Tentamen Vertalerbouw—5 december 2002

De gecorrigeerde tentamens zijn af te halen op de begane grond in de tentamenkast.

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 First en Follow voor alle nonterminals in de navolgende grammatica:
   $G = (\{b, c\}, \{A, B, C\}, P, A)$, met

$$
\begin{array}{lllll}
P = & \{ & A & \rightarrow & B\ C \\
 & , & B & \rightarrow & C\ b \\
 & , & B & \rightarrow & b \\
 & , & C & \rightarrow & c\ A \\
 & , & C & \rightarrow & \\
 & \} & & &
\end{array}
$$

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

2. (45 minuten)

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

$$
\begin{array}{lll}
Assign & \rightarrow & Var\ becomessym\ Expr \\
Var & \rightarrow & identifier \\
Var & \rightarrow & identifier\ lbracksym\ Expr\ rbracksym \\
Expr & \rightarrow & Term\ plussym\ Expr \\
Expr & \rightarrow & Term \\
Term & \rightarrow & Factor\ mulsym\ Term \\
Term & \rightarrow & Factor \\
Factor & \rightarrow & Var \\
Factor & \rightarrow & intdenotation \\
Factor & \rightarrow & realdenotation \\
Factor & \rightarrow & lpar\ Expr\ rpar
\end{array}
$$

Het is de bedoeling de syntaxregels te voorzien van attributen en rekenvoorschriften, zodanig dat van elke assignment de typecorrectheid gechecked wordt. Hierbij moet ofwel typegelijkheid gelden, ofwel een type integer (intern) geconverteerd worden naar type real (impliciete coercie). Bij een array dient de indexexpressie altijd het type integer te hebben.

Voorbeeld 1: $r := 5 + (j + 2) * 3$ is typecorrect wanneer r van type int is en j ook, of r real is en j van type int of real.

Voorbeeld 2: $a[i] := a[j] * c$ is typecorrect wanneer zowel i als j type int heeft, en als array a type real heeft mag c zowel int als real zijn, wanneer array a type int heeft moet c van type int zijn.

Je mag hierbij gebruik maken van de elders gedefinieerde en geïmplementeerde begrippen

```
type typeTp = (intTp, realTp, unknownTp);
var Sym: tsymbol;   { current scanned symbol }
    SymId: integer; { index of current symbol in symboltable }
procedure getType (id: SymId; var t: typeTp);
{ gegeven de index in de symboltable retourneer het
  bijbehorend type; als de identifier onbekend is retourneer
  dan unknownTp.
}
```

3. (50 minuten)

Gegeven is het volgende (Pascal-) programma:

```
PROGRAM tentamen;

CONST upb = 6;

VAR a, b, c: integer;

PROCEDURE p1 (y: integer);
  VAR b   : integer;
      a   : ARRAY [0..upb] OF integer;

  PROCEDURE p2 (VAR x: integer);
    VAR b: integer;
    BEGIN ...
          x := a[b] + y;                    (* 1 *)
          ...
    END (* p2 *);
```

```
  PROCEDURE q2 (VAR x: integer; y: integer);
    VAR c, e: integer;
    BEGIN ...
          x := c*y + a[e];                          (* 2 *)
    END (* q2 *);

  BEGIN ...
        p2 (b);                                     (* 3 *)
        ...
        q2 (a[c]);                                  (* 4 *)
        ...
  END (* p1 *);

BEGIN ...  p1(a);  ...
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 lable` 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 $p1$, $p2$ en $q2$.

b) Geef de te genereren (pseudo-)instructies voor de procedure-entry en exit van $p1$ en $q2$.

c) Geef de te genereren (pseudo-)instructies voor de 4 gemarkeerde statements. Controle op index–waarden, die buiten array grenzen gaan, is niet nodig!

4. (45 minuten)

Gegeven is de volgende grammatica:

```
Block: letsym, Def, insym, Comp, nisym
     .
Def  : ident, Parlst, equalsym, Expr, semicolonsym
     .
Comp : Expr, questionsym
     .
Parlst: lparsym, Pars, rparsym
     ; Empty
     .
Pars : ident, RstPrs
     .
RstPrs: commasym, ident, RstPrs
     ; Empty
     .
Expr : Factor, RstXpr
     .
RstXpr: plussym, Factor, RstXpr
     ; Empty
     .
Factor: lparsym, Expr, rparsym
     ; intdenotation
     ; ident, Arglst
     .
Arglst: lparsym, Args, rparsym
     ; Empty
     .
Args : Expr, RstArgs
     .
RstArgs: commasym, Expr, RstArgs
     ; Empty
     .
Empty:
     .
```

a). Toon aan dat deze grammatica $LL(1)$ is.
b). Schrijf een topdown parser die een expliciete stack manipuleert, zodanig dat deze parser invoer volgens bovenstaande grammatica accepteert. De parser moet foutieve invoer (dwz. een fout in de invoer) signaleren.

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

```
TYPE
  symbol     = (Block, ... Empty,
                ident, ... endoffile);
  tsymbol    = ident..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 *)

FUNCTION first (s: symbol): tsymbolset;
  (* retourneert de first set van symbool s *)

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

{ er is een stack gedeclareerd met de volgende operaties:}
PROCEDURE push (sym: symbol);
PROCEDURE pop;
FUNCTION top: symbol;
PROCEDURE initstack;
FUNCTION isempty: boolean;
```