Vordiplom Wirtschaftswissenschaften

Lösung zu Allgemeine Informatik I

WS 1999/2000

28. Februar 2000

Aufgabe		Punkte	Bewertung
1	a)	4	
	b)	6	
	c)	4	
2	a)	4	
	b)	4	
	c)	4	
3	a)	2	
	b)	4	
	c)	4	
	d)	4	
4	a)	8	
	b)	12	
5		6	
6		14	
Summe		**80**	

☞ Bearbeitungszeit: 120 Min.

Aufgabe 1 (14 Punkte)

Die formale Sprache **L** über dem Vokabular **V = {0,1}** sei verbal wie folgt definiert:

L = { alle mit einer '0' beginnenden Zeichenfolgen aus V^*,
bei denen die Anzahl der unmittelbar nebeneinander stehenden
Nullen ungerade ist, die Anzahl der
Einsen jedoch beliebig ist }

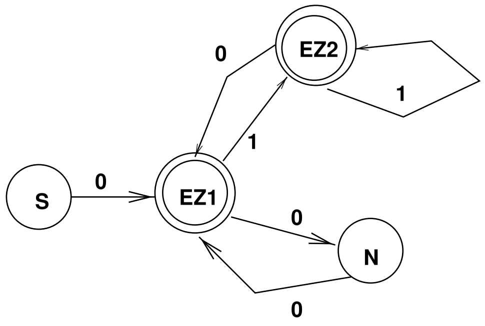
(*a*) 4 Punkte
Beschreiben Sie L mittels einer regulären Grammatik!

S → 0X
X →
X → 0N
N → 0X
X → 1Y
Y → 1Y
Y →
Y → S

(*b*) 6 Punkte
Beschreiben Sie L mittels eines regulären Ausdrucks!

0(00*)(11*[0(00)*])*1*

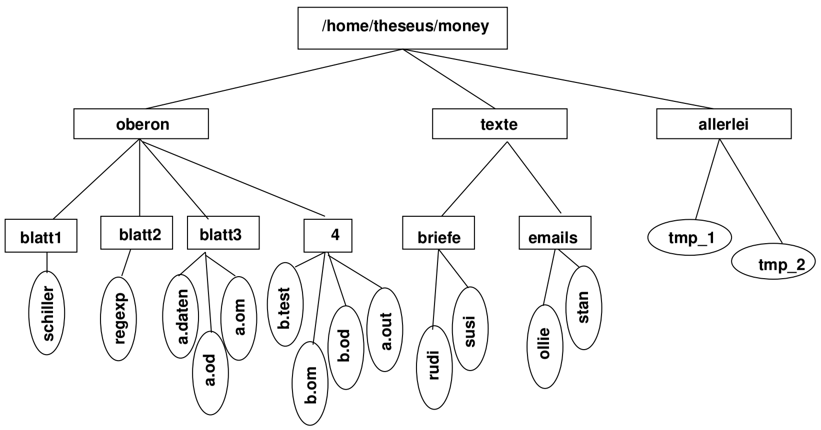
(*c*) 4 Punkte
Beschreiben Sie L mittels eines endlichen deterministischen Automaten (in grafischer Darstellung)!



nicht definierte Übergänge: Fehlerzustand

Aufgabe 2 (12 Punkte)

Gegeben sei in grafischer Darstellung ein Auszug aus der Verzeichnisstruktur des Hörers der Vorlesung "Allgemeine Informatik I" mit dem Login-Namen *money* (Die Rechtecke stellen Katalogdateien, die Ovale reguläre Dateien dar):



(*a*) 4 Punkte
money hat sich angemeldet und ist in den Katalog *allerlei* gewechselt. Er hat für seine Übungsaufgaben im Unterkatalog *oberon* für jedes Blatt einen eigenen Unterkatalog angelegt. Als Name des Unterkatalogs für die ersten drei Blätter hat er *blatt1*, *blatt2* und *blatt3* gewählt, als Name des Unterkatalogs für das vierte Blatt hat er nur die Blattnummer gewählt. Er möchte nun den Katalog *4* in *blatt4* umbenennen. Geben Sie das notwendige Kommando an (Sie stehen immer noch im Katalog *allerlei* und dürfen diesen nicht verlassen!):

```
theseus$ mv ../oberon/4 ../oberon/blatt4
```

(*b*) 4 Punkte
Er möchte nun die Datei *a.out* löschen (Arbeitskatalog ist immer noch *allerlei*). Geben Sie ihm die notwendigen Kommandos (Sie dürfen es machen, wie Sie wollen) - die Zahl der Kommandozeilen, die im folgenden angegeben ist, hat nichts mit der Zahl der notwendigen Kommandos zu tun!

```
theseus$ rm ../oberon/blatt4/a.out
theseus$ ____________________
theseus$ ____________________
```


(*c*) 4 Punkte
Das im Rahmen der 3. Übungsaufgabe erstellte Programm möchte er noch einmal zu Demonstrationszwecken übersetzen, die Eingaben für das Programm kommen via Datei-Umlenkung aus der Datei *a.daten*. Alles, was nach der Übersetzung nicht mehr benötigt wird, soll nach Ausführung des Programms wieder entfernt werden. Geben Sie die notwendigen Kommandos an (Sie können dabei beliebig die Kataloge wechseln)! Die Zahl der Kommandozeilen, die im folgenden angegeben ist, hat wieder nichts mit der Zahl der notwendigen Kommandos zu tun!

```
theseus$ cd ../oberon/blatt3
theseus$ mmo >makefile
theseus$ make
theseus$ a < a.daten
theseus$ make clean
theseus$ ____________________
theseus$ ____________________
theseus$ ____________________
theseus$ ____________________
theseus$ ____________________
```

Aufgabe 3 (14 Punkte)

(*a*) 2 Punkte
Was versteht man im UNIX-Jargon unter einem **Filter**?

Ein Programm, das von der Standardeingabe (stdin) liest, die Ergebnisse auf auf die Standardausgabe (stdout) schreibt und Fehler- / Diagnosemeldungen an die Diagnoseausgabe (stderr) liefert

(*b*) 4 Punkte
Sie haben ein Programm namens **doit**, das seine Eingabe von der Tastatur erhält, seine Ausgabe auf den Bildschirm und Fehler-/Diagnosemeldungen ebenfalls auf den Bildschirm schreibt. Wie können Sie das Programm starten, so dass es seine Eingabe aus der Datei `input`, seine Ausgabe in die Datei `output` und seine Fehler-/Diagnosemeldungen in die Datei `errors` schreibt?

```
theseus$ doit <input >output 2>errors
```

(*c*) 4 Punkte
Was leistet folgende Kommandozeile?

```
theseus$ cat file | egrep '^.[ABC][^0-9]*$' | sort
```

sortierte Ausgabe der Zeilen aus "file", die mit einem beliebigen Zeichen beginnen, auf das ein Buchstabe "A", "B" oder "C" folgt, nachdem bis zum Zeilenende keine Ziffer kommt.

(*d*) 4 Punkte
Sie wollen wissen, wie viele Einträge in Ihrem Home-Directory mit einem Punkt beginnen und Katalogdateien bezeichnen. Folgende Angaben können Sie zur Lösung der Aufgabe benutzen:

- `ls -al` listet alle Einträge (auch die mit einem Punkt beginnen) im Langformat auf, die Namen stehen dabei jeweils am Zeilenende, bei Katalogdateien ist das erste Zeichen in der jeweiligen Ausgabezeile ein "d".

  Beispiel für eine Ausgabe von ls -al:

  ```
  drwxr-xr-x   6 swg      sai          512 Nov 24  1995 .NeXT
  -rw-rw-r--   1 swg      sai         2181 Jul 19  1999 .gv
  drwxr-sr-x   2 swg      sai          512 Sep 19  1997 Mail
  -rw-rw-rw-   1 swg      sai        42586 Feb 19 15:40 Witze.htm
  ```

- `egrep` kann "prima" (insbesondere über reguläre Ausdrücke) suchen
- `wc -l` kann Zeilen zählen.

Lösung (eine Kommandozeile):

```
theseus$ ls -al | egrep '^d.*[    ]\.[^    ]*$' | wc -l
```

Aufgabe 4 (20 Punkte)

Zur Produktion eines neuen Produkts liege eine mathematische Funktion zur mengenabhängigen Kostenbestimmung vor (Fixkosten, variable Kosten), ebenso eine Funktion zur mengen- und stückpreis-abhängigen Erlösbestimmung. Beide Funktionen sind im folgenden jeweils als Oberon-Funktions-Prozeduren implementiert (der Prozedurrumpf ist nicht angegeben, um keine falschen Annahmen über den Kurvenverlauf zu implizieren). Sie können allerdings davon ausgehen, dass die Funktionen die Eigenschaften haben, die die gestellten Aufgaben lösbar machen.

```
PROCEDURE Kosten(menge: INTEGER):REAL;
PROCEDURE Erloes(menge: INTEGER; stueckPreis: REAL):REAL;
```

(*a*) 8 Punkte

Den *break even point* hat das Management wie folgt definiert: Abweichung der Kosten vom Erlös um weniger als 1% von den Kosten. Der Stückpreis ist nicht variabel, also fest vorgegeben!

Schreiben Sie eine Oberon-Funktions-Prozedur, die für eine als Parameter übergebene Stückzahl und einen als Parameter übergebenen Stückpreis TRUE liefert, wenn diese obiges Kriterium erfüllen, ansonsten FALSE liefert.

```
PROCEDURE IsBreakEvenPoint(menge: INTEGER; stueckPreis: REAL): BOOLEAN;
BEGIN
    RETURN ABS(Kosten(menge) - Erloes(menge,stueckpreis)) < 0.01*Kosten(menge)
END IsBreakEvenPoint;
```


(*b*) 12 Punkte

Zur Berechnung des *break even point* (wieder bei festem Stückpreis) können Sie die Funktion aus a) benutzen, auch wenn Sie diese Teilaufgabe nicht gelöst haben. Es sei bekannt, dass dieser Punkt eindeutig existiert und zwischen einer Mengenausbringung (produzierte Stückzahl) **m1** und **m2** liegt. In Anlehnung an den Algorithmus zur binären Suche aus der Vorlesung sollen Sie nun diesen Punkt bestimmen. Der Modul-Rahmen sei wie folgt bereits vorgegeben:

```
MODULE BreakEvenPoint;
   IMPORT Read, Write;
   VAR m1,m2,mitte: INTEGER;
       preis: REAL; gefunden: BOOLEAN;
   PROCEDURE Kosten(m: INTEGER):REAL;
   (* dieser Teil ist unwichtig*)
   END Kosten;
   PROCEDURE Erloes(m: INTEGER; preis: REAL):REAL;
   (* dieser Teil ist unwichtig*)
   END Erloes;
   PROCEDURE IsBreakEvenPoint(m: INTEGER; preis: REAL): BOOLEAN;
   (* von Teilaufgabe a) *)
   END IsBreakEvenPoint;
BEGIN
     m1 := 100; m2 := 1000; preis := 1.50;
     gefunden := FALSE;
(*-------------- hier geht's los! --------------*)

     WHILE (m1 < m2) & ~gefunden DO
        mitte := (m1+m2) DIV 2;
        IF IsBreakEvenPoint(mitte,preis) THEN
           gefunden := TRUE;
        ELSIF Erloes(mitte,preis) < Kosten(mitte) THEN
            m1 := mitte+1;
           ELSE
            m2 := mitte-1;
           END;
        END;
(* Bitte hier den ermittelten Mengenwert ausgeben: *)
        IF gefunden THEN
           Write.String("Break even point = ");
           Write.Int(mitte,0); Write.Ln;
        END
END BreakEvenPoint.
```

Aufgabe 5 (6 Punkte)

Gegeben ist folgendes Programm:

```
MODULE WhatIsThat;
  IMPORT Write;
  VAR
     a,b,c: INTEGER;
  PROCEDURE WhatIsThat(VAR x,y: INTEGER);
     VAR z: INTEGER;
  BEGIN
     z := x;
     x := y;
     y := z;
  END WhatIsThat;
BEGIN
   a := 6; b := 1; c := 9;
   Write.String("  a  |  b  |  c"); Write.Ln;
   Write.String("_____|_____|___"); Write.Ln;

   Write.Int(a,3); Write.String("  |");
   Write.Int(b,3); Write.String("  |");
   Write.Int(c,3); Write.Ln;

   IF a < b THEN
     WhatIsThat(b,a);
        Write.Int(a,3); Write.String("  |");
        Write.Int(b,3); Write.String("  |");
        Write.Int(c,3); Write.Ln;
   END;
   IF a < c THEN
     WhatIsThat(c,a);
        Write.Int(a,3); Write.String("  |");
        Write.Int(b,3); Write.String("  |");
        Write.Int(c,3); Write.Ln;
   END;
   IF b < c THEN
     WhatIsThat(c,b);
        Write.Int(a,3); Write.String("  |");
        Write.Int(b,3); Write.String("  |");
        Write.Int(c,3); Write.Ln;
   END;
END WhatIsThat.
```

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

a	b	c
6	**1**	**9**
9	**1**	**6**
9	**6**	**1**

Aufgabe 6 (14 Punkte)

Das im folgenden zu erstellende Oberon-Programm erhält als Eingabe (Standardeingabe oder aus Datei via Umlenkung) Datensätze mit folgendem Aufbau:

- Die Datensätze sind durch *newline* getrennt.
- Die Datensätze bestehen aus mehreren Feldern, die jeweils durch Doppelpunkt getrennt sind. Die Felder gelten als von 1 an durchlaufend nummeriert!
- Sie können davon ausgehen, dass der Aufbau der Eingabe korrekt ist (also z.Bsp. keine Leerzeilen).

Beispiel-Datensatz:

`Hubert:Meier:Wirtschaftswissenschaften:1. Semester:Berlin`

Formulieren Sie die Prozedur `Cut`, die aus einem als Parameter übergebenen Datensatz das Feld liefert, das durch einen weiteren Parameter (Feldnummer) bestimmt wird. Ist kein Feld mit dieser Nummer enthalten, so soll die Prozedur `FALSE` liefern, ansonsten `TRUE`.

Der Aufruf soll wie folgt möglich sein:

```
IF Cut(data,field,3) THEN
   Write.String(field);
END
```

<u>Lösung:</u>

```
PROCEDURE Cut(a: Data; VAR field: Field; nr: INTEGER):BOOLEAN;
   VAR i,j: INTEGER; found: BOOLEAN;
BEGIN
   IF nr <= 0 THEN RETURN FALSE END;
   i := 0; j := 1; found := FALSE;
   WHILE ~ found & ( i < SHORT(LEN(a)) ) & (a[i] # 0X) DO
   IF a[i] = ":" THEN
      INC(j);
      IF j = nr THEN found := TRUE END;
      END;
   INC(i);
   END;
   IF j < nr THEN RETURN FALSE END;
   j := 0;
   WHILE (i < SHORT(LEN(a))) & (j < SHORT(LEN(field))-1) &
      ( a[i] # 0X ) & (a[i] # ":") DO
     field[j] := a[i]; INC(i); INC(j);
   END;
   field[j] := 0X;
   RETURN TRUE;
END Cut;
```