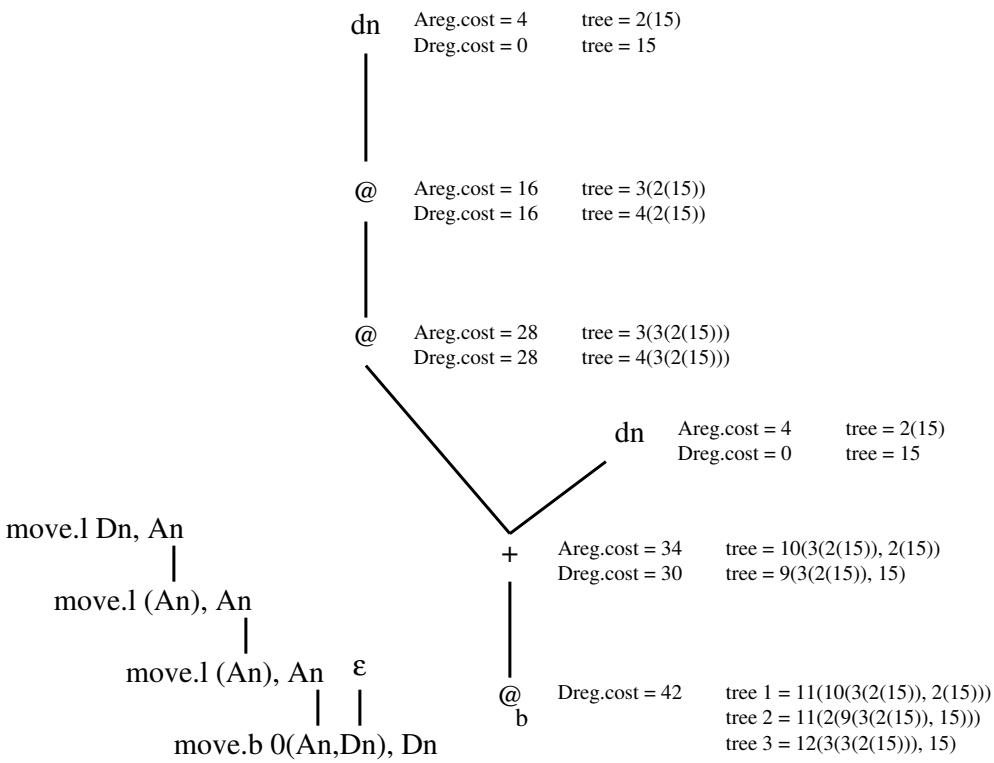
Prüfung aus Übersetzerbau 21.6.1996 Musterlösung

1. 25 % Codeerzeugung

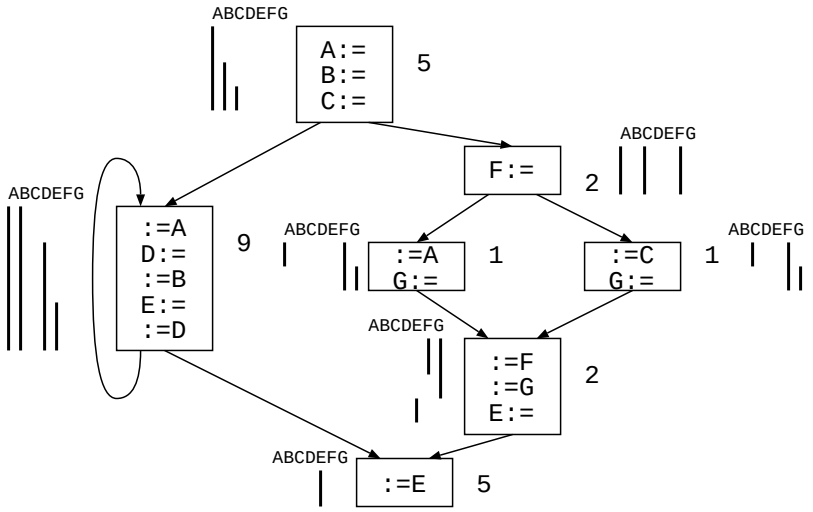
Erzeugen Sie für den folgenden Zwischencode-Baum optimalen Maschinencode für den 68000, wobei das Ergebnis in einem Datenregister landen soll. Verwenden Sie dazu die im Skriptum angegebene Befehlsauswahl-Baumgrammatik. (Optimaler Code hat in erster Linie die geringsten Kosten, in zweiter Linie die geringste Anzahl Befehle.)



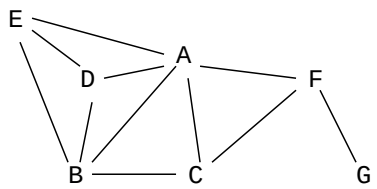
Die Bäume 1, 2 und 3 haben alle die gleichen Kosten, nämlich 42. Baum 3 beinhaltet jedoch nur vier Befehle im Gegensatz zu den anderen Bäumen mit je fünf Befehlen. Deshalb wird als optimale Codefolge Baum 3 ausgewählt.

2. 25 % Konfliktgraph und Auslagerungskosten

Gegeben sei der folgende Kontrollflußgraph. Neben den Blöcken sind die Aktivitätsbereiche der Pseudoregister und die erwarteten Ausführungshäufigkeiten angegeben.



a) (15 %) Geben Sie den Konfliktgraphen und die Auslagerungskosten für alle Pseudoregister an. Dabei soll angenommen werden, daß eine Zuweisung (`R:=`) einen Zyklus und eine Verwendung (`:=R`) zwei Zyklen kostet.



	R:=	:=R	Kosten
A	5	10	25
B	5	9	23
C	5	1	7
D	9	9	27
E	11	5	21
F	2	2	6
G	2	2	6

$Kosten = (\texttt{R} :=) + (:= \texttt{R}) * 2$

b) (10 %) Bestimmen Sie eine Reihenfolge der Registerbelegung. Belegen Sie die Pseudoregister mit realen Registern und kennzeichnen Sie die auszulagernden Pseudoregister. Nehmen Sie an, daß drei reale Register zur Verfügung stehen.

Registerbelegung
Reihenfolge von links nach rechts

A	B	D	E	C	F	G
1	2	3	*	3	2	1

Das Register `E` muß ausgelagert werden weil seine Priorität mit $\frac{21}{3}$ am niedrigsten ist (`A`: $\frac{25}{3}$, `B`: $\frac{23}{3}$, `D`: $\frac{27}{3}$).
Anmerkung: Die obige Reihenfolge der Registerbelegung ist nur eine von mehreren möglichen Lösungen.

3. 25 % Quadrupel-Code

```
VAR
  a: ARRAY[5,10,20] OF LONG;
  r: LONG;
  i,j,k: INTEGER;
⋮
WHILE (j<10) AND (NOT ((j>i) OR (j>k)) ) DO
  r := a[i,j,20-j];
  j := j + 1;
END
```

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

```
Wbegin:
    if j<10 goto ANDtrue
    goto Wfalse
ANDtrue:
    if j>i goto Wfalse
    goto ORfalse
ORfalse:
    if j>k goto Wfalse
    goto Wtrue
Wtrue:
    t1 := i * 10
    t2 := t1 + j
    t3 := t2 * 20
    t4 := 20 - j
    t5 := t3 + t4
    t6 := t5 * 8
    t7 := t6 + adr(a)
    t8 := @t7
    r := t8
    t9 := j + 1
    j := t9
    goto Wbegin
Wfalse:
```

4. 25 % Attributierte Grammatik

In einem Supermarkt gibt es eine Maschine zur Flaschenrückgabe. Erweitern Sie die nachfolgende Grammatik um Attribute, um den Flaschen- bzw. Kisteneinsatz zu berechnen.

S	→	RL
RL	→	R RL | R
R	→	K FL | F
K	→	`milch` | `bier`
FL	→	F FL | F | ε
F	→	`milch` | `bier`

	Einsatz	
	Flasche	Kiste
Milch	4 S	30 S
Bier	5 S	50 S

Ein Element `R` der Rückgabeliste `RL` besteht aus einer Flasche `F` oder einer Kiste `K` mit Flaschen `FL`. In einer Kiste müssen *alle* Flaschen vom gleichen Typ sein (z.B. passen in Milchkisten nur Milchflaschen) sonst wird für die *ganze* Kiste kein Einsatz berechnet.

Die Anzahl der Flaschen, Kisten und fehlerhaften Flaschen (z.B. eine Bierflasche in einer Milchkiste) sowie der errechnete Einsatz der angenommenen Flaschen und Kisten sollen in den synthetisierten Attributen S.f, S.k, S.ff und S.e geliefert werden.

S	→	RL	S.f = RL.f; S.k = RL.k; S.ff = RL.ff; S.e = RL.e
RL	→	R RL_1	RL.f = R.f + RL_1.f; RL.k = R.k + RL_1.k RL.ff = R.ff + RL_1.ff; RL.e = R.e + RL_1.e
RL	→	R	RL.f = R.f; RL.k = R.k; RL.ff = R.ff; RL.e = R.e
R	→	K FL	FL.kt = K.kt if(FL.fehler) R.f = 0; R.k = 0; R.ff = FL.n; R.e = 0 else R.f = FL.n; R.k = 1; R.ff = 0; R.e = K.e + FL.e
R	→	F	R.f = 1; R.k = 0; R.ff = 0; R.e = F.e
K	→	`milch`	K.kt = milch; K.e = 30
K	→	`bier`	K.kt = bier; K.e = 50
FL	→	F FL_1	FL_1.kt = FL.kt FL.n = 1 + FL_1.n if(FL.kt == F.ft) FL.e = F.e + FL_1.e; FL.fehler = FL_1.fehler else Fl.e = 0; FL.fehler = true
FL	→	F	FL.n = 1 if(FL.kt == F.ft) FL.e = F.e; FL.fehler = false else FL.e = 0; FL.fehler = true
FL	→	ε	FL.n = 0 FL.e = 0 FL.fehler = false
F	→	`milch`	F.ft = milch; F.e = 4
F	→	`bier`	F.ft = bier; F.e = 5