Übungsklausur zur Allgemeinen Informatik I
WS 2000/01

Hinweise

- Zur Lösung der 10 Aufgaben stehen Ihnen 3 Stunden (180 Minuten) Bearbeitungszeit zur Verfügung. Insgesamt können Sie 140 Punkte erreichen.
- Hilfsmittel: Skript, Vorlesungsmitschrift und ein Modula-2-Buch. Elektronische Hilfsmittel sind nicht erlaubt!

1. Aufgabe 5

Beantworten Sie die folgenden Fragen jeweils mit Ja oder Nein (ohne weitere Erläuterungen!).

1. Gibt es kontextfreie Grammatiken, die sich nur mit Syntaxdiagrammen, nicht aber in EBNF definieren lassen? 1
2. Dürfen im lokalen Deklarationsteil einer Modula-2-Prozedur Variablennamen verwendet werden, die auch im Hauptprogramm vorkommen? 1
3. Ist Modula-2 eine funktionale Programmiersprache? 1
4. Wird für Variablen, die als `VAR`-Parameter an eine Prozedur übergeben werden, bei Aufruf der Prozedur eine Kopie angelegt? 1
5. Sind x und y zwei positive natürliche Zahlen, so gilt stets: 1

$$(x\ DIV\ y) * y = x - (x\ MOD\ y)$$

2. Aufgabe 8

Gegeben sei die Syntax einer Sprache L in der folgenden EBNF-Darstellung:

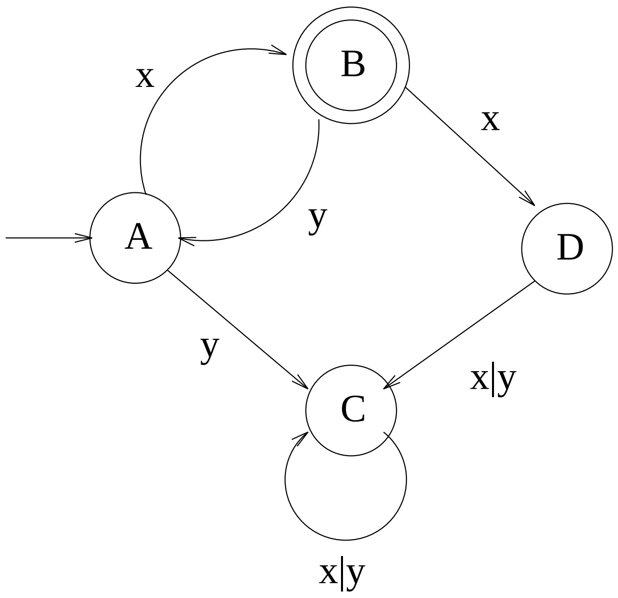
```
Wort ::= { A }  { B }  ( ( B Wort A ) | X )  { B }.
```

1. Übertragen Sie diese Darstellung in ein äquivalentes Syntaxdiagramm! 4
2. Geben Sie für die folgenden Zeichenketten jeweils an, ob sie ein syntaktisch korrektes Wort der Sprache darstellen (eine einfache Ja/Nein-Antwort ist ausreichend). 4

   1) `AXXXA`   2) `ABAXA`   3) `XXXB`   4) `AXXBBA`

3. Aufgabe 20

Gegeben sei nachfolgender durch seinen Zustandsgraphen beschriebener endlicher Automat $M = (Z, I, \delta, z_o, T)$. Mit L sei die von M akzeptierte Sprache bezeichnet.



1. Konstruieren Sie eine reguläre Grammatik $G = (S_N, S_T, P, \omega_s)$, die genau die Sprache L erzeugt, d.h. $L = L(G)$. 10

2. Geben Sie einen regulären Ausdruck α an, der genau die Sprache L erzeugt, d.h. $L = L(\alpha)$. 5

3. Geben Sie ein Syntaxdiagramm an, durch das genau die Sprache L erzeugt wird. 5

4. Aufgabe

Gegeben sei das folgende Modula-2-Programm. Welche Ausgabe erzeugt das Programm?

```
MODULE obscure;
   FROM InOut IMPORT WriteInt, WriteString, WriteLn;
   VAR  x, y, z1, z2 : INTEGER;

   PROCEDURE Obscure(x : INTEGER; VAR y : INTEGER) : INTEGER;
   BEGIN
      x := x + 1;
      y := y + 1;
      RETURN (x-y)
   END Obscure;
BEGIN
   WriteString("X / Y"); WriteLn;
   x := 4; y := 4;
   WriteInt(x,0); WriteString(" / "); WriteInt(y,0); WriteLn;

   z1 := Obscure(x,y);
   WriteInt(x,0); WriteString(" / "); WriteInt(y,0); WriteLn;

   z2 := Obscure(y,x);
   WriteInt(x,0); WriteString(" / "); WriteInt(y,0); WriteLn;

   IF z1 >= z2 THEN
      z1 := Obscure(x,y);
      WriteInt(x,0); WriteString(" / "); WriteInt(y,0); WriteLn;

      z2 := Obscure(y,x);
      WriteInt(x,0); WriteString(" / "); WriteInt(y,0); WriteLn
   ELSE
      y := Obscure(y,x);
      WriteInt(x,0); WriteString(" / "); WriteInt(y,0); WriteLn;

      x := Obscure(x,y);
      WriteInt(x,0); WriteString(" / "); WriteInt(y,0); WriteLn
   END
END obscure.
```

5. Aufgabe 30

In einem Modula-2-Programm soll eine Menge von Dreiecken verwaltet werden. Die Beschreibung eines **Dreiecks** umfaßt einen Eckpunkt, die Längen der beiden aus diesem Punkt hervorgehenden Seiten und den von diesen Seiten eingeschlossenen Winkel. Es sollen folgende drei Arten von Dreiecken unterschieden werden: GLEICHSEITIG (alle drei Seiten haben die gleiche Länge), GLEICHSCHENKELIG (genau zwei von drei Seiten haben gleiche Länge) und BELIEBIG (weder GLEICHSEITIG noch GLEICHSCHENKELIG).
Es sind folgende Datentypen vereinbart worden:

```
TYPE DREIECKTYP = (Gleichseitig, Gleichschenkelig, Beliebig);
     PUNKT      = RECORD
                     x, y : INTEGER
                  END;
```

1. Definieren Sie einen Datentyp `DREIECK` zur Speicherung der Dreiecke. Die Speicherung eines solchen Objektes soll den **Dreicktyp**, die angegebenen **Parameter** sowie den **Umfang** des Dreieckes umfassen. 6

2. Zur Verwaltung der Dreiecke wurde der folgende Datentyp definiert: 12

   ```
   TYPE OBJEKTE = ARRAY [0..99] OF DREIECK;
   ```

   Schreiben Sie nun eine Prozedur

   ```
   PROCEDURE Eintragen(pos            : CARDINAL;
                       VAR objects    : OBJEKTE);
   ```

   Diese soll bei Eingabe eines Dreieckes seinen Umfang berechnen und den Dreiecktyp ermitteln. Dreieckumfang und -typ sollen zusammen mit den beschreibenden Geometrieparametern als Gesamtstruktureintrag in dem Feld ablegt werden.

   Die Position `pos` an der das einzulesende geometrische Objekt im Feld eingetragen werden soll und das Feld `objects` selbst sind die Formalparameter der Prozedur.

   Umfangberechnung: Nehmen Sie an, daß eine Funktionsprozedur `DritteSeiteDreieck (Laenge1, Laenge2, Winkel)` zur Verfügung steht, die als Ergebnis die Länge der dritten Seite des durch die Parameter spezifizierten Dreiecks zurückliefert. Nehmen Sie vereinfachend an, daß zum Einlesen der Geometrieparameter die folgende externe Prozedur zur Verfügung steht (wobei die Formalparameter sämtlich als Referenzparameter definiert seien und somit Ergebnisse der Prozeduren liefern):

```
ReadTriangle(Eckpunkt, Laenge1, Laenge2, Winkel).
```

3. Schreiben Sie eine Prozedur 12

   ```
   PROCEDURE MittlererUmfang (belegt  : CARDINAL; objects : OBJEKTE);
   ```

   Diese soll für jeden Dreiecktyp den **mittleren Umfang** der jeweils in dem Feld gespeicherten Dreiecke berechnen und anschließend ausgeben.

   Als Formalparameter werden das Feld der geometrischen Objekte (`objects`) und die Anzahl der darin insgesamt gespeicherten Objekte (`belegt`) übergeben.

   Nehmen Sie an, das die Objekte im Feld konsekutiv, beginnend mit der Indexuntergrenze (= 0), abgespeichert wurden.

6. Aufgabe 18

Eine bereits eingelesene Zeichenkette (abgeschlossen durch `0C`) sei in einer Variablen vom Datentyp `STRING` gespeichert. Dieser Datentyp ist definiert durch:

```
TYPE STRING = ARRAY [0..255] OF CHAR;
```

Schreiben Sie die folgenden drei Modula-2-Prozeduren:

```
PROCEDURE LoescheNichtBuchstaben(VAR str : STRING);
```
Funktionalität: In der Eingabe-Zeichenkette werden Nicht-Buchstaben gestrichen. Das Ergebnis der dann nur aus Buchstaben bestehenden Zeichenkette wird zurückgegeben.

```
PROCEDURE GrossNachKlein(VAR str : STRING);
```
Funktionalität: Die vorhandenen Großbuchstaben werden in Kleinbuchstaben umgewandelt (für die Transformation, siehe beiliegende ASCII-Code Tabelle!).

```
PROCEDURE Reverse(VAR str : STRING);
```
Funktionalität: Die Zeichenkette wird in umgekehrter Reihenfolge abgespeichert und anschließend zurückgeben.

Beispiel: Wendet man diese drei Prozeduren nacheinander (in beliebiger Reihenfolge) auf die Zeichenkette `A9bCx` an, so erhält man die Zeichenkette `xcba` als Ergebnis.

7. Aufgabe 12

Gegeben sei das folgende Modula-2-Programm `Trickreich`.

```
MODULE Trickreich;
   FROM InOut IMPORT WriteInt, WriteLn;

   VAR  a, b, c : INTEGER;

   PROCEDURE P( (* Parameterliste wie unten *) );
   BEGIN
      c := x + y;
      y := x
   END P;

BEGIN
   a := 2;   b := 3;   c := 4;
   P(a+c, b);

   WriteInt(a,3); WriteLn;
   WriteInt(b,3); WriteLn;
   WriteInt(c,3); WriteLn
END Trickreich.
```

Für welche der folgenden Deklarationen der Prozedur `P` ist der Aufruf von `P` syntaktisch korrekt? Begründen Sie gegebenenfalls die Ablehnung.

```
a)    PROCEDURE P(    x: INTEGER;     y: INTEGER);
b)    PROCEDURE P(VAR x: INTEGER;     y: INTEGER);
c)    PROCEDURE P(    x: INTEGER; VAR y: INTEGER);
d)    PROCEDURE P(VAR x: INTEGER; VAR y: INTEGER);
```

Welche Ausgaben liefern die `WriteInt`-Anweisungen in den erlaubten Fällen?

8. Aufgabe 10

Gegeben sei das folgende Modula-2-Programm.

```
MODULE Q;
   FROM InOut IMPORT WriteInt, WriteLn;

   VAR  a, b, c, d, e : INTEGER;

   PROCEDURE P(VAR a, b : INTEGER; c : INTEGER) : INTEGER;
      VAR result : INTEGER;
   BEGIN
      b := a - b + d + c;
      IF c < b THEN
         result := c
      ELSE
         result := b
      END;
      RETURN result
   END P;

BEGIN
   a := 5;   b := 1;   c := 9;   d:= b;
   e := P(b,a,d) + c;

   WriteInt(a,3); WriteLn;
   WriteInt(b,3); WriteLn;
   WriteInt(c,3); WriteLn;
   WriteInt(d,3); WriteLn;
   WriteInt(e,3); WriteLn
END Q.
```

1. Welche Ausgaben erzeugt das Programm? 5

2. Kernstück des Programms ist die deklarierte Funktionsprozedur `P`, die als Ergebnis ihrer Ausführung einen Wert vom Typ `INTEGER` liefert. Verändern Sie das Programm derart, daß `P` eine normale Prozedur wird, die das Ergebnis über einen formalen Referenz-Parameter liefert. Passen Sie die Aufrufe und Berechnungen im Modul-Rumpf entsprechend an. 5

9. Aufgabe 15

Konstruieren Sie einen endlichen Automaten $M = (Z, I, \delta, z_0, T)$ über dem Eingabealphabet $I = \{0, 1\}$. Z ist die Menge der Zustände, z_0 der Anfangszustand, δ die Zustandsübergangsfunktion und T die Menge der Endzustände des Automaten.
Der Automat soll genau dann in einen Endzustand terminieren, wenn die zuletzt gelesene Zeichenfolge `1101` war.
Geben Sie den Automaten durch seinen Zustandsgraphen an.

10. Aufgabe 10

Zur Berechnung der Summe $H(n) = \sum_{i=1}^{n}(-1)^{i+1}/i$ wurde die folgende `REPEAT`-Schleife innnerhalb eines Modula-2-Programms implementiert.

```
ReadCard(n);   (* 0 <= n <= maxCard *)
summe   := 0.0;
IF n > 0 THEN
   i       := 0;
   REPEAT
      i  := i + 1;
      IF (i MOD 2 = 1) THEN
         vorzeichen :=  1.0
      ELSE
         vorzeichen := -1.0
      END;
      summand  := vorzeichen / FLOAT(i);
      summe    := summe + summand
   UNTIL  ((i MOD 2 = 1) AND (ABS(summand) < EPSILON)) OR (i >= n)
END;
WriteReal(summe,0); WriteLn;
```

Dabei sei `EPSILON` eine konstante `REAL`-Zahl.

1. Transformieren Sie die `REPEAT`-Schleife in eine äquivalente `WHILE`-Schleife. 5

2. Kann die `REPEAT`-Schleife in eine äquivalente `FOR`-Schleifenkonstruktion transformiert werden? Wenn ja, wie? 5