Tentamen Vertalerbouw 1—20 november 1998

De gecorrigeerde tentamens zijn af te halen op 21-12-98 bij het Onderwijsbureau, in de Vertalerbouw-map.

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)

   Gegeven is een grammatica met de volgende produktieregels:

   $$
   \begin{array}{lll}
   \{ & P & \rightarrow \ \mathit{Decls\ Eqs} \\
   , & \mathit{Decls} & \rightarrow \ D \mid D\ \mathit{Decls} \\
   , & D & \rightarrow \ \mathit{Tp\ Lst\ semicolon} \\
   , & \mathit{Tp} & \rightarrow \ \mathit{int} \mid \mathit{real} \\
   , & \mathit{Lst} & \rightarrow \ \mathit{id} \mid \mathit{id\ comma\ Lst} \\
   , & \mathit{Eqs} & \rightarrow \ \mathit{Eq} \mid \mathit{Eq\ Eqs} \\
   , & \mathit{Eq} & \rightarrow \ \mathit{E\ equal\ E\ semicolon} \\
   , & E & \rightarrow \ T \mid T\ \mathit{plus}\ E \\
   , & T & \rightarrow \ \mathit{id} \\
   \} &&
   \end{array}
   $$

   Is deze grammatica $LL(1), LR(0), SLR(1), LALR(1), LR(1)$? Geef in geval van conflicten duidelijk aan waarom dit conflicten zijn.

2. (45 minuten)

   Zie opnieuw de grammatica-regels uit som 1. Het is de bedoeling deze regels te voorzien van attributen en rekenvoorschriften, zodanig dat er een typecontrole plaatsvindt: expressies (E) links en rechts van een *equal* symbol hebben hetzelfde type. Er zijn geen (impliciete) coercies van *int* naar *real* (of omgekeerd) toegestaan. Geef duidelijk aan of de door U geïntroduceerde attributen synthesized danwel inherited zijn!

   De terminal symbol *id* heeft een (door de scanner ingevuld) synthesized attribuut $\mathit{SymId} : \mathit{integer}$, waarmee de stringrepresentatie van de identifier kan worden teruggevonden in de symboltable. Tevens is er in de symboltable entries ruimte voor het type van een identifier. U mag hiervoor gebruik maken van onderstaande declaraties:

```
TYPE tipe = (inttp,realtp);

PROCEDURE storeid (Sid: integer; t:tipe);
(* slaat de identifier Sid op met type t *)

FUNCTION gettype (Sid: integer): tipe;
(* geeft het type van de identifier Sid terug *)
```

3. (50 minuten)

Gegeven is het volgende pseudo-Pascal programma:

```
PROGRAM tentamen;

TYPE int = integer;

VAR a: int;

PROCEDURE p (FUNCTION f (i: int): int);
  VAR i: int;
  BEGIN FOR i:=1 TO 10 DO BEGIN
            a := a + f(i)                    (* 1 *)
        END
  END;

PROCEDURE q (VAR x: int);
  VAR a: int;
  PROCEDURE r (b: int);
    FUNCTION f (i: int): int;
      BEGIN f := i * i                       (* 2 *)
      END;
    BEGIN (* r *)
          x := b + f(a);                     (* 3 *)
          p(f)                               (* 4 *)
    END;
  BEGIN (* q *)
        r(x)                                 (* 5 *)
  END;

BEGIN (* main *)
      a := 0;
      q(a)                                   (* 6 *)
END;
```

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.

Een function-call wordt alsvolgt geïmplementeerd: eerst wordt ruimte voor het functieresultaat op de stack vrijgemaakt, vervolgens worden de argumenten op de stack geplaatst, etcetera.

a) Teken de activation records van de procedure p en q.

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

c) Geef de te genereren (pseudo-)instructies voor de 6 gemarkeerde statements.

4. (45 minuten)

Indien nodig, herschrijf de produktieregels uit som 1, zodanig dat deze regels met de LL(1) methode te parsen zijn. Schrijf vervolgens een topdown parser, die gebruik maakt van een expliciete stapel, die de taal accepteert als beschreven door de (aangepaste) grammatica. Deze parser dient fouten slechts te signaleren, recovery is dus niet nodig.

U mag hierbij gebruik maken van de volgende declaraties:

```
TYPE
  symbol    = (P, Decls, Eqs, D, Tp, Lst, Eq, E, T,
               semicolon, id, comma, int, real, equal,
               plus, eofs);
  tsymbol   = semicolon..eofs;

VAR
  sym: tsymbol;
  SymId: integer;

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,
     en in geval van een  id  in SymId een integer    *)

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