Tentamen Vertalerbouw—3 februari 2006

De nagekeken tentamens zijn af te halen op het onderwijsbureau.

Opmerkingen:

- Schrijf **netjes** en duidelijk, met zwarte of blauwe pen.
- Zet op het eerste blad alle gegevens als naam, etc., en het totaal aantal ingeleverde bladen, en nummer de ingeleverde bladen.
- Lees de opgaven eerst goed door.
- Motiveer uw antwoorden.
- Wilt u 7 EC (oude stijl) halen zet dat dan boven het tentamen! U dient dan ook som 2 over attributen-grammatica's te maken.

1. (45 minuten)

   a) Geef voor alle nonterminals uit onderstaande produkties de sets *first* en *follow*.

   b) Is de grammatica, gegeven door de volgende produkties met startsymbool S, $LL(1), LR(0)$, $SLR(1)$, $LR(1)$?
   Geef in geval van conflicten deze duidelijk aan. Geef, ingeval de conflicten volgens U oplosbaar zijn, aan hoe de oplossing verloopt.

   $$\begin{array}{cccl} & S & \rightarrow & AbC \\ , & A & \rightarrow & Ab \\ , & A & \rightarrow & c \\ , & A & \rightarrow & S \\ , & C & \rightarrow & bC \\ , & C & \rightarrow & c \end{array}$$

2. (40 minuten, **oude stijl!!**)

   Gegeven is een eenvoudig taaltje, dat syntactisch gespecificeerd wordt door:

   $$\begin{array}{cccl} & Decls & \rightarrow & Decl\ ;\ Decls \\ , & Decls & \rightarrow & \\ , & Decl & \rightarrow & ident\ :\ Type \\ , & Type & \rightarrow & int \\ , & Type & \rightarrow & real \\ , & Type & \rightarrow & record\ Fields\ end \\ , & Fields & \rightarrow & Field\ ;\ Fields \\ , & Fields & \rightarrow & \\ , & Field & \rightarrow & ident\ :\ Type \end{array}$$

   Een voorbeeldzin uit dit taaltje is:

   $$i:\ int; r:\ real; s:\ real; a:\ record\ n:\ int; f:\ real; end;$$

Gegeven is verder dat een `int` 4 bytes memory kost, en een `real` 8 bytes. Het is de bedoeling de syntaxregels te voorzien van attributen en reken-voorschriften, zodanig dat van elke declaratie wordt berekend hoeveel memory bytes nodig zijn. Daarnaast dient de totale hoeveelheid ruimte in bytes (voor alle declaraties tesamen) afgedrukt te worden.

Voor onze voorbeeldzin moet de uitvoer dus zijn: 4 8 8 12 32.

Je mag daarbij gebruik maken van de volgende globale declaraties:

```
sizes : array [1..maxint) of integer;
n : integer = 0; (* sizes[1..n] is ingevuld *)
```

3. (45 minuten)

Gegeven is het volgende Pascal(-achtige) programma:

```
PROGRAM tent (input,output);

TYPE paar = array [1..2] of integer;

VAR p: paar;
    i: integer;

PROCEDURE copy (VAR a: paar; b: paar);
VAR i: integer;

   PROCEDURE one (i: integer);
   BEGIN IF (i<3) THEN
            a[i+1] := b[i];              (* 1 *)
         ELSE a[1] := b[i];
   END;

BEGIN FOR i := 1 TO 2 DO BEGIN
          one(i);                        (* 2 *)
          a[i] := a[i] + p[i];           (* 3 *)
      END
      ...
END (* set_all *)

BEGIN ...
      copy(p,p)                          (* 4 *)
      ...
END (* tent *).
```

Voor het geheugenbeheer worden de volgende registers gebruikt:

gp het base address van het activation record van het hoofdprogramma

lnb het base address van het huidige activation record

lfa het adres van de eerste vrije stack locatie

(We gaan ervan uit dat het return adres op een aparte stack wordt bewaard, zodat U daarmee geen rekening hoeft te houden.)

Voor het overdragen van de omgeving van een aan te roepen procedure kan het register **env** worden gebruikt. Verder zijn er voldoende registers (R0, R1, . .) voor het opslaan van tussenresultaten. Merk op dat het (Pascal) keyword `VAR` betekent dat het argument *by reference* wordt doorgegeven.

- Teken de AR (activation records) voor procedure `copy`;
- Geef de te genereren (pseudo-)instructies voor de **entry** en **exit** van `one`;
- Geef de te genereren (pseudo-)instructies voor de vier genummerde regels. U hoeft geen arraybound checks te doen!

4. (50 minuten)

Gegeven is een eenvoudig taaltje, dat syntactisch gespecificeerd wordt door:

$$
\begin{array}{lcl}
Test & \rightarrow & E == E \\
E & \rightarrow & T + E \\
E & \rightarrow & T \\
T & \rightarrow & F * T \\
T & \rightarrow & F \\
F & \rightarrow & (E) \\
F & \rightarrow & intdenotation \\
F & \rightarrow & realdenotation \\
F & \rightarrow & Var \\
Var & \rightarrow & identifier \\
Var & \rightarrow & identifier [E]
\end{array}
$$

a). Verander deze grammatica zodanig dat deze voldoet aan de $LL(1)$ eis (laat wel de beschreven taal gelijk!).
b). Schrijf een topdown recursive descent parser, zonder error recovery.

Onderstaande declaraties mogen worden gebruikt, danwel aangepast, in de implementatie van de parser. Alle overige zaken dient U volledig te declareren.

```
TYPE
  symbol     = (Test, E, T, F, Var, identifier, ... endoffile);
  tsymbol    = identifier .. endoffile;
  tsymbolset = SET OF tsymbol;
VAR
  sym: tsymbol;

PROCEDURE initscanner;
  (* Initialisatie van de scanner *)

PROCEDURE nextsym;
  (* Levert bij aanroep de tokenwaarde op (in de variabele
     sym) van het eerstvolgende symbool in de invoer *)

FUNCIION first (s: symbol): tsymbolset;
  (* levert de first symbols voor (non-)terminal s *)

FUNCTION follow (s: symbol): tsymbolset;
  (* levert de follow symbols voor (non-)terminal s *)
```