# Prüfung aus Übersetzerbau 18.6.1999 Musterlösung

## 1. 25 % Quadrupel-Code

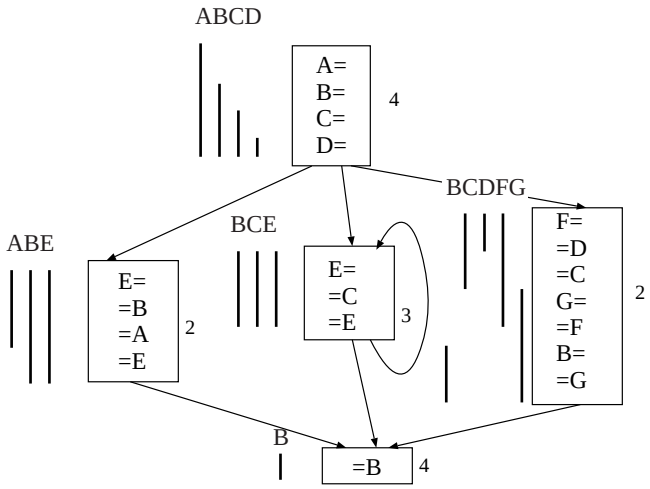
Erzeugen Sie für das rechte Programmstück Quadrupel-Code nach der Kontrollfluss-methode. Ein `INTEGER` ist 4 Byte und ein `LONG` 8 Byte groß. Die Untergrenze aller Indexbereiche ist 0.

```
VAR
  a: ARRAY[10,15] OF LONG;
  b: ARRAY[5] OF INTEGER;
  m,n: INTEGER;
  s: LONG
⋮
IF NOT( (m>n) OR (m>5) ) THEN
  s := a[b[m],n];
ELSE
  n := b[m-5];
END
```

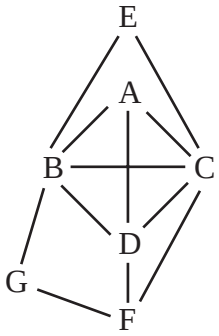
```
  if m>n goto IFfalse
  goto ORfalse
ORfalse:
  if m>5 goto IFfalse
  goto IFtrue
IFtrue:
  t1 = m * 4
  t2 = t1 + adr(b)
  t3 = @t2
  t4 = t3 * 15
  t5 = t4 + n
  t6 = t5 * 8
  t7 = t6 + adr(a)
  t8 = @t7
  s = t8
  goto IFend
IFfalse:
  t9 = m - 5
  t10 = t9 * 4
  t11 = t10 + adr(b)
  t12 = @t11
  n = t12
IFend:
```

**2. 30 %** Konfliktgraph

Gegeben sei der folgende Kontrollflussgraph. Rechts von den Blöcken sind die erwarteten Ausführungshäufigkeiten angegeben.




**a) (20 %)** Geben Sie den **Konfliktgraphen** und die **Auslagerungskosten** für alle Pseudoregister an—dabei soll angenommen werden, dass ein Speicherbefehl *zwei* Zyklen und ein Ladebefehl *drei* Zyklen kostet.




| | R= | =R | Kosten |
|---|---|---|---|
| A | 6 | 2 | $4 \cdot 2 + 2 \cdot 3 =$ |
| B | 4 | 6 | $6 \cdot 2 + 6 \cdot 3 = 30$ |
| C | 4 | 5 | $4 \cdot 2 + 5 \cdot 3 = 23$ |
| D | 4 | 2 | $4 \cdot 2 + 2 \cdot 3 = 14$ |
| E | 5 | 5 | $5 \cdot 2 + 5 \cdot 3 = 25$ |
| F | 2 | 2 | $2 \cdot 2 + 2 \cdot 3 = 10$ |
| G | 2 | 2 | $2 \cdot 2 + 2 \cdot 3 = 10$ |

**b) (10 %)** Bestimmen Sie eine Reihenfolge der **Registerbelegung**, belegen Sie die Pseudoregister mit realen Registern und kennzeichnen Sie eventuell auszulagernde Pseudoregister. Nehmen Sie an, dass *vier* reale Register zur Verfügung stehen.

| Reihenfolge | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| Register | 1 | 2 | 3 | 4 | 4 | 2 | 1 |

Diese Reihenfolge ist eine von mehreren Möglichkeiten.

**3. 20 %** Grammatik

Gegeben sei folgende Grammatik (Kleinbuchstaben und Sonderzeichen sind Terminalsymbole):

```
X → L R
R → + L R
R → ε
L → E
L → [ E , L ]
E → a
E → [ L ]
```

**a) (15 %)** Bestimmen Sie die First- und Follow-Mengen für alle Nonterminale der Grammatik.

| | First | Follow |
|---|---|---|
| X | [ a | $ |
| L | [ a | + ] $ |
| R | + ε | $ |
| E | [ a | + , ] $ |

**b) (5 %)** Ist die Grammatik LL(1)? Begründung!

Die Grammatik ist nicht LL(1) weil Bedingung 1 für LL(1)-Grammatiken verletzt ist. Für die Alternativen $\alpha_i$ der Produktionen für jedes N muss gelten

$$\text{First}(\alpha_i) \cap \text{First}(\alpha_j) = \{\} \text{f.a.i,j} (i \neq j)$$

Für die Alternativen der Produktion für L gilt das jedoch nicht:

```
L → E
L → [ E , L ]
```

| $a_i$ | First |
|---|---|
| E | [ a |
| [ E , L ] | [ |

First(E) ∩ First([ E , L ]) = {[} ≠ {}, d.h. die Grammatik ist nicht LL(1).

## 4. 25 % Attributierte Grammatik

Ein **B**ild besteht aus **O**bjekten die in **G**ruppen zusammengefaßt sind. Es gibt **k**ugeln, **q**uader und **p**yramiden mit Breite, Höhe und Farbe. Hinter jeder Gruppe können auch optionale **F**ilter stehen (**g**rößen- oder **f**arbänderung) die die Objekte in einer Gruppe verändern. In den Attributen `b`, `h` und `f` eines Objekts stehen Breite, Höhe und Farbe. In `g.w` bzw. `f.w` steht der Wert des Filters.

```
B → GL
GL → ( G ) OF | ( G ) OF GL
G → O | O G
O → k | q | p
OF → F | ε
F → g | f
```

Erweitern Sie die Grammatik um Attribute zur Ausgabe einer Objektliste bei der alle vorhandenen Filter auf die Objektgruppen angewendet wurden und die im Attribut `B.x` geliefert werden soll, d.h., die gruppierten Objekte sollen aufgeflacht werden. Jedes Objekt wird als Struktur in der Form (objekt, breite, höhe, farbe) ausgegeben. Bei einer Größenänderung werden Breite und Höhe des Objekts mit g.w multipliziert. Bei einer Farbänderung ergibt sich die neue Farbe des Objekts durch die Funktion `neu = farbe(alt, f.w)`. Der Operator `&` hängt Zeichenketten und Zahlen aneinander.

**Beispiel:** Aus der Eingabe `(q) (p k) g` mit den Attributen q.b=1, q.h=2, q.f=3, p.b=2, p.h=1, p.f=6, k.b=2, k.h=4, k.f=9 und g.w=2 wird `(q, 1, 2, 3) (p, 4, 2, 6) (k, 4, 8, 9)` erzeugt. Der Wert des Größenfilters wurde dabei auf Pyramide und Kugel angewendet.

```
B → GL              B.x = GL.x

GL → ( G ) OF       GL.x = G.x; G.f = OF.f; G.w = OF.w
GL → ( G ) OF GL    GL0.x = G.x & GL1.x; G.f = OF.f; G.w = OF.w

G → O               G.x = '(' & O.x & ')'; O.f = G.f; O.w = G.w
G → O G             G0.x = '(' & O.x & ')' & G1.x; O.f = G0.f; O.w = G0.w

O → k               if( O.f == 'g' ) { O.x = 'k' & k.b * O.w & k.h * O.w & k.f }
                    else if( O.f == 'f' ) { O.x = 'k' & k.b & k.h & farbe(k.f, O.w) }
                    else { O.x = 'k' & k.b & k.h & k.f }
O → q               analog zu k
O → p               analog zu k

OF → F              OF.f = F.f; OF.w = F.w
OF → ε              OF.f = ''; OF.w = 0

F → g               F.f = 'g'; F.w = g.w
F → f               F.f = 'f'; F.w = f.w
```