Vordiplom Wirtschaftswissenschaften
Allgemeine Informatik I
SS 2001
25. April 2001
Beispiellösung

Bearbeitungszeit: 120 Minuten

Name:
Vorname:
Matrikelnummer:

Aufgabe	Punkte	Bewertung
1	**18**	
a)	6	
b)	6	
c)	6	
2	**18**	
a)	2	
b)	2	
c)	4	
d)	2	
e)	4	
f)	4	
3	**14**	
a)	4	
b)	6	
c)	4	
4	**6**	
5	**8**	
6	**10**	
7	**6**	
8	**10**	
Summe:	**90**	
Note:		

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!
Viel Erfolg!!!

Aufgabe 1 **(18 Punkte)**

Die formale Sprache L über dem Vokabular $V = \{0, 1\}$ sei verbal wie folgt definiert:
$L = \{$ alle mit '0' endenden, nichtleeren Zeichenfolgen über V, bei denen die Anzahl der unmittelbar aufeinander folgenden Einsen '1' gerade und die Anzahl der unmittelbar aufeinander folgenden Nullen '0' ebenfalls gerade ist $\}$

a) 6 Punkte
Beschreiben Sie L mittels einer regulären Grammatik.

Lösung:
S ist in der folgenden regulären Grammatik das Startsymbol.

$$\begin{aligned} S &\rightarrow 0A \mid 1C \\ A &\rightarrow 0 \mid 0B \\ B &\rightarrow 0A \mid 1C \\ C &\rightarrow 1D \\ D &\rightarrow 0A \mid 1C \end{aligned}$$

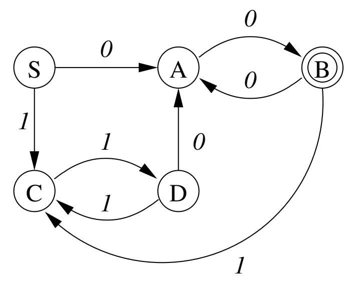
b) 6 Punkte
Beschreiben Sie L mittels eines regulären Ausdrucks.

Lösung:

$$(00|11) * 00$$

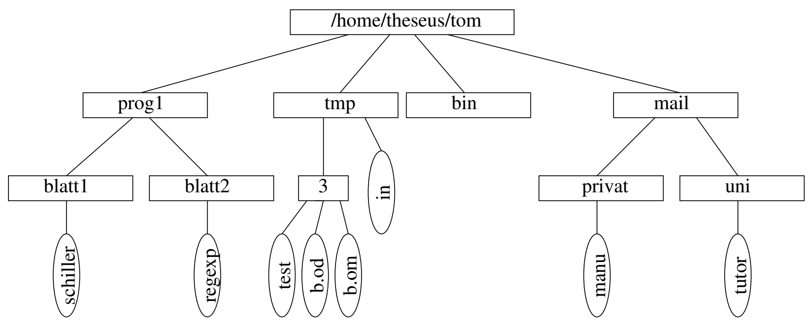
c) 6 Punkte
Beschreiben Sie L mittels eines deterministischen endlichen Automatens (in grafischer Darstellung).

Lösung:
Beim folgenden Automaten ist S der Startzustand und alle nicht angegebenen Übergänge führen in den (impliziten) Fehlerzustand.



Aufgabe 2 **(18 Punkte)**

Gegeben ist im Folgenden die grafische Darstellung eines Auszugs aus der Verzeichnisstruktur des Benutzers *tom*, der ebenfalls an der Vorlesung Allgemeine Informatik I für WiWis teilnimmt. (Die Rechtecke stellen Katalogdateien und die Ellipsen reguläre (gewöhnliche) Dateien dar):



a) 2 Punkte
Der Benutzer *tom* bekommt von seinem Tutor regelmäßig Mails mit den Punkten, die er für das aktuelle Blatt erhalten hat. Diese Mails möchte er in einem eigenen Katalog aufbewahren. Also legt er sich von seinem Heimatkatalog aus im Katalog *mail/uni* einen Unterkatalog *prog* für diese Mails an, ohne dabei seinen Heimatkatalog zu verlassen. Welches Kommando tippt er ein?

Lösung:
```
theseus$ mkdir mail/uni/prog
```

b) 2 Punkte
tom befindet sich weiterhin in seinem Heimatkatalog. Er hat schon etwas für das 3. Übungsblatt gemacht, das sich momentan in */home/theseus/tom/tmp/3* befindet. Diesen Katalog möchte er in den Katalog *prog1* verschieben und zugleich in *blatt3* umbenennen, ohne dabei seinen Heimatkatalog zu verlassen. Geben Sie das notwendige Kommando an!

Lösung:
```
theseus$ mv tmp/3 prog1/blatt3
```


c) 4 Punkte
Nun wechselt er vom Heimatkatalog in den Katalog für Blatt 3 und kopiert danach alle Dateien, die zur Erzeugung des ausführbaren Programms zu *b.om* notwendig sind, in den *tmp*-Katalog im Heimatverzeichnis. Geben Sie die hierzu erforderlichen Kommandos an!

Lösung:
```
theseus$ cd prog1/blatt3
theseus$ cp b.o? ~/tmp
```

d) 2 Punkte
tom wechselt nun in seinen *tmp*-Katalog (bei dem von ihm verwendeten Kommando spielt sein momentaner Katalog keine Rolle) und erzeugt das ausführbare Programm. Dieses startet er, wobei dieses seine Eingabe aus der Datei *in* beziehen soll. Die Ausgabe soll in die Datei *out* umgeleitet werden. Geben Sie die erforderlichen Kommandos an.

Lösung:
```
theseus$ cd ~/tmp
theseus$ mmo -c makefile b.o?
theseus$ make
theseus$ b < in > out
```

e) 4 Punkte
Bei der Ausführung des Programms trat ein Laufzeitfehler auf und jetzt befindet sich eine Datei mit Namen *core* im *tmp*-Katalog. Was muss er eingeben, um diese Datei und die Datei *out* wieder loszuwerden? Danach reicht es ihm und er möchte zusätzlich auch *alle* vom Compiler erzeugten Dateien entfernen. Was muss er dazu noch eingeben?

Lösung:
```
theseus$ rm core out
theseus$ make realclean
```

f) 4 Punkte
Vor einiger Zeit hat er das (eigene) Kommando `longrun` im Hintergrund gestartet. Danach wurde von der Shell die Zahl `4711` ausgegeben und der Shell-Prompt erschien wieder. Da der Prozess nun schon eine ganze Weile läuft, besteht die Vermutung, dass eine Endlos-Schleife enthalten ist; daher möchte er ihn beenden. Was kann er tun?

Lösung:
Er gibt auf der Kommandozeile `kill -9 4711` ein.

Aufgabe 3 **(14 Punkte)**

Die Datei *pruefungen* enthält eine Datenbank der von Prof. Schweiggert durchgeführten Prüfungen. Jede Zeile der Datei entspricht einem Datensatz. Jeder Datensatz besteht wiederum aus den Feldern Matrikelnummer des geprüften Studenten, Datum der Prüfung und Note. Diese sind voneinander durch ':' getrennt. Die Matrikelnummer ist eine ganze Zahl. Das Datum liegt im Format *Tag* '.' *Monat* '.' *Jahr* vor. Als Note können nur die Werte 1.0, 1.3, 1.7, 2.0, 2.3, 2.7, 3.0, 3.3, 3.7, 4.0 und 5.0 auftreten. Beispiel (einige Zeilen aus der Datei *pruefungen*):

```
347544:12.3.2001:1.0
56543211:03.7.1999:3.3
4711:2.1.2001:1.7
```

Sie können davon ausgehen, dass die Datei *pruefungen* genau das beschriebene Format besitzt (und in keinem Feldwert das Zeichen ':' vorkommt) und kein Feld leer ist.

a) 4 Punkte
Beschreiben Sie in Worten, was die folgende Kommandozeile liefert:
`egrep '^[^:]*1:' pruefungen | wc -l`

Lösung:
Es werden alle Datensätze der Datei *pruefungen*, bei denen die Matrikelnummer mit '1' endet, gezählt und diese Anzahl wird auf die Standardausgabe geschrieben.

b) 6 Punkte
Geben Sie eine Kommandozeile an, die alle Datensätze der Datei *pruefungen* auf die Standardausgabe ausgibt, bei denen die Note 2.7 oder besser ist. (Gültige Werte für die Note sind oben angegeben.)

Lösung:
`theseus$ egrep ':[12]\..$' pruefungen`

c) 4 Punkte
Manchmal kommt es vor, dass bei der Eingabe der Daten ein Fehler unterläuft. Matrikelnummern sollten (in unserem vereinfachten Fall) immer sechsstellig sein. Sie sollen nun eine Kommandozeile angeben, die alle Zeilen der Datei *pruefungen* ausgibt, die eine nicht sechsstellige Matrikelnummer enthalten. (Hier liegt wohl ein Fehler bei der Eingabe vor.)

Lösung:
`theseus$ grep -v '^[0-9]\{6\}:' pruefungen`

Aufgabe 4 **(6 Punkte)**

Definieren Sie eine Oberon-Datenstruktur, um einen Prüfungs-Datensatz so *ähnlich* wie in Aufgabe 3, d. h. Matrikelnummer des geprüften Studenten, Datum und Uhrzeit der Prüfung, Note und der Name des Prüfers zu speichern. Verwenden Sie hierzu Konstanten- und Typdefinitionen. Die Matrikelnummer ist eine ganze Zahl. Tag, Monat und Jahr des Datums sowie Stunde und Minute der Uhrzeit sollen über die Datenstruktur direkt („einstufig") zugreifbar sein. Die Note kann als reelle Zahl betrachtet werden. Der Name des Prüfers bestehe aus höchstens 60 Zeichen.
Lösung:

```
CONST MaxName  = 60;
TYPE  Datum    = RECORD
                    tag,
                    monat,
                    jahr  : INTEGER;
                 END;
      Uhrzeit  = RECORD
                    stunde,
                    minute : INTEGER;
                 END;
      Name     = ARRAY MaxName + 1 OF CHAR;
      Pruefung = RECORD
                    matrikelNummer : INTEGER;
                    datum          : Datum;
                    uhrzeit        : Uhrzeit;
                    note           : REAL;
                    pruefer        : Name;
                 END;
```

Aufgabe 5 **(8 Punkte)**

Der Vorstand eines Unternehmens benötigt für die einzelnen Produkte, deren Anzahl bekannt und fest ist, die jährlichen Verkaufsmengen (Stückzahlen) für eine fest vorgegebene Anzahl zurückliegender Jahre. Die Produkte seien von 0 an durchnummeriert. Dazu hat er sich eine Datenstruktur überlegt, die in Oberon wie folgt aussieht:

```
CONST
  MaxJahre     = 5;
  AnzahlProdukte = 25;

TYPE
  JahresMenge =   RECORD
                    jahr, menge: INTEGER;
                  END;

  ProduktZahlen = RECORD
                    anzahl: INTEGER;
                   (* anzahl: Anzahl der vorliegenden
                    * Verkaufsjahre eines Produktes
                    *)
                    jahresMenge: ARRAY MaxJahre OF JahresMenge
                  END;

  GesamtTabelle = ARRAY AnzahlProdukte OF Produktzahlen;
```

Es ist davon auszugehen, dass für alle Produkte (*AnzahlProdukte*) Verkaufsmengen vorliegen, dass aber nicht bei jedem davon genau *MaxJahre* jährliche Verkaufsmengen bekannt sind, sondern gegebenenfalls auch weniger. Die Komponente *anzahl* in *ProduktZahlen* gibt also an, wieviele Elemente (von 0 an!) in der Komponente *jahresMenge* tatsächlich belegt sind.

Ihre Aufgabe:

Schreiben Sie eine Prozedur, die als Parameter ein Objekt vom Typ *GesamtTabelle* sowie eine Produktnummer (im Bereich 0..AnzahlProdukte-1) erhält und die zugehörigen Verkaufsmengen in der Form **jahr:menge** an die Standardausgabe schreibt.
(Bitte benutzen Sie für Ihre Lösung die folgende Leerseite!)

Lösung:

```
PROCEDURE WriteTable(t: GesamtTabelle; pId: INTEGER);
  VAR i: INTEGER;
BEGIN
  i:=0;
  WHILE i < t[pId].anzahl DO
    Write.Int(t[pId].jahresMenge[i].jahr,4); Write.Char(":");
    Write.Int(t[pId].jahresMenge[i].menge,4); Write.Ln;
    INC(i);
  END
END WriteTable;
```

Aufgabe 6 **(10 Punkte)**

Schreiben Sie eine Oberon-Prozedur **UpAndDown**, die als Parameter eine Folge von Zeichen bekommt und die darin enthaltenen Kleinbuchstaben in Großbuchstaben und die Großbuchstaben in Kleinbuchstaben umwandelt. Die so erzeugte Zeichenfolge soll dem Aufrufer der Prozedur zurückgeliefert werden.

Lösung:

```
MODULE MyString;
  IMPORT Write;
  CONST Dim = 30;
  TYPE  CharSequence = ARRAY Dim OF CHAR;
  VAR   w: CharSequence;
(* HIER SOLLTE IHRE PROZEDUR STEHEN: *)
  PROCEDURE UpAndDown(VAR w: ARRAY OF CHAR);
    VAR i: INTEGER;
  BEGIN
    i := 0;
    WHILE (i < LEN(w)) & (w[i] # 0X) DO
      CASE w[i] OF
      | "a" .. "z" : w[i] := CAP(w[i]);
      | "A" .. "Z" : w[i] := CHR(ORD(w[i]) - ORD("A") + ORD("a"));
      ELSE
      END;
      INC(i);
    END
  END UpAndDown;
(*DER PLATZ BIS HIER MUESSTE GEREICHT HABEN *)
BEGIN
  w := "Franz Schweiggert!";
  UpAndDown(w);
  Write.String(w); Write.Ln
END MyString.
```


Aufgabe 7 **(6 Punkte)**

Gegeben ist das folgende Programm:

```
MODULE Obscure;

   IMPORT Write, Conclusions;

   VAR a, b, c : INTEGER;

   PROCEDURE Obscure(VAR x, y : INTEGER; z : INTEGER);
   BEGIN
      z := x; x := y; y := z;
      a := z;
      z := b;
   END Obscure;

BEGIN
   Write.String(" a | b | c "); Write.Ln;
   Write.String("---+---+---"); Write.Ln;

   a := 2; b := 3; c := 4;

   Write.Char(" ");
   Write.Int(a, 0); Write.String(" | ");
   Write.Int(b, 0); Write.String(" | ");
   Write.Int(c, 0); Write.Ln;

   Obscure(a, b, c);

   Write.Char(" ");
   Write.Int(a, 0); Write.String(" | ");
   Write.Int(b, 0); Write.String(" | ");
   Write.Int(c, 0); Write.Ln;

   Obscure(c, b, a);

   Write.Char(" ");
   Write.Int(a, 0); Write.String(" | ");
   Write.Int(b, 0); Write.String(" | ");
   Write.Int(c, 0); Write.Ln;
END Obscure.
```

Tragen Sie in folgender Tabelle die vom Programm erzeugten Ausgaben für die Werte der Variablen a, b und c ein.

Lösung:

a	b	c
2	3	4
2	2	4
4	4	2

Aufgabe 8 **(10 Punkte)**

Bei der Übertragung von Daten können sich Fehler „einschleichen". Daher verwendet man bei der Übertragung oft Verfahren, um solche Fehler zu erkennen oder – noch besser – zu korrigieren. Ein sehr einfaches Verfahren, um Übertragungsfehler zu erkennen, ist die Übermittlung der Quersumme der übertragenen Daten. Bei den empfangenen Daten kann dann die Quersumme berechnet werden und diese mit der empfangenen Quersumme verglichen werden. Stimmen diese nicht überein, so liegt garantiert ein Übertragungsfehler vor. Sind beide identisch, so kann keine Aussage getroffen werden. Schreiben Sie dazu nun die folgende Oberon-Funktionsprozedur

```
PROCEDURE Quersumme(VAR summe: INTEGER;
                    zahl : ARRAY OF CHAR): BOOLEAN;
```

Diese berechnet die Quersumme *summe* der Dezimalzahl, die in *zahl* als Feld von Dezimalziffern übergeben wird. Der Rückgabewert ist *TRUE* genau dann, wenn *zahl* eine Dezimalzahl ist, das heißt nur aus Ziffern besteht. (Andernfalls ist der Rückgabewert *FALSE*. In diesem Fall ist der Inhalt von *summe* nicht von Bedeutung.) Der String *zahl* muss nicht durch ein Null-Byte terminiert sein. In diesem Fall bilden alle Zeichen des Arrays *zahl* den String.

Lösung:

```
PROCEDURE Quersumme(VAR summe: INTEGER;
                    zahl : ARRAY OF CHAR): BOOLEAN;
   VAR i : INTEGER;
BEGIN
   summe := 0; i := 0;
   WHILE (i < LEN(zahl)) & (zahl[i] # 0X) DO
      CASE zahl[i] OF
      | "0" .. "9" : summe := summe +
                              ORD(zahl[i]) - ORD("0");
      ELSE
         RETURN FALSE;
      END;
      INC(i);
   END;
   RETURN TRUE;
END Quersumme;
```