Vordiplom Wiwi und Wima-Schein-klausur

Lösung zur Allgemeine Informatik II

SS 2000

19. + 26.Oktober 2000

Aufgabe		Punkte	Bewertung
1		8	
2		8	
3	a)	8	
	b)	8	
4		8	
5		6	
6	a)	8	
	b)	8	
7	a)	3	
	b)	3	
	c)	6	
	d)	6	
Summe		**80**	

☞ Bearbeitungszeit: 120 Min.

Aufgabe 1 (8 Punkte)

Argumentverarbeitung:

Ein Oberon-Programm namens `Check` soll beim Aufruf auf der Kommandozeile als *flags* nur Kleinbuchstaben erhalten; sollten andere Zeichen als *flags* gesetzt sein, so soll der Hinweis *flag: illegal character* auf die Standardausgabe ausgegeben werden; es sind **alle** übergebenen *flags* zu prüfen.

Bespiele:

`Check -yX -8 :`

Ausgabe:

```
X: illegal character
8: illegal character
```

Realisieren Sie das Oberon-Programm **Check.om**!

Lösung:

```
MODULE Check;
  IMPORT AV := UnixArguments, Write;

  VAR flag: CHAR;

BEGIN
  WHILE AV.GetFlag(flag) DO
     IF (flag < "a") OR (flag > "z") THEN
       Write.Char(flag);
       Write.String(": illegal character");
       Write.Ln;
     END
  END;

END Check.
```

Aufgabe 2 (10 Punkte)

Streams

Sie erhalten eine Datei `umsaetze` -- zu lesen durch Eingabeumlenkung von der Standardeingabe -- mit folgendem Aufbau:

- Jede Zeile (abgeschlossen durch ein *newline*) enthält beliebig viele (mindestens eine!) REAL-Zahlen, die durch *blanks* oder *tabs* getrennt sind.
- Die maximale Länge einer Zeile in Bytes gemessen sei 1024 (inkl. *newline*).
- Die j-te Zahl der i-ten Zeile sind die Einnahmen zu einem Produkt P_j zum Zeitpunkt T_i.

Ergänzen Sie untenstehendes Oberon-Programm so, dass pro Eingabezeile die

Mittelwerte der Produkteinnahmen P_j ausgegeben werden:

i: <Mittelwert_der_Zahlen_der_i-ten_Zeile>

Beispiel: Bei folgender Eingabedatei

1.2 3.2 5.6 5.0
7.5 2.0 2.5 10.0
3.6 8.4 6.0 10.0

soll das Programm folgende Ausgabe liefern:

1 : 3.750000D+0000
2 : 5.500000D+0000
3 : 7.000000D+0000

Lösung:

```
MODULE Count;
   IMPORT Streams, Strings, Read, Write;
   TYPE
      Line = ARRAY 1024 OF CHAR;
   VAR
      line: Line;
      sum, value: REAL;
      count, lineNo: INTEGER;
      str: Streams.Stream;
BEGIN
      lineNo := 1;
      Read.Line(line);
      WHILE Streams.stdin.count > 0 DO
         count := 0; sum := 0.0;
         Strings.Open(str, line);
         Read.RealS(str, value);
         WHILE str.count > 0 DO
           INC(count); sum := sum + value;
           Read.RealS(str, value);
         END;
(*         IF Streams.Close(str) THEN END;
*)
         Write.Int(lineNo,4); Write.String(" : ");
         Write.Real(sum / count, 4); Write.Ln;
         INC(lineNo);
         Read.Line(line);
      END;
END Count.
```


Aufgabe 3 (20 Punkte)

File-I/O und Argumentverarbeitung:

(*a*) 10 Punkte
Implementieren Sie die Prozedur `PointLines` (siehe unten), die die Anzahl der Zeilen bestimmt, die mit einem Punkt (**“.”**) beginnen. Die Zeilen sollen aus dem bereits geöffneten Stream `str` via `Read.LineS` gelesen werden.

```
PROCEDURE PointLines(str: Streams.Stream): INTEGER;
    VAR
      count: INTEGER;
      line: ARRAY 256 OF CHAR;
BEGIN
      count := 0;
      Read.LineS(str, line);
      WHILE ~ str.eof DO
        IF line[0] = "." THEN
          INC(count);
        END;
        Read.LineS(str, line);
      END;
      RETURN count
END PointLines;
```

(*b*) 10 Punkte

Auch wenn Sie Teilaufgabe a) nicht gelöst haben sollten, können Sie mit dieser Aufgabe fortfahren! Unter Verwendung der Prozedur aus a) sollen Sie nun ein Oberon-Programm `Count` schreiben das wie folgt aufgerufen werden kann:

`Count [file1 file2 ...]`

Wird *kein* Dateiname angegeben, so werden die Zeilen, die mit einem Punkt (**"."**) beginnen, in der *Standardeingabe* bestimmt und ausgegeben! Kommt mindestens ein Dateiname, so werden die Anzahl der Zeilen, die mit einem Punkt beginnen, in den angegegebenen Dateien bestimmt und die Ausgabe in der Form

<Dateiname> : <Anzahl "Punktzeilen">

erzeugt. Bei der Standardeingabe wird für "Dateiname" der String "stdin" ausgegeben. Sollte eine der angegebenen Dateien nicht zum Lesen geöffnet werden können, soll das Programm die Meldung "Cant't open <dateiname>" ausgeben und mit den weiteren Argumenten fortfahren. Ergänzen Sie das Programm:

```
MODULE Count;
   IMPORT Streams, UF := UnixFiles, AV := UnixArguments,
          Read, Write;
  VAR
     file: ARRAY 256 OF CHAR;
     str: Streams.Stream;
     nr: INTEGER;

  PROCEDURE PointLines(str: Streams.Stream): INTEGER;
  (* diese Prozedur ist als gegeben zu betrachten *)

BEGIN

    IF ~AV.GetArg(file) THEN
       nr := PointLines(Streams.stdin);
       Write.String("stdin : ");
       Write.Int(nr ,0);
       Write.Ln;
    ELSE
       REPEAT
         IF ~UF.Open(str,file,0,1,NIL) THEN
            Write.String("Can't open ");
            Write.String(file); Write.Ln
         ELSE
            Write.String(file); Write.String(" : ");
           Write.Int(PointLines(str),4);
         END;
         Write.Ln;
       UNTIL ~AV.GetArg(file);
    END

END Count.
```


Aufgabe 4 (10 Punkte)

<u>Rekursion</u>

Gegeben sei folgendes Programm:

```
MODULE Rekursi;
  IMPORT Write;

  VAR x,y : INTEGER;

  PROCEDURE Rek(n: INTEGER; VAR m: INTEGER): INTEGER;
  BEGIN
    IF n > 0 THEN
      m := m+n;
      Write.String("m = "); Write.Int(m,0); Write.Ln;
      RETURN n * Rek(n-1,m);
    ELSE
      RETURN 1
    END;
  END Rek;

BEGIN
  x:=1;
  y := Rek(6,x);
  Write.String("x = "); Write.Int(x, 0); Write.Ln;
  Write.String("y = "); Write.Int(y, 0); Write.Ln;
END Rekursi.
```

Geben Sie **exakt** an, was dieses Programm an die Standardausgabe liefert!

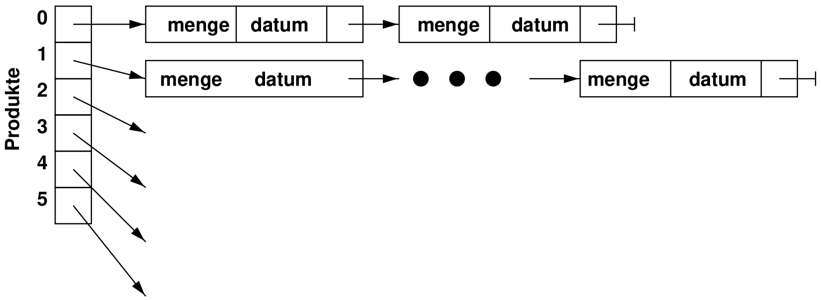
m = 7
m = 12
m = 16
m = 19
m = 21
m = 22
x = 22
y = 720

Aufgabe 5 (10 Punkte)

Dynamische Datenstrukturen

Im Wareneingang einer Firma werden für eine fest vorgegebene Anzahl **N** von Produkten die Wareneingänge verwaltet. Für die einzelnen Produkte wird bei jedem Lieferdatum (Form: tt.mm.yyyy) die Liefermenge (Stückzahl) festgehalten; die Lieferungen für die einzelnen Produkte können unterschiedlich oft erfolgen.

Dazu hat sich jemand folgende Datenstruktur überlegt:



Geben Sie für diese Datenstruktur alle notwendigen Typvereinbarung in Oberon an, so dass eine Variable `wareneingang` vereinbart werden kann.

Lösung:

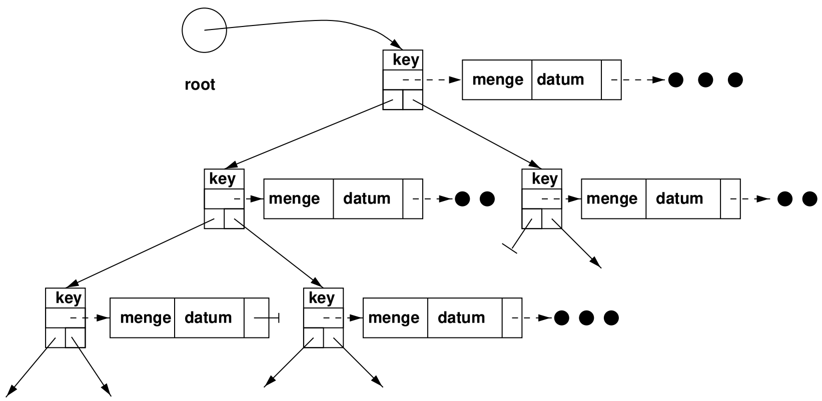
Lösung:

```
CONST N = 6;

TYPE
     Plist     = POINTER TO ListRec;
     ListRec   = RECORD
                    menge: INTEGER;
                    Datum: DATUM;
                (* oder: Datum: ARRAY 11 OF CHAR *)
                    next: Plist;
                  END;
     DATUM      = RECORD
                    tag, monat, jahr: INTEGER;
                  END;

VAR
     wareneingang: ARRAY N OF Plist;
```


Aufgabe 6 (10 Punkte)

Dynamische Datenstrukturen (Listen)

Es sei wie in Aufgabe 5 wieder der Wareneingang einer Firma gegeben. Die Anzahl der Produkte ist jetzt variabel. Die für ein Produkt geführten Daten seien wieder Liefermenge und Datum. Dies kann in einem sortierten Binärbaum verwaltet werden:



Der Eintrag `key` ist die eindeutige Identifikationsnummer des Produkts, die Angabe `menge` ist die Stückzahl der zum durch die Angabe `datum` definierten Zeitpunkt gelieferten Menge.

Diese Struktur wird durch folgende Typvereinbarungen beschrieben:

```
TYPE
  Datum = RECORD
         tag, monat, jahr: INTEGER
          END;

  ProduktBaum = POINTER TO ProduktBaumRec;
  ProduktListe = POINTER TO ProduktListeRec;
  ProduktBaumRec = RECORD
               key: INTEGER;
               next: ProduktListe;
               li, re: ProduktBaum;
                   END;
  ProduktListeRec = RECORD
                menge: INTEGER;
                datum: Datum;
                next: ProduktListe
                    END;
```

Fortsetzung nächste Seite!

Schreiben Sie eine **rekursive** Oberon-Funktion, die als Rückgabewert liefert, wieviel **verschiedene** Produkte im Baum enthalten sind.

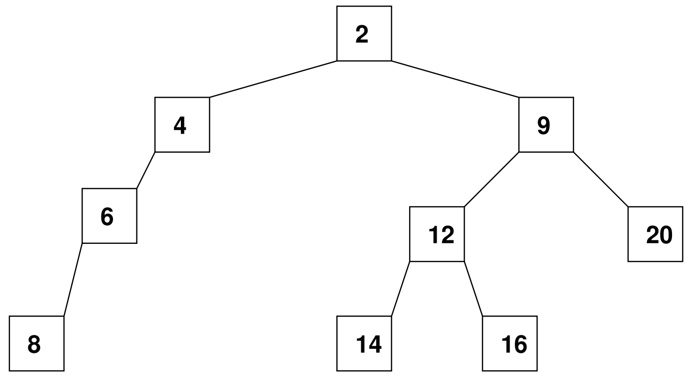
Lösung:

```
PROCEDURE AnzahlProdukte(w: ProduktBaum): INTEGER;
     VAR n: INTEGER;
     list: ProduktListe;
BEGIN
     IF w = NIL THEN RETURN 0; END;
     n := 0;
     (* Aufsummieren der Elemente in den Listen war NICHT
      * ausdruecklich verlangt!!
     list := w.next;
     WHILE list#NIL DO
          INC(n, list.menge); list := list.next;
     END;
     *)
     RETURN n + AnzahlProdukte(w.li) + AnzahlProdukte(w.re);

END Anzahl;
```


Aufgabe 7 (12 Punkte)

Binärbäume

Gegeben sei folgender binärer Baum, bei dem ganze Zahlen als Schlüssel dienen:

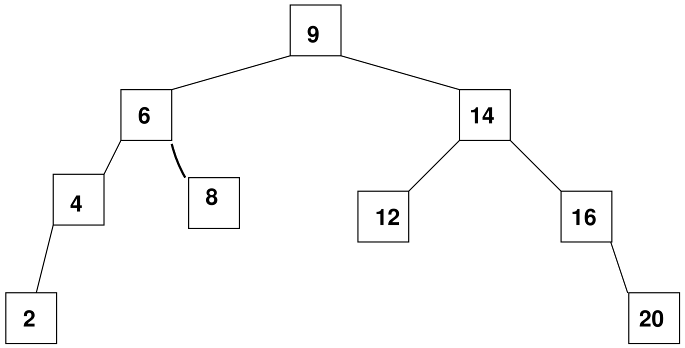


(*a*) 3 Punkte
Bitte kreuzen Sie das jeweils zutreffende an:

Aussage	Richtig	Falsch
Der Baum ist bezüglich der Höhe ausgeglichen.		X
Der Baum ist nach dem Gewicht ausgeglichen.		X
Der Baum ist nach **Preorder** sortiert.	X	

(*b*) 5 Punkte

Zeichnen Sie den Baum mit den gleichen Schlüsselinhalten wie in a), so dass er nach Inorder sortiert und nach Gewicht ausgeglichen ist.



(*c*) 4 Punkte

Geben Sie die Schlüssel des Ursprungsbaumes in **Postorder** an.

Lösung: 8 6 4 14 16 12 20 9 2