Vordiplom für Wirtschaftswissenschaften
Allgemeine Informatik II – WS 2001/02
12. Oktober 2001
Bearbeitungszeit: 120 Minuten

Name:	
Vorname:	
Matrikelnummer:	

Aufgabe	Punkte	Bewertung Teilaufgaben	Gesamt
1	**12**		
2	**8**		
3	**12**		
4	**8**		
5	**10**		
a)	5		
b)	5		
6	**8**		
7	**8**		
8	**14**		
Summe:	**80**		

Bitte benutzen Sie für die Lösungen den freigelassenen Platz nach der jeweiligen Angabe; sollte dieser nicht ausreichen, verwenden Sie bitte die Rückseite, wobei die Zuordnung zur jeweiligen Aufgabe deutlich erkennbar sein muss!

Bitte nicht mit Bleistift schreiben
Die Angabe besteht aus 10 Seiten (ohne Titelblatt)!
Viel Erfolg!!!

Aufgabe 1 (Rekursion) (12 Punkte)

Gegeben sei folgende Oberon-Programm:

```
MODULE Do;
   IMPORT Write;
   CONST Length = 10;
   VAR a: ARRAY Length OF INTEGER;
       erg, i, el: INTEGER;
(**************************************************************)
   PROCEDURE Rekursi(f: ARRAY OF INTEGER;
                     left, right,el: INTEGER;
                     VAR count: INTEGER):INTEGER;
     VAR m: INTEGER;
   BEGIN
     IF (left > right) OR (left < 0) OR (right >= LEN(f))
     THEN RETURN -1;
     END;
     Write.String("left = "); Write.Int(left,0);
     Write.String(" : right = "); Write.Int(right,0);
     Write.Ln;
     m := (left + right) DIV 2;
     IF f[m] = el THEN RETURN m
     ELSIF el < f[m] THEN
       INC(count); RETURN Rekursi(f, left, m-1,el,count)
     ELSE
       INC(count); RETURN Rekursi(f, m+1, right, el,count)
     END;
   END Rekursi;
(**************************************************************)
BEGIN
   i := 0;
   WHILE (i < Length) DO
     a[i] := 2*i; INC(i);
   END;
   i := 1;
   erg := Rekursi(a,0,Length-1,13,i);
   Write.String("Ergebnis von Rekursi: ");
   Write.Int(erg, 0); Write.Ln;
   Write.String("Anzahl rekursiver Aufrufe: ");
   Write.Int(i,0); Write.Ln
END Do.
```

Weiter nächste Seite

Das im Programm verwendete Feld **a** sieht also wie folgt aus:

0	**2**	**4**	**6**	**8**	**10**	**12**	**14**	**16**	**18**

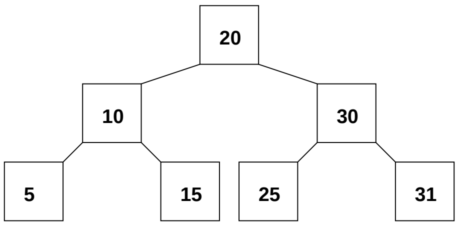
Aufgabe: Geben Sie an, was dieses Programm an die Standardausgabe ausgibt!
Lösung:

Aufgabe 2 (Rekursion, Bäume) (8 Punkte)

Durch folgende Typvereinbarungen wird ein Binärbaum definiert:

```
TYPE Tree = POINTER TO Node;
     Node = RECORD
               content: INTEGER;
               left, right: Tree;
            END;
```

In einem Programm wird auf dieser Basis ein sortierter (genauer: nach *inorder* sortierter) Binärbaum aufgebaut.
Beispiel:



Schreiben Sie nun eine **rekursive** Prozedur, die die Knoteninhalte eines solchen Baumes absteigend sortiert ausgibt.
Lösung:

```
PROCEDURE PrintDesc(root: Tree);
```

Aufgabe 3 (Streams) **(12 Punkte)**

In einer Datei erhalten Sie Daten Ihres Partners, in denen er sein “Börsengeschäft” aufgezeichnet hat. Eine Beispieldatei sehe wie folgt aus:
Die erste Zahl in einer Zeile beschreibt den Einstandspreis, die zweite den Verkaufspreis. Nur die Zeilen, in denen am Zeilenanfang (abgesehen von führenden Leerzeichen) zwei Zahlen stehen, sind relevant. Sie sollen ihm nun ein Oberon-Programm schreiben, das den durchschnittlichen Gewinn (negativer Gewinn = Verlust) ermittelt.
Die ersten drei Zeilen des Programms (nennen wir es **guv**) sind Ihnen vorgeben, die einen Hinweis auf die Vorgehensweise enthalten. Die Datei soll dem Programm üeber die Standardeingabeverbindung zugeführt werden (also: `guv < datei`)
Lösung:

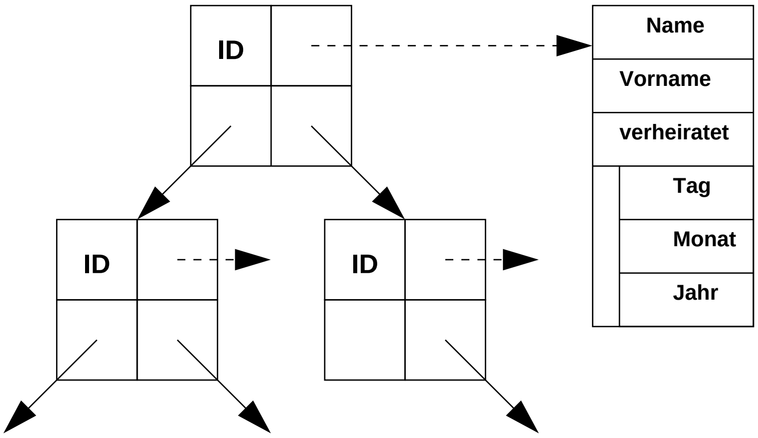
```
MODULE guv;
  IMPORT Read, Write, Streams, Strings;
  CONST LineLength = 80;
```

Aufgabe 4 (Bäume) (8 Punkte)

In einem Personalführungssystem sollen folgende Daten verwendet werden:

Personalnummer	ganze positive Zahl	Eindeutiger Schluessel
Familienname	max. 31 Zeichen lang	
Vorname	max. 31 Zeichen lang	
verheiratet	ja / nein	
Geburtsdatum	Tag, Monat, Jahr	

Um diese Daten zu verwalten, schlägt Ihnen jemand skizzenhaft folgende Struktur vor:



Die Personalnummern (ID) wird in den Knoten eines Binärbaumes geführt, die zusätzlichen Informationen werden über einen Zeiger an den Knoten angefügt.
Aufgabe: Geben Sie die notwendigen Vereinbarungen (Konstanten, Typen) in Oberon an, so dass eine Variable für diesen Baum vereinbart werden kann.

Aufgabe 5 (Quicksort) **(10 Punkte)**

Gegeben sein ein Array mit ganzen Zahlen:

```
VAR a: ARRAY Length OF INTEGER;
```

Um mit dem Quicksort-Algorithmus zu arbeiten, muss in jedem Teilungsschritt eines zu behandelnden Teilfeldes $a[i] \ldots a[j], i < j$ ein Bezugselement (typischerweise eines der Feldelemente) – nennen wir es **x** – gewählt werden. Sie entscheiden sich für folgende Strategie:

```
x := a[ (i + j) DIV 2 ]
```

a) (5 Punkte)
Unter welchen Bedingungen ist diese Wahl extrem schlecht für die Performance des Quicksort?

Kreuzen Sie an, bei welchen der drei Felder A, B, oder C dies zutrifft!

Feld A: ☐

0	2	4	6	8

Feld B: ☐

2	6	0	4	8

Feld C: ☐

8	6	4	2	0

b) (5 Punkte)
Unter welchen Bedingungen ist diese Wahl optimal für die Performance des Quicksort? Geben Sie eine Begründung an!

Aufgabe 6 (Argumentverarbeitung) (8 Punkte)

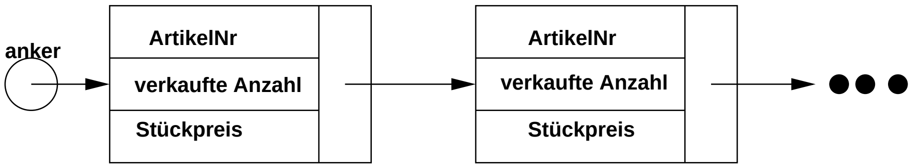
Schreiben Sie ein Oberon-Programm, welches beim Aufruf auf der Kommandozeile „beliebig" viele ganze Zahlen als Argumente erhalten kann und als Ergebnis deren Summe an die Standardausgabe schreibt. Ohne Argument soll **0** als Summe geliefert werden! Zeichen, die keine Ziffern sind, sollen mit einer entsprechenden *Usage*-Meldung zum Abbruch führen.
Beispielaufrufe:

```
thales$ Sum 1 2 33
36
thales$ Sum 1.2 3
Usage: Sum {integer}
thales$
```

Hinweis: Prozedur *OpenArg()* – Skript!
Lösung:

Aufgabe 7 (Listen) **(8 Punkte)**

Gegeben sei folgende Liste:



Also in Oberon:

```
TYPE
    List = POINTER TO ListRec;
    ListRec = RECORD
                artNr, anzahl, preis (*in Cent*): INTEGER;
                next: List;
              END;
    Betrag = RECORD
               euro,cent: INTEGER;
             END;
```

Schreiben Sie eine Oberon-Prozedur, die dem Aufrufer den Gesamtumsatz aller in der Liste enthaltenen Artikel liefert, und zwar aufgeschlüsselt in Euro und Cent, also bei einem Gesamtumsatz von 2750 Cent sollen die beiden Zahlen 27 und 50 geliefert werden.

Lösung:

Aufgabe 8 (File-I/O) (14 Punkte)

Manchmal besteht die Notwendigkeit, die Inhalte von zwei Dateien zeilenweise „zusammenzukleben“, d.h. die i-te Zeile der ersten Datei wird an die Standardausgabe ausgegeben, danach wird ein Trenn-Zeichen (hier: das Pipeline-Symbol |) ausgegeben, danach die i-te Zeile der zweiten Datei. Ist die i-te Zeile in einer der beiden Dateien nicht (mehr) vorhanden, so wird der leere String ausgegeben.

Beispiel: Erste Datei (*file1*):

```
Deutsche Bank
Dresdner Bank
HypoVereinsbank
Sparkasse Ulm
```

Zweite Datei (*file2*):

```
-5.2%
-1.2%
+/- 0%

XXXX
```

Programmaufruf:

```
thales$ Paste file1 file2

Deutsche Bank|-5.2%
Dresdner Bank|-1.2%
HypoVereinsbank|+/- 0%
Sparkasse Ulm|
|XXXX
```

Ihre Aufgabe: Realisieren Sie dieses Programm! Es müssen genau zwei Argumente (Dateinamen) als Argumente kommen; wenn eine der angegebenen Datei nicht lesbar ist, kann das Programm mit einer einfachen Fehlermeldung auf die Diagnoseausgabe beendet werden. Die ersten Zeilen des Programms sind als kleine Hilfestellung vorgebenen.

Weiter nächste Seite $\hookrightarrow$

Lösung:

```
MODULE Paste;
   IMPORT UF := UnixFiles, UA := UnixArguments, Process,
          Streams, Read, Write;
   CONST Length = 256;
   TYPE String = ARRAY Length OF CHAR;
   VAR
        line: String;
        str1, str2: Streams.Stream;
        file1, file2: String;
BEGIN
```