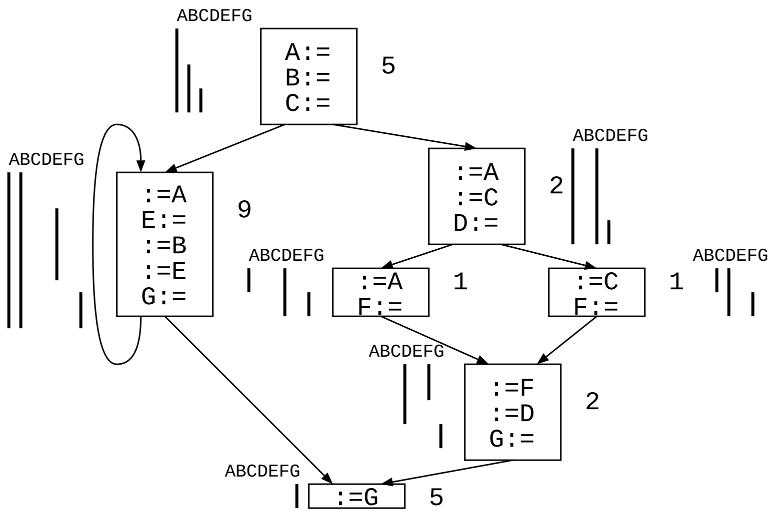
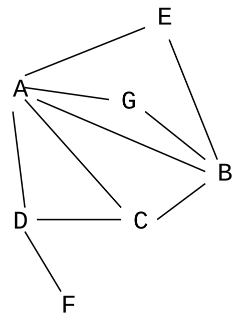
Prüfung aus Übersetzerbau 15.3.1996 Musterlösung

1. 25 % Konfliktgraph und Auslagerungskosten

Gegeben sei der folgende Kontrollflußgraph. Die Ausführungshäufigkeiten stehen rechts neben den Blöcken. Geben Sie den *Konfliktgraphen* und die *Auslagerungskosten* für alle Pseudoregister an. Dabei wird angenommen, daß ein Speicherbefehl *zwei* Zyklen und ein Ladebefehl *drei* Zyklen kostet.





R	R:=	:=R	Summe
A	5	12	46
B	5	9	37
C	5	3	19
D	2	2	10
E	9	9	45
F	2	2	10
G	11	5	37

2. 20 % Reguläre Definition

Ein URL (Uniform Resource Locator) gibt die Adresse eines Objekts im World Wide Web an. Er besteht aus der Zugriffsart (`http`, `ftp`, `gopher`, `mailto`, `news`) gefolgt von einem Doppelpunkt “`:`” und einer genaueren Beschreibung des Objekts, abhängig von der Zugriffsart. Bei `http`, `ftp` und `gopher` folgen zwei Schrägstriche “`//`”, ein *Hostname* und ein Zugriffspfad. Ein *Hostname* besteht entweder aus vier Zahlen mit maximal drei Ziffern, *getrennt* durch einen Punkt (z.B. 128.130.173.8) oder aus einer beliebigen Anzahl von Buchstabengruppen (Groß- und Kleinbuchstaben und “`-`”) *getrennt* durch einen Punkt (der *Hostname* muß in diesem Fall mit einem Buchstaben beginnen, z.B. ftp.uni-paderborn.de). Ein Zugriffspfad ist vom *Hostname* durch einen Schrägstrich getrennt. Er besteht aus beliebig vielen (nicht leeren) Filenamen, die durch Schrägstriche getrennt sind und darf auch mit einem Schrägstrich enden. Ein Filename besteht aus beliebig vielen Groß- und Kleinbuchstaben, “`-`” und “`.`”. Bei der Zugriffsart `news` folgt nach dem Doppelpunkt nur ein *Hostname*. Bei `mailto` folgt nach dem Doppelpunkt ein Name, ein “`@`” und ein *Hostname*. Ein Name besteht aus beliebig vielen Groß- und Kleinbuchstaben.

gültige URLs	ungültige URLs
`http://www.Info.ch/info.html`	`http://name@host.name`
`ftp://1.2.3.4/..A--B../`	`ftp:/123.4567.2.9873//file.txt`
`news:news.tuwien.ac.at`	`news://info.at/news`
`mailto:alex@complang.tuwien.ac.at`	`mailto:complang..tuwien.ac.at`

Erstellen Sie eine *reguläre Definition* für einen URL.

```
ziff  =  [0-9]
zahl  =  [0-9] ziff? ziff?
 bst  =  [a-z]|[A-Z]
host  =  ip|hn
  ip  =  zahl "." zahl "." zahl "." zahl
  hn  =  bst (bst|"-")* ("." (bst|"-")+)*
file  =  (bst|"-"|".")+
pfad  =  file ("/" file)* "/"?
 art  =  "http"|"ftp"|"gopher"
 URL  =  (art "://" host ("/" pfad)|("/"?) ) |
         ("news:" host) |
         ("mailto:" bst* "@" host)
```

3. 25 % Quadrupel-Code

```
VAR
  a: ARRAY[4,20,40] OF LONG;
  b: ARRAY[60] OF INTEGER;
  i,j,k,f: INTEGER;
⋮
IF NOT((a[i,j,k] > 5) OR (i > j)) THEN
  f := b[j + k];
ELSE
  f := k + i + j;
END
k := k + 1;
```

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.

```
           t1 := i * 20
           t2 := t1 + j
           t3 := t2 * 40
           t4 := t3 + k
           t5 := t4 * 8
           t6 := t5 + adr(a)
           t7 := @t6
           if t7 > 5 goto IFfalse
           goto ORfalse
 ORfalse:  if i>j goto IFfalse
           goto IFtrue
  IFtrue:  t8 := j + k
           t9 := t8 * 4
           t10 := t9 + adr(b)
           t11 := @t10
           f := t11
           goto end
 IFfalse:  t12 := k + i
           t13 := t12 + j
           f := t13
     end:  t14 := k + 1
           k := t14
```

4. 30 % Attributierte Grammatik

In HTML (Hypertext Markup Language) gibt es Konstrukte, um verschachtelte Listen zu definieren. Das Kommando <OL> (ordered list) leitet eine numerierte Liste ein, <UL> (unordered list) eine Liste mit *. Ein Listeneintrag beginnt mit <LI> (list item) und eine ganze Liste wird mit </OL> bzw. </UL> beendet. Auf dem Bildschirm wird der Text entsprechend seiner logischen Struktur eingerückt dargestellt.

<table>
<tr><th>HTML</th><th>Bildschirm</th></tr>
<tr><td>Das sind Listen:
<OL><LI> Text eins
<LI> Text zwei <UL>
<LI> erste Zeile
<LI> zweite Zeile
</UL>
<LI> Text drei
</OL>
Hier geht's weiter.</td><td>Das sind Listen:

    1. Text eins
    2. Text zwei

        * erste Zeile
        * zweite Zeile

    3. Text drei

Hier geht's weiter.</td></tr>
</table>

Erweitern Sie die rechts gezeigte Grammatik um Attribute für die Darstellung von Listen in einem HTML-Text. Unterscheiden Sie numerierte von nicht numerierten *Items* (Is). Der Text soll im synthetisierten Attribut S.x geliefert werden. Die Funktion tabs(n) liefert n Tabulatoren. Ein *newline* wird durch "\n" dargestellt. Das Attribut chars.x enthält die Textstücke zwischen den Kommandos. Verwenden Sie den Operator ||, um Zeichenketten zusammenzuhängen.

<table>
<tr><td>S</td><td>→</td><td>Html</td></tr>
<tr><td>Html</td><td>→</td><td>Text Html | Text</td></tr>
<tr><td>Text</td><td>→</td><td>chars | List</td></tr>
<tr><td>List</td><td>→</td><td><UL> Is </UL> |
<OL> Is </OL></td></tr>
<tr><td>Is</td><td>→</td><td>Item Is | Item</td></tr>
<tr><td>Item</td><td>→</td><td><LI> Html</td></tr>
</table>

<table>
<tr><td>S</td><td>→</td><td>Html</td><td>S.x := Html.x
Html.i := 0 /* noch keine Einrückung */</td></tr>
<tr><td>Html</td><td>→</td><td>Text $Html_1$</td><td>Html.x := Text.x || "\n" || $Html_1$.x
Text.i := Html.i; $Html_1$.i := Html.i</td></tr>
<tr><td>Html</td><td>→</td><td>Text</td><td>Html.x := Text.x; Text.i := Html.i</td></tr>
<tr><td>Text</td><td>→</td><td>chars</td><td>Text.x := chars.x</td></tr>
<tr><td>Text</td><td>→</td><td>List</td><td>Text.x := List.x; List.i := Text.i + 1 /* einrücken */</td></tr>
<tr><td>List</td><td>→</td><td><UL> Is </UL></td><td>Is.i := List.i; List.x := Is.x
Is.n := 1; Is.o := false /* keine Numerierung */</td></tr>
<tr><td>List</td><td>→</td><td><OL> Is </OL></td><td>Is.i := List.i; List.x := Is.x
Is.n := 1; Is.o := true /* mit Numerierung */</td></tr>
<tr><td>Is</td><td>→</td><td>Item Is_1</td><td>Item.i := Is.i; Is_1.i := Is.i
Is_1.n := Is.n + 1 /* nächste Nummer */
Item.n := Is.n; Item.o := Is.o; Is_1.o := Is.o
Is.x := Item.x || "\n" || Is_1.x</td></tr>
<tr><td>Is</td><td>→</td><td>Item</td><td>Item.i := Is.i; Item.n := Is.n
Item.o := Is.o; Is.x := Item.x</td></tr>
<tr><td>Item</td><td>→</td><td><LI> Html</td><td>Html.i := Item.i
IF Item.o THEN
    Item.x := tabs(Item.i) || Item.n || ". " || Html.x
ELSE Item.x := tabs(Item.i) || "* " || Html.x</td></tr>
</table>