

# Prüfung aus Übersetzerbau 10.3.1995

## Musterlösung

1. 20 % Gegeben sei folgende LL(1)-Grammatik:

```

S      →  Stm R
R      →  ; Stm R | ε
Stm    →  if id then Stm Else endif | id
Else   →  else Stm | ε

```

Erstellen Sie die Analysetabelle mit Fehleraktionen (**skip**, **stop**) für die tabellengesteuerte Top-Down-Analyse.

```

First(S) = { if, id }      Follow(S) = { $ }
First(R) = { ";", ε }      Follow(R) = { $ }
First(Stm) = { if, id }    Follow(Stm) = { ";", $, else, endif }
First(Else) = { else, ε }  Follow(Else) = { endif }

```

	" ; "	if	id	then	endif	else	\$
S	skip	Stm R	Stm R	skip	skip	skip	stop
R	" ; "Stm R	ε(*)	ε(*)	ε(*)	ε(*)	ε(*)	ε
Stm	stop	if ...endif	id	skip	stop	stop	stop
Else	ε(*)	ε(*)	ε(*)	ε(*)	ε	else Stm	ε(*)

**2. 25 %** Gegeben sei ein Modula-2-Programm. Zeichnen Sie den Stack der Activation Records (mit lokalen Größen, Wert- und Variablenparametern, statischer und dynamischer Kette) zum Zeitpunkt (**\*here\***) ohne Verwendung eines Displays.

```

MODULE m;
TYPE IP=PROCEDURE(VAR INTEGER);
TYPE IPP=PROCEDURE(IP, VAR INTEGER);
VAR x,y,z:INTEGER;

  PROCEDURE a(b:IP; VAR y:INTEGER);
  BEGIN
    b(y);
  END a;

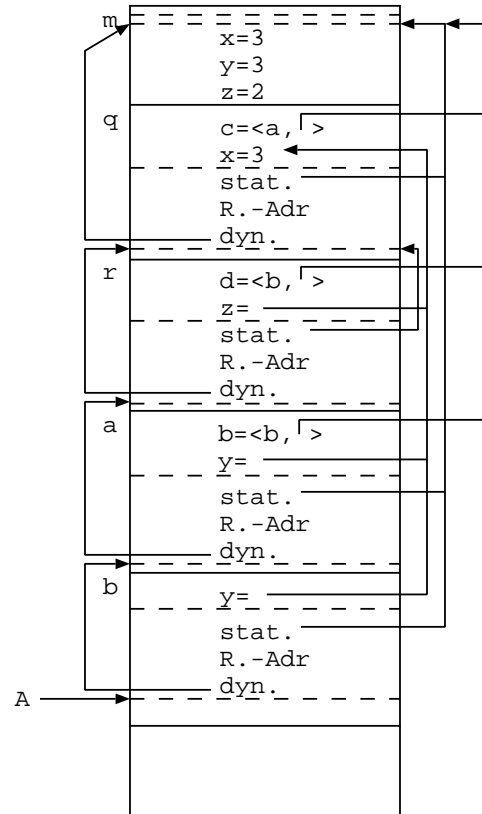
  PROCEDURE b(VAR y:INTEGER);
  BEGIN
    z:=x+1; x:=y; y:=z+1;    (*here*)
  END b;

  PROCEDURE q(c:IPP; x:INTEGER);

    PROCEDURE r(d:IP; VAR z:INTEGER);
    BEGIN
      c(d,z);
    END r;

    BEGIN
      r(b,x);
    END q;
  BEGIN
    x:=1;y:=3;z:=1;
    q(a,y);
  END m.

```



**3. 30 %** In einer einfachen Sprache zur Beschreibung von Bildschirmmasken stehen die Ausgabeelemente **vbox**, **hbox** und **feld** zur Verfügung: **vbox** hat als Parameter eine Liste von Ausgabeelementen, die vertikal untereinander dargestellt werden sollen, **hbox** hat als Parameter eine Liste von Ausgabeelementen, die horizontal nebeneinander dargestellt werden sollen; **feld** ist ein nicht weiter zerlegbares Ausgabefeld.

Beispiel:

```
hbox(
  vbox(
    hbox(feld1,feld2),
    feld3,
    hbox(feld4,feld5) ),
  vbox(feld6,feld7) )
```

erzeugt folgende Ausgabe:

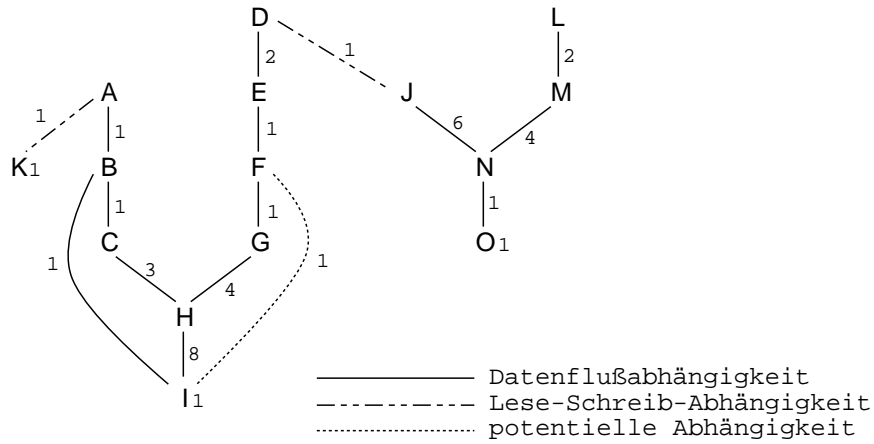
```
feld1 feld2   feld6
feld3         feld7
feld4 feld5
```

**a) (10 %)** Geben Sie eine Grammatik zur Beschreibung dieser Sprache an. Token sind die Symbole **vbox**, **hbox**, **feld**, "(", ")", " " und ",", ".".

**b) (20 %)** Erweitern Sie die Grammatik zu einer Attribuierten Grammatik, die für jedes Ausgabeelement die Anfangskoordinaten am Bildschirm errechnet und diese den Attributen **px** und **py** zuweist. Die Anfangsposition der gesamten Darstellung ist (0,0) (gibt die Koordinaten **x** und **y** an). Die Feldgröße von **feld** steht in den synthetisierten Attributen **sx** (horizontal) und **sy** (vertikal) zur Verfügung. Die vertikale Größe eines Elements **hbox** ergibt sich aus dem Maximum der vertikalen Größen seiner Parameter. Die horizontale Größe eines Elements **hbox** ergibt sich aus der Summe der horizontalen Größen seiner Parameter. Die vertikale Größe eines Elements **vbox** ergibt sich aus der Summe der vertikalen Größen seiner Parameter. Die horizontale Größe eines Elements **vbox** ergibt sich aus dem Maximum der horizontalen Größen seiner Parameter. Nehmen wir zum Beispiel an, daß jedes **feld** in obigem Beispiel eine vertikale Größe von 1 und eine horizontale von 3 besitzt, dann beginnt **feld<sub>1</sub>** an der Stelle (0,0), **feld<sub>2</sub>** auf (3,0), **feld<sub>3</sub>** auf (0,1), **feld<sub>4</sub>** auf (0,2), **feld<sub>5</sub>** auf (3,2), **feld<sub>6</sub>** auf (6,0) und **feld<sub>7</sub>** auf (6,1).

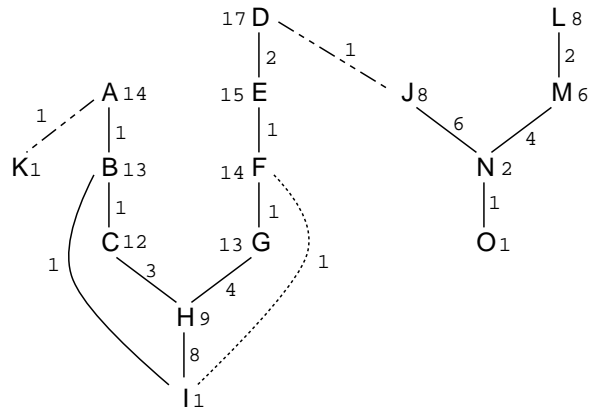
<p><b>a)</b></p> <p>S → L</p> <p>L → hbox ( L<sub>1</sub> )</p> <p>L → vbox ( L<sub>1</sub> )</p> <p>L → L<sub>1</sub> " , " L<sub>2</sub></p> <p>L → feld</p>	<p><b>b)</b></p> <p>L.px:=0; L.py:=0; L.hflag:=true</p> <p>L<sub>1</sub>.hflag:=true; L<sub>1</sub>.px=L.px; L<sub>1</sub>.py=L.py; L.sx=L<sub>1</sub>.sx; L.sy=L<sub>1</sub>.sy;</p> <p>L<sub>1</sub>.hflag:=false; L<sub>1</sub>.px=L.px; L<sub>1</sub>.py=L.py; L.sx=L<sub>1</sub>.sx; L.sy=L<sub>1</sub>.sy;</p> <p>L<sub>1</sub>.px=L.px; L<sub>1</sub>.py=L.py;</p> <p>IF L.hflag THEN {L.sy:=max(L<sub>1</sub>.sy,L<sub>2</sub>.sy); L.sx:=L<sub>1</sub>.sx+L<sub>2</sub>.sx;</p> <p style="padding-left: 100px;">L<sub>2</sub>.px=L.px+L<sub>1</sub>.sx; L<sub>2</sub>.py=L.py;}</p> <p>ELSE {L.sx:=max(L<sub>1</sub>.sx,L<sub>2</sub>.sx); L.sy=L<sub>1</sub>.sy+L<sub>2</sub>.sy;</p> <p style="padding-left: 100px;">L<sub>2</sub>.py=L.py+L<sub>1</sub>.sy; L<sub>2</sub>.px=L.px;}</p> <p>feld.px=L.px; feld.py=L.py; L.sx=feld.sx; L.sy=feld.sy;</p>
--	--

4. 25 % Gegeben sei der folgende Datenabhängigkeitsgraph für die Befehle A bis O. Im Datenabhängigkeitsgraphen sind die Kantenlängen bereits eingetragen.



Bestimmen sie die Pfadlängen im Datenabhängigkeitsgraphen und geben Sie die optimierte Befehlsanordnung für die Befehle A bis O an, die Sie mittels *list scheduling* erhalten. Keiner der Befehle sei ein Sprungbefehl.

Graph mit Pfadlängenangaben:



Optimierte Befehlsanordnung:

D A E F B G C L J H M K N O I ( $\rightarrow$  18 statt 30 Zyklen)