f(n) = \Theta(n^{\log_b a})$, dann ist $T(n) = \Theta(n^{\log_b a} \cdot \log_b n)$
  - Falls $f(n) = \Omega(n^{log_b a + \varepsilon})$, $\varepsilon > 0$, und $af(n/b) \leq cf(n)$ mit $c < 1$ und genügend großem n, dann ist $T(n) = \Theta(f(n))$.

Aufgabe D1: Rekursionsgleichungen **8 Punkte**

a) Schätzen Sie die folgenden Rekursionsgleichungen unter Verwendung der Substitutionsmethode möglichst exakt ab. Formulieren Sie Ihre Antwort mit Hilfe der O-Notation.

i) $T(n) = T(\lfloor \sqrt[3]{n} \rfloor) + 1 \qquad T(3) = 1$ (2 Punkte)

ii) $T(n) = T(n-1) + n \qquad T(0) = 0$ (2 Punkte)

iii) $T(n) = T(n-a) + T(a) + 2n \qquad T(0) = 0$ (2 Punkte)

b) Verwenden Sie das Master-Theorem, um die folgenden Rekursionsgleichungen zu lösen:

i) $T(n) = 7T(n/2) + 5n^2$ (1 Punkt)

ii) $T(n) = 2T(n/2) + 3n$ (1 Punkt)

Aufgabe D2: Laufzeitanalyse **8 Punkte**

Untersuchen Sie die folgende Modula-3-Prozedur:

```
1  TYPE ItemType = ... ;
   CONST N = 100 ;
   PROCEDURE A(VAR a : ARRAY [1..N] OF ItemType ; v : ItemType) : BOOLEAN =
       PROCEDURE B(n : INTEGER) : BOOLEAN  =
5      VAR m : INTEGER ;
       BEGIN
           IF n > 0 THEN
               m := (1 + n) DIV 2 ;
               IF v = a[m] THEN
10                 RETURN TRUE ;
               ELSIF v < a[m] THEN
                   RETURN B(m-1) ;
               ELSE
                   FOR i := m + 1 TO n DO
15                     a[i-m] := a[i] ;
                   END ;
                   RETURN B(n - m) ;
               END ;
           ELSE
20             RETURN FALSE ;
           END ;
       END B ;
   BEGIN
       RETURN B(N) ;
25 END A ;
```

Bei dem ersten Argument a handele es sich stets um ein *aufsteigend sortiertes* Array.

Aufgaben:

a) In welchem Fall liefert diese Prozedur TRUE, wann FALSE zurück? Wie funktioniert der zugrundeliegende Algorithmus? Wieviele Vergleiche zwischen Werten des Typs ItemType benötigt er größenordnungsmäßig in Abhängigkeit von N? Und wie heißt er? (2 Punkte)

b) Analysieren Sie den Laufzeitbedarf dieser Implementierung. Stellen Sie dazu eine Rekursionsgleichung auf, lösen Sie diese und geben Sie Ihr Ergebnis in der O-Notation an. (4 Punkte)

c) Warum ist diese Implementierung (hinsichtlich des Laufzeitbedarfs) ziemlich schlecht? Beschreiben Sie wie man die Implementierung verbessern kann, so daß sich eine günstigere Laufzeitkomplexität (Welche?) ergibt. (2 Punkte)

Aufgabe D3: QuickSort / HeapSort — 8 Punkte

Gegeben sei die folgende Sequenz von Schlüsselelementen (unter Annahme der üblichen Ordnung der natürlichen Zahlen):

5, 2, 7, 8, 3, 4, 1, 6

a) Sortieren Sie diese Folge der Schlüsseln mittels QuickSort, wobei für jede Partitionierung der am weitesten rechts stehende Schlüssel als Pivot-Element verwendet werden soll. Geben Sie bei Ihrer Lösung auch sämtliche Teilfolgen an. (4 Punkte)

b) Sortieren Sie die gleiche Schlüsselfolge nun mittels HeapSort unter Verwendung der aus der Vorlesung bekannten DownHeap-Prozedur („Versickern"). Geben Sie sämtliche Zwischenergebnisse an. (4 Punkte)

Aufgabe D4: Hashing — 5 Punkte

Gegeben sei ein unsortiertes Array der Länge n, wobei jedes Element einen Schlüssel k sowie den dazugehörigen Hashwert $h(k)$ enthält. Bei den Schlüsseln handelt es sich um Zahlenfolgen der Länge m. Die Hashwerte belegen jeweils nur eine Speicherzelle. Wie läßt sich die Suche nach einem Schlüssel durch die Einbeziehung der Hashwerte beschleunigen? Formulieren Sie einen entsprechenden Algorithmus unter Verwendung der folgenden Deklaration:

```
VAR a : ARRAY [1..n] OF RECORD
                          k : ARRAY [1..m] OF CARDINAL;
                          h : CARDINAL;
                        END;

PROCEDURE h(READONLY k : ARRAY [1..m] OF CARDINAL) : CARDINAL;
```

Aufgabe D5: AVL-Baum — 5 Punkte

a) Geben Sie den AVL-Baum an, der entsteht, wenn man nacheinander die Ziffern

1, 3, 2, 5, 6, 4, 0

in den anfangs leeren Baum einfügt (unter Annahme der üblichen Ordnung der natürlichen Zahlen). Geben Sie dabei nach jedem Einfügen auch den entstandenen Zwischenbaum an. (4 Punkte)

b) Entfernen Sie anschließend die Ziffer 3 aus dem im vorhergehenden Teil entstandenen Baum und geben Sie den resultierenden Baum an. (1 Punkt)

Aufgabe D6: Dynamische Programmierung — 8 Punkte

Ede und Greedy planen ihren nächsten Coup. Diesmal wollen Sie einen Computer-Laden am Templergraben ausrauben. Dabei interessieren sie die folgenden Artikel, die jeweils in *beliebiger* Stückzahl vorhanden sind:

i	Artikel	Gewicht g_i	Wert w_i
1	BNE Athlet-CPU 1 GHz	5 kg	1500 DM
2	256 MB RAM	3 kg	800 DM
3	NVideo H-Force Grafikkarte	2 kg	500 DM
4	Else 15" TFT-Bildschirm	8 kg	2500 DM

Edes Rucksack faßt maximal 12 kg. Welche Artikel müssen die Ganoven in welcher Stückzahl einpacken, um den maximalen Beutewert zu erhalten?

Lösen Sie das Problem mittels dynamischer Programmierung unter Verwendung der folgenden Hilfsgröße:

m(G) := Wert der besten Zusammenstellung, deren Gesamtgewicht kleiner oder gleich G ist.

Aufgaben:

a) Stellen Sie die Rekursionsgleichung für $m(G)$ auf. (2 Punkte)

b) Formulieren Sie einen Algorithmus, der die optimale Bepackung des Rucksacks berechnet. (3 Punkte)

c) Geben Sie die Laufzeit Ihres Algorithmus in Abhängigkeit von der Kapazität des Rucksacks und der Anzahl der verschiedenen Artikel an. (1 Punkt)

d) Demonstrieren Sie Ihren Algorithmus anhand des obigen Beispiels. (2 Punkte)

Aufgabe D7: Übersetzungen **8 Punkte**

Eine europäische Behörde beschäftigt ein Team von Übersetzern, die eingehende Akten in die unterschiedlichsten Sprachen übersetzen müssen.

Beispiel: Es werden fünf Personen beschäftigt, die die folgenden Sprachen beherrschen:

A: Russisch – Tschechisch
B: Spanisch – Portugisisch
C: Spanisch – Französisch – Italienisch
D: Französisch – Englisch – Italienisch
E: Russisch – Polnisch

Es kann z.B. vom Englischen ins Spanische (über das Französische oder Italienisch) übersetzt werden, nicht aber vom Polnischen ins Englische.

Aufgaben:

a) Modellieren Sie diese Problemstellung als ungerichteten Graphen, bei dem die Knoten den einzelnen Sprachen entsprechen. Formulieren Sie eine Vorschrift der Form „Die Kante $A \leftrightarrow B$ existiert genau dann, wenn ... ". Geben Sie die Adjazenz-Matrix für das obige Beispiel an. Beschriften Sie dabei die Zeilen und Spalten der Matrix in alphabetischer Reihenfolge. (1 Punkt)

b) Unter welcher Bedingung kann ein Dokument von einer Sprache A in eine Sprache B übersetzt werden? (1 Punkt)

c) Formulieren Sie einen Algorithmus, der für jedes Paar von Sprachen (A, B) berechnet ob eine Übersetzung möglich ist, und wenn ja über wieviele Zwischenstationen diese mindestens erfolgen muss. (3 Punkte)

d) Welche Laufzeitkomplexität hat Ihr Algorithmus? (1 Punkt)

e) Demonstrieren Sie ihren Algorithmus an dem obigen Beispiel, und geben Sie Ihr Ergebnis in Form einer Tabelle an. (2 Punkte)