Hertentamen Vertalerbouw—21 januari 2004

De nagekeken tentamens zijn af te halen bij 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.
- De opgaven zullen gewogen meetellen in het totaalcijfer, volgens de vermelde bestedingstijd.

1. (40 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 E, $LL(1), LR(0)$, $SLR(1)$, $LR(1)$?
   Geef in geval van conflicten deze duidelijk aan.

   $$\begin{array}{lll} & E & \rightarrow & FbG \\ , & F & \rightarrow & aE \\ , & F & \rightarrow & aG \\ , & G & \rightarrow & \\ , & G & \rightarrow & cE \end{array}$$

2. (50 minuten)

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

   $$\begin{array}{lll} & B: & \textbf{LET}\ Ds\ \textbf{IN}\ E\ ? \\ , & Ds: & D\ ;\ Ds \\ , & Ds: & D \\ , & D: & \textbf{FUNC}\ \text{id}\ Ft \\ , & Ft: & (\ Pl\ )\ :\ T \\ , & Pl: & T\ Ps \\ , & Ps: & \\ , & Ps: & ,\ Pl \\ , & T: & \textbf{int} \\ , & T: & \textbf{real} \\ , & E: & \text{id}\ (\ Al\ ) \\ , & E: & \text{intdenot} \\ , & E: & \text{realdenot} \\ , & Al: & E\ Es \\ , & Es: & \\ , & Es: & ,\ Al \end{array}$$

Ziehier een (korrekt) voorbeeld uit dit taaltje:

```
LET
    FUNC p (int,real) : real;
    FUNC q (real) : int
IN
    p(17,q(1.2))?
```

Het is de bedoeling deze syntaxregels te voorzien van attributen. De volgende restricties dienen opgelegd te worden:

- gebruik van niet-gedeclareerde identifiers wordt gesignaleerd;
- gebruik van dubbel-gedeclareerde identifiers wordt gesignaleerd;
- ongelijk aantal argumenten en parameters wordt gesignaleerd;
- ongelijk type van argument en parameter wordt gesignaleerd;

Geef bij elk grammatica symbool aan welke attributen erbij horen, en van ieder attribuut of het inherited danwel synthesized is. Declareer ook de door U gebruikte datastructuren en procedures.

U mag aannemen dat er een procedure `nextsym` is, die de terminal symbols (zoals de komma, identifier, intdenot...) herkent.

3. (45 minuten)

Gegeven is het volgende pseudo-Pascal programma:

```
PROGRAM tentamen;

VAR a,b : integer; r: real;

FUNCTION f (a: integer; VAR c: integer; d: integer): real;
  VAR x: real;
  BEGIN x := (a+b)/d;                          (* 1 *)
        c := a + d;                            (* 2 *)
        f := 2 * x;                            (* 3 *)
  END;

PROCEDURE p (q: real);
  VAR i: integer;
      a: ARRAY [1..10] OF real;
  BEGIN FOR i:=1 TO 10 DO BEGIN
            a[i] := f(i,b,10) * q;             (* 4 *)
        END;
        r := a[10]                             (* 5 *)
  END;

BEGIN p(2.0); writeln(r)
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) Teken het activation record van de function f.

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 arrayindex–waarden is niet nodig!

4. (45 minuten)

Zij G een uitgebreide contextvrije grammatica met de set P van produktieregels:

$$
\begin{array}{rlcll}
P = \{ & Z & \rightarrow & S\ eof & \} \\
, & S & \rightarrow & Si & \\
, & S & \rightarrow & & \\
, & E & \rightarrow & T\ Xt & \\
, & T & \rightarrow & number & \\
, & T & \rightarrow & ident & \\
, & T & \rightarrow & lpar\ E\ rpar & \\
, & Xt & \rightarrow & op\ T\ Xt & \\
, & Xt & \rightarrow & & \\
, & Ep & \rightarrow & else\ S & \\
, & Ep & \rightarrow & & \\
, & Si & \rightarrow & assignment & \\
, & Si & \rightarrow & begin\ Sl\ end & \\
, & Sl & \rightarrow & S\ St & \\
, & St & \rightarrow & semicol\ S\ St & \\
, & St & \rightarrow & &
\end{array}
$$

- Geef voor elke nonterminal in G een default productie aan. (In de grammatica gegeven door de default producties moet elke nonterminal termineren.)
- Geef de functiewaarden van de functie *follow* voor de nonterminals Xt en St.
- Toon aan dat de richters van de producties voor de nonterminal Xt disjunct zijn. Evenzo voor de nonterminal St.
- Geef de implementatie van het hoofdprogramma en van de procedures voor de nonterminals Si, Sl en St in een recursive descent parser voor G, *inclusief* syntactische error recovery.

Onderstaande declaraties mogen worden gebruikt in de implementatie van de recursive descent parser. Alle overige gebruikte procedures e.d. moeten gedeclareerd worden.

```
TYPE
  tsymbol    = (assignment, begin, else, end, fi, ident, if,
                lpar, number, op, rpar, semicol, 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 *);
```