Hertentamen Vertalerbouw—24 augustus 2005

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.
- 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 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.

   $$\begin{array}{cccl} & S & \rightarrow & ABC \\ , & A & \rightarrow & aB \\ , & A & \rightarrow & b \\ , & B & \rightarrow & Bb \\ , & B & \rightarrow & S \\ , & C & \rightarrow & cA \end{array}$$

2. (50 minuten)

   Gegeven zijn de volgende produkties:

   $$\begin{array}{lcl} Prog & \rightarrow & Pdecls\ Pcall \\ Pdecls & \rightarrow & Pdecl\ Pdecls \mid Empty \\ Pdecl & \rightarrow & procsym\ id\ lpar\ Parlst\ rpar\ Result\ semicolon \\ Pcall & \rightarrow & callsym\ id\ lpar\ Arglst\ rpar \\ Parlst & \rightarrow & Par\ Rstpars \\ Rstpars & \rightarrow & comma\ Parlst \mid Empty \\ Arglst & \rightarrow & Arg\ Rstargs \\ Rstargs & \rightarrow & comma\ Arglst \mid Empty \\ Par & \rightarrow & intsym \mid charsym \\ Result & \rightarrow & Par \mid voidsym \\ Arg & \rightarrow & Pcall \mid int \mid character \\ Empty & \rightarrow & \end{array}$$

Een voorbeeld uit dit taaltje is:

```
proc a (int,int) int;
proc b (char,int) void;
call b('c',call a(4,call a(12,17)))
```

Voorzie deze syntaxregels van attributen en rekenvoorschriften, zodanig dat:

- bij iedere declaratie alle later benodigde gegevens (globaal) worden opgeslagen;
- bij iedere aanroep wordt gecontroleerd op:
  - gedeclareerd zijn,
  - dat het aantal parameters en argumenten gelijk is,
  - dat type parameter en argument overeenkomt.

Geef bij elk grammatica symbool ook aan welke attributen erbij horen, en van ieder attribuut of het inherited danwel synthesized is.
Men mag van de volgende gegevens gebruik maken (alle overige benodigde procedures, types etc. dient U apart te specificeren!):

```
TYPE tsymbol = (comma,semicolon,id,int,character,
                intsym,charsym,voidsym,procsym,callsym,
                lpar,rpar,endoffilesym);
(* De door de lexical scanner herkende terminal symbols.
   Het terminal symbol id heeft een lexicaal attribuut
   SymId  van integer type, zijnde een index in een stringtabel.
*)

VAR symtab: ... ;
(* Een tabel, waarin identifiers met type worden bewaard.
   De implementatie hiervan is grotendeels irrelevant.
*)

TYPE simple =  (inttipe,chartipe,voidtipe);

TYPE tipe   = RECORD tp: simple; (* is ook te gebruiken voor
                                    het result tp van een proc *)
                     parstp: ... (* definieer dit zelf!!! *)
              END;

FUNCTION IsDef (SymId: integer): boolean;
(* IsDef = "er is een id met id.SymId  in de tabel  symtab " *)

PROCEDURE StoreDef (SymId: integer; tp: tipe);
(* Slaat tp op in de tabel  symtab  onder: SymId *)

FUNCTION GetTp (SymId: integer): tipe;
(* Preconditie: IsDef(SymId);
   Resultaat: het tipe horend bij id in de symtab
*)
```

3. (45 minuten)

Gegeven is het volgende Pascal(-achtige) programma:

```
PROGRAM vb (input,output);

TYPE rij = array [1..3] of integer;

VAR a: rij;

PROCEDURE set_all (VAR a: rij; b: rij);
VAR i: integer;

   PROCEDURE p (c: rij);
   BEGIN b[i] := c[i]+2;                (* 1 *)
   END;

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

BEGIN ...
      set_all(a,a)                      (* 4 *)
      ...
END (* vb *).
```

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 het AR (activation record) voor procedure `set_all`;
- Geef de te genereren (pseudo-)instructies voor de **entry** en **exit** van `p`;
- Geef de te genereren (pseudo-)instructies voor de vier genummerde regels.

4. (40 minuten)

   De grammatica in som 2 is topdown parseerbaar. Een recursive descent parser bestaat onder andere uit een aantal procedures voor nonterminals van de grammatica.

   - Beschrijf in woorden hoe een recursive descent parser te werk gaat.
   - Geef de recursive descent code voor de nonterminals `Pdecls, Pcall` en `Arg`.
     Hierbij mag je gebruik maken van het volgende:

     ```
     Type tsymbol = ...;
     Var sym: tsymbol;
     procedure nextsym;
     procedure match(fs: tsymbol);
     ```

   - Doe nu hetzelfde als de vorige vraag, maar dan voor een parser *met error-recovery*.
     Hierbij mag je gebruik maken van het volgende:

     ```
     Type tsymbol = ...;
     Type tsymbolset = Set Of tsymbol;
     Var sym: tsymbol;
     Procedure nextsym;
     Procedure insertion (fs: tsymbol);
     Procedure delete (keys: tsymbolset);
     Procedure match (fs: tsymbol; keys: tsymbolset);
     ```