Vordiplom Wirtschaftswissenschaften
Allgemeine Informatik I
SS2003 – 28.4.2003
Beispiellösung – **Gruppe B**

Bearbeitungszeit: 120 Minuten

Name:	
Vorname:	
Matrikelnummer:	

Aufgabe	Punkte	Bewertung
1	**14**	
a)	2	
b)	2	
c)	2	
d)	2	
e)	3	
f)	3	
2	**16**	
a)	4	
b)	4	
c)	4	
3	**12**	
a)	4	
b)	4	
c)	4	
4	**6**	
5	**12**	
6	**8**	
7	**6**	
8	**16**	
a)	8	
b)	8	
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 **(16 Punkte)**

Sie möchten in Ihrem Heimatverzeichnis Ordnung schaffen.

(a) **2 Punkte**

Im Unterverzeichnis **allerlei** sind jede Menge Dateien und Unterkataloge, die sie allesamt löschen wollen – das Verzeichnis **allerlei** soll also komplett geleert werden, aber als Verzeichnis als solches erhalten bleiben! Sie können davon ausgehen, dass die Dateinamen nur aus Buchstaben und Ziffern bestehen!
Geben Sie die Unix-Befehlszeile an, mit der Sie dies bewerkstelligen (Ihr Arbeitskatalog ist Ihr Heimatverzeichnis – Sie dürfen dieses nicht verlassen)!
Lösung:

```
thales$ rm -r allerlei/*
```

(b) **2 Punkte**

Sie befinden sich nach wie vor in Ihrem Heimatverzeichnis. Legen Sie einen Unterkatalog **texte** an und verschieben Sie alle Dateien mit Endung **.txt** im Heimatverzeichnis in diesen Unterkatalog ohne den Heimatkatalog zu verlassen! Geben Sie die dazu notwendigen Unix-Befehlszeilen an!
Lösung:

```
thales$ mkdir texte
thales$ mv *.txt texte
```

(c) **2 Punkte**

Sie haben in Ihrem Heimatkatalog zwei Dateien namens *namen-1* und *namen-2*. Was ist der Unterschied in der Wirkung der folgenden beiden Kommandozeilen:

```
thales$ cat namen-1 >> namen-2
thales$ cat namen-1 > namen-2
```

Lösung:

1.Zeile: Inhalt von namen-2 wird mit Inhalt von namen-1 **überschrieben**
2.Zeile: Inhalt von namen-1 wird an Inhalt von namen-2 **angefügt**, namen-2 wird somit um Inhalt von namen-1 verlängert

(d) **4 Punkte**

Sie sind inzwischen in den Unterkatalog *sonstiges* gewechselt. Sie geben folgendes Kommando ein:
`thales$ ls -a`
und erhalten als Ausgabe

```
. .. .aloisius aloisius.od  aloisius.od.alt  aloisius.om
aloisius.om-old  makefile
```

Mit der Kommandozeile
`thales$ rm *.*`
wollen Sie auch hier aufräumen. Danach geben Sie wieder ein:
`thales$ ls -a`
Welche Ausgabe erscheint nun?
Lösung:

```
. .. .aloisius makefile
```

(e) **2 Punkte**

In Ihrem Heimatkatalog hatten Sie vor langer Zeit einen Unterkatalog **tmp** angelegt, in dem sich viel „Schrott" angesammelt hat – ausschließlich reguläre Dateien. Deren Namen sind allesamt einbuchstabig. Geben Sie an, wie Sie diejenigen löschen, deren Name aus einem Kleinbuchstabe besteht!
Lösung:

```
thales$ rm [a-z]
```

(f) **4 Punkte**

Ein Programm namens **tue** werde wie folgt gestartet:
`thales$ tue > c 2>b < a`

Beschreiben Sie kurz die Bedeutung der einzelnen Angaben!
Lösung:

Kommando *tue* liest von der Standardeingabe, die von der Shell auf Datei a umgelenkt wurde
Kommando *tue* schreibt auf die Standardausgabe, die von der Shell auf Datei c umgelenkt wurde
Kommando *tue* schreibt Fehlermeldungen u.dgl. auf die Diagnoseausgabe, die von der Shell auf Datei b umgelenkt wurde

Aufgabe 2 **(16 Punkte)**

Über dem Vokabular $V = \{+, 0, 1\}$ ist folgende Sprache definiert:
alle Zeichenfolgen über V,

- die mit einem + (Plus-Zeichen) beginnen und wieder aufhören,
- die Plus-Zeichen nicht unmittelbar nebeneinander vorkommen
- die Länge der Einser-Folgen (unmittelbar nebeneinanderstehenden Einsen (**1**)) eine ungerade Zahl größer oder gleich 1 ist,
- die Länge der Null-Folgen (unmittelbar nebeneinanderstehenden Nullen (**0**)) eine gerade Zahl größer oder gleich 2 ist,
- und bei denen diese 0-er bzw. 1-er Folgen durch ein Plus-Zeichen voneinander getrennt sind.

Beispiele für korrekte Folgen: $+$ (beginnt mit einem $+$ und hört mit einem $+$ auf), $+00 + 0000 + 1 + 00 + 111 + 1+$
Beispiele für nicht korrekte Folgen: $++$ (zwei nebeneinanderstehende $+$), $+0+$ (ungerade 0-Folge)

(a) **8 Punkte**

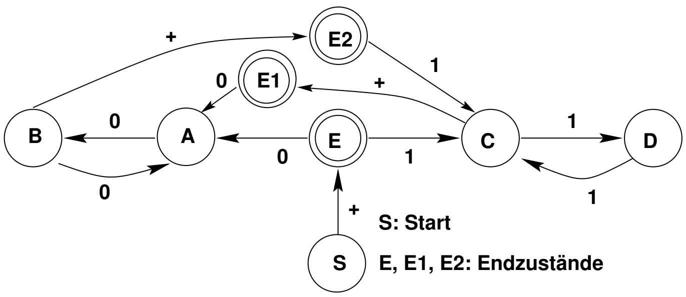
Beschreiben Sie diese Sprache durch einen regulären Ausdruck! Als Metasymbole dürfen die im Skript im Zusammenhang mit Unix-Kommandos dargestellten Symbole verwendet werden – z.B. + (Plus-Zeichen) für die mindestens einmalige Wiederholung!

Lösung: `\+(((00)+|(1(11)*))\+)*`

(b) **8 Punkte**

Beschreiben Sie diese Sprache mit einem grafisch dargestellten endlichen, deterministischen Automaten! Start- und Endzustand (oder Endzustände) sind zu kennzeichnen!

Lösung:



Aufgabe 3 **(12 Punkte)**

In der Datei *score* werden die Ergebnisse eines Wettbewerbs geführt. Die Einträge haben die Form *name***:***points***:***grade* (jeder Eintrag kommt in eine neue Zeile). Für die folgenden Aufgaben habe die Datei *score* den folgenden Inhalt:

```
theo:70:2.3
susi:90:1.0
ivo:0:5.0
ingo:74:2.0
hugo:45:4.0
heiko::
hannah:4:5.0
gigi:3:
gerda:82:1.3
babs:78:1.7
```

(a) **4 Punkte**

Welche Ausgabe liefert folgendes Kommando?

```
egrep ':.*:2\..$'  score
```

Lösung:

```
theo:70:2.3
ingo:74:2.0
```

(b) **4 Punkte**

Welche Ausgabe liefert folgendes Kommando? Zusätzlich ist hier zu beschreiben, **was** diese Ausgabe beschreibt!

```
cat score | egrep -c ':[^:]+$'
```

Lösung:

```
8
```

Bei wievielen Zeilen ist in der letzten Spalte (grade) etwas eingetragen

(c) **4 Punkte**

Welche Ausgabe liefert folgendes Kommando?

```
egrep '^.*:[0-9]:$' < score
```

Lösung:

```
gigi:3:
```

Aufgabe 4 **(6 Punkte)**

Schreiben Sie eine Oberon-Prozedur **Reverse**, die als Parameter eine Folge von Zeichen erhält, die darin enthaltenen Großbuchstaben in die entsprechenden Kleinbuchstaben und darin enthaltenen Kleinbuchstaben in die entsprechenden Großbuchstaben wandelt, die sonstigen Zeichen aber unverändert lässt.

Beispiel für die Verwendung:

```
 VAR str: ARRAY 256 OF CHAR;

s := "UnD WiEdeR EinE KlAUsur gESCHafFT";
Reverse(str);
Write.String(str); (*liefert: uNd wIeDEr eINe kLauSUR GeschAFft*)
```

Lösung:

```
PROCEDURE Reverse(VAR s: ARRAY OF CHAR);
  VAR i: INTEGER;
BEGIN
  i := 0;
  WHILE (i < LEN(s)) & (s[i] # 0X) DO
      IF (s[i] >= "A") & (s[i] <= "Z") THEN
         s[i] := CHR(ORD("a") + ORD(s[i]) - ORD("A"));
      ELSIF (s[i] >= "a") & (s[i] <= "z") THEN
         s[i] := CHR(ORD("A") + ORD(s[i]) - ORD("a"));
      END;
      INC(i);
  END;
END Reverse;
```

Aufgabe 5 **(8 Punkte)**

Schreiben Sie ein **komplettes** Oberon-Programm (nur das **.om**-File), welches von der Standardeingabe einen Text einliest und die Häufigkeiten der darin enthaltenen (Dezimal-) Ziffern ermittelt. In der Ausgabe sollen zeilenweise aufsteigend die Ziffers und dahinter ihre Häufigkeiten erscheinen!
Hinweis: Um die Häufigkeit der 10 Ziffern zu zählen, sollen natürlich nicht 10 verschiedene Zählvariable verwendet werden, ein ARRAY ist hier angebracht!

Lösung:

```
MODULE Histo;
   IMPORT Read, Write, Streams;
   VAR ch: CHAR;
       h: ARRAY 10 OF INTEGER;
       i: INTEGER;
BEGIN
   i := 0;
   WHILE i < 10 DO
     h[i] := 0;
     INC(i);
   END;
   Read.Char(ch);
   WHILE ~ Streams.stdin.eof DO
      IF (ch >= "0") & (ch <= "9") THEN
          INC(h[ORD(ch)-ORD("0")]);
      END;
      Read.Char(ch);
    END;
   i := 0;
   WHILE i < 10 DO
      Write.Int(i,3); Write.String(" : ");
      Write.Int(h[i],0); Write.Ln;
      INC(i);
   END;
END Histo.
```

Aufgabe 6 **(8 Punkte)**

Gegeben ist das folgende Programm:

```
MODULE Obscure;
  IMPORT Write;

  VAR u,v,w: INTEGER;

  PROCEDURE Magic(VAR x: INTEGER; y: INTEGER; VAR z: INTEGER);
  BEGIN
   z:=y; y:=x; x:=z;
  END Magic;

BEGIN
  u :=8; v:=4; w:=1;
  Magic(u,v-w,v);
  Write.Int(u,3); Write.Int(v,3); Write.Int(w,3);Write.Ln;
  Magic(w,v DIV 2,u);
  Write.Int(u,3); Write.Int(v,3); Write.Int(w,3);Write.Ln;
  Magic(v,u*u,w);
  Write.Int(u,3); Write.Int(v,3); Write.Int(w,3);Write.Ln;
  Magic(w,v+v,u);
  Write.Int(u,3); Write.Int(v,3); Write.Int(w,3);Write.Ln;
END Obscure.
```

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

```
  3  3  1
  1  3  1
  1  1  1
  2  1  2
```

Aufgabe 7 **(8 Punkte)**

Gegeben sei folgendes Programm:

```
MODULE Logic;
  IMPORT Write;

  PROCEDURE WhichNumber(x,y: INTEGER): INTEGER;
  BEGIN
     IF (x <= 0) OR (y <= 0) THEN
        IF (x <= -2) & (y >= 2 ) THEN
            RETURN  1
        ELSE
            RETURN 2
        END
     ELSE
        RETURN 3
     END
   END WhichNumber;
BEGIN
  Write.Int(WhichNumber(-1,1),0); Write.Ln;
  Write.Int(WhichNumber(-3,4),0); Write.Ln;
  Write.Int(WhichNumber(1,2),0); Write.Ln;
  Write.Int(WhichNumber(-1,2),0); Write.Ln;

END Logic.
```

Geben Sie an, was dieses Programm ausgibt!
Lösung:

```
2
1
3
2
```

Aufgabe 8 **(16 Punkte)**

Als privater Aktienanleger wollen Sie etwas Übersicht in Ihre Aktivitäten bringen. Dazu beschreiben Sie in einer Datei Ihre Anlagen wie folgt:

```
<Nr> <Kaufpreis> <Kurs> <Stueckzahl>
```

`Nr` ist eine ganze Zahl, mit der Sie eine Anlage identifizieren. `Kaufpreis` ist der Preis, den Sie für eine Aktie bezahlt haben, `Kurs` ist der aktuelle Kurs dieser Aktie und mit `Stueckzahl` halten Sie fest, wieviele Aktien Sie in dieser Anlage besitzen. Als Oberon-Datenstruktur haben Sie dies wie folgt beschrieben:

```
CONST Anzahl = 100;
TYPE
    AnlageDaten = RECORD
                        nr: INTEGER;
                        ePreis, kurs: REAL;
                        st: INTEGER;
                    END;
VAR anlage: ARRAY Anzahl OF AnlageDaten;
```

(a) **8 Punkte**

Schreiben Sie eine Oberon-Prozedur **ReadData**, die von der Standardeingabe solche Anlagedaten in ein ARRAY mit Elementen vom Typ **AnlageDaten** einliest (aktueller Parameter könnte dann obige Variable **anlage** sein) und die Anzahl der eingelesenen Datensätze als Rückgabewert liefert.
Beispieleingabe:

```
1 31.50 26.30 200
6 36.00 24.50  80
4 72.00  7.40  40
2 65.90 82.00  10
```

Sie können davon ausgehen, dass die Eingabedaten genau in dieser Form korrekt und vollständig kommen!

Lösung auf der nächsten Seite

Lösung von 8a)

```
PROCEDURE ReadData(VAR daten: ARRAY OF AnlageDaten):INTEGER;
  VAR i: INTEGER;
BEGIN
  i := 0;
  Read.Int(daten[i].nr);
  Read.Real(daten[i].ePreis);
  Read.Real(daten[i].kurs);
  Read.Int(daten[i].st);

  WHILE (~ Streams.stdin.eof) & (i < LEN(daten)-1) DO
    INC(i);
    Read.Int(daten[i].nr);
    Read.Real(daten[i].ePreis);
    Read.Real(daten[i].kurs);
    Read.Int(daten[i].st);
  END;

  IF Streams.stdin.eof THEN
     RETURN i
  ELSE
     RETURN i + 1
  END;
END ReadData;
```

(b) **8 Punkte**

Schreiben Sie eine Oberon-Prozedur, die ein ARRAY mit solchen Anlagedaten erhält und die `Id` der Anlage ermittelt, die aktuell den größten Verlust in Ihrer "Sammlung" darstellt – dieser Verlust soll ebenfalls ermittelt werden! Durch einen weiteren Parameter wird angegeben, wieviele Positionen von Anfang her tatsächlich belegt sind!

Eine Anlage hat einen Verlust, wenn der Kaufpreis größer als der aktuelle Kurs ist!

Lösung:

```
PROCEDURE MaxVerlust(u: ARRAY OF AnlageDaten; anz: INTEGER;
                     VAR nr: INTEGER; VAR verlust: REAL);
    VAR i: INTEGER; tmp: REAL;
BEGIN
    IF (anz <= 0) OR (anz >= LEN(u))
    THEN nr := -1; RETURN
    END;

    verlust:= (u[0].ePreis - u[0].kurs) * u[0].st;

    nr := u[0].nr;

    i := 1;
    WHILE i < anz DO
      tmp := (u[i].ePreis - u[i].kurs) * u[i].st;

      IF verlust < tmp THEN
         verlust := tmp;
         nr := u[i].nr;
      END;
      INC(i);
    END;
 END MaxVerlust;
```