Tentamen Vertalerbouw 1—26 juni 2001

- 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.
- De opgaven zullen gewogen meetellen in het totaalcijfer, volgens de vermelde bestedingstijd.

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, $LR(0)$, $SLR(1)$, $LR(1)$? Geef in geval van conflicten deze duidelijk aan.

   $$\begin{array}{lcl} S & \rightarrow & AbSB \mid AB \\ , \; A & \rightarrow & Aa \mid b \\ , \; B & \rightarrow & bB \mid \end{array}$$

2. (45 minuten)

   Gegeven is de volgende syntax voor arithmetische expressies:

   $$\begin{array}{lcl} E & \rightarrow & T\ X \\ , \; T & \rightarrow & F\ Y \\ , \; F & \rightarrow & intdenot \mid lpar\ E\ rpar \\ , \; X & \rightarrow & plus\ T\ X \mid \\ , \; Y & \rightarrow & times\ F\ Y \mid \end{array}$$

   Het is de bedoeling deze syntaxregels te voorzien van attributen en rekenvoorschriften, zodanig dat de expressie-waarde wordt berekend.
   Gegeven is, dat de terminal `intdenot` een synthesized attribuut `value` heeft van type integer.

   Geef bij elk grammatica symbool aan welke attributen erbij horen, en van ieder attribuut of het inherited danwel synthesized is.

3. (50 minuten)

Gegeven is het volgende 'programma' in (pseudo-) Pascal:

```
PROGRAM tentamen;
  VAR a: ARRAY [5..15] OF integer;
      m,n: integer;
      ...
  PROCEDURE p (VAR m: integer);
    VAR k: integer;
    BEGIN k := a[m];                    (* 1 *)
          m := m+k-n                    (* 2 *)
    END;

  FUNCTION q (n: integer): integer;
    BEGIN p(m);                         (* 3 *)
          q := a[n]+m                   (* 4 *)
    END;
  ...
BEGIN
  ...
  m := 5;
  n := 2*q(1);                          (* 5 *)
  ...
END (* tentamen *).
```

Voor het geheugenbeheer en de adresberekeningen 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, en

`LFA` het adres van de eerste vrije geheugenlokatie.

Voor het overdragen van de omgeving van een aan te roepen procedure kan het register `ENV` worden gebruikt.

In de machineinstructies `CALL` en `RETURN` van de doelmachine wordt impliciet gebruik gemaakt van een (aparte) return stack. U hoeft zich dus niet druk te maken over terugkeer-adressen!

Er zijn voldoende registers (R0, R1, R2, ...) voor het opslaan van de tussenresultaten.

a) Geef de layout van de activation records van p, q en *tentamen*.

b) Geef de te genereren (pseudo-)instructies voor de procedure-entry en exit van p.

c) Geef de te genereren (pseudo-)instructies voor de 5 gemarkeerde statements. Controle op array grenzen is niet nodig!

4. (40 minuten)

Zij $G = (T,\ N,\ Z,\ P)$ een uitgebreide contextvrije grammatica met

$T = \{assign, beg, else, end, fi, ident, if, lpar, number, op, rpar, sc, then, eof\}$

$$N = \{S, Si, Sl, St, Ep, E, T, X\}$$

$$\begin{array}{lrcl} P = \{ & S & \rightarrow & Si \mid if\ E\ then\ Si\ Ep\ fi \quad \} \\ , & E & \rightarrow & T\ X \\ , & T & \rightarrow & number \mid ident \mid lpar\ E\ rpar \\ , & X & \rightarrow & op\ T\ X \mid \\ , & Ep & \rightarrow & else\ S \mid \\ , & Si & \rightarrow & assign \mid beg\ Sl\ end \\ , & Sl & \rightarrow & S\ St \\ , & St & \rightarrow & sc\ S\ St \mid \end{array}$$

(De terminal sc staat voor de semicolon.) De grammatica G is $LL(1)$.

a) Geef de default productieregels aan.
b) Gevraagd is een recursive descent parser zonder error-recovery voor deze grammatica.

U mag gebruik maken van de volgende declaraties:

```
TYPE
  tsymbol    = (assign, beg, else, end, fi, ident, if,
                lpar, number, op, rpar, sc, then, eof);
  tsymbolset = SET OF tsymbol;

VAR
  sym: tsymbol;

PROCEDURE initscanner;
  (* Initialisatie van de scanner *)
  BEGIN  ...  END (* initscanner *);

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

PROCEDURE error (sy: tsymbol;  str: string);
  (* Genereert een foutmelding in de vorm:
     representatie van sy  waarde van str   *)
  BEGIN  ...  END (* error *);
```