Tentamen Vertalerbouw—21 juni 2013

De nagekeken tentamens zijn in te zien op kamer Bernoulliborg 366.

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.

1. (40 minuten)

   a) Geef voor alle nonterminals uit onderstaande produkties de sets *first* en *follow*. Let goed op, maak geen rekenfouten!

   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.

   $$\begin{array}{lll} & S & \rightarrow & EF \\ , & E & \rightarrow & Fa \\ , & E & \rightarrow & ES \\ , & F & \rightarrow & b \\ , & F & \rightarrow & \end{array}$$

   Zie ommezijde voor de tweede opgave.

2. (50 minuten)

Gegeven is het volgende Pascal-achtige programma:

```
PROGRAM tentamen;

TYPE reeks = array [1..8] of int;

VAR r: reeks;
    i: int;

PROCEDURE bewerk (REF r: reeks; VAL a: int);
VAR i: int;

   PROCEDURE one (VAL i: int);
   BEGIN r[i] := a*i                          (* 1 *)
   END;

BEGIN FOR i := 1 TO 8 DO BEGIN
          one(i);                             (* 2 *)
          r[i] := r[i] + i                    (* 3 *)
      END
      ...
END (* bewerk *)

BEGIN ...
      bewerk(r,i)                             (* 4 *)
      ...
END (* tentamen *).
```

Hierbij staat REF voor een parameter via call by reference, VAL voor een parameter via call by value.

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.

- Teken de AR (activation record) voor procedure `bewerk`;
- Geef de te genereren (pseudo-)instructies voor de **entry** en **exit** van `one`;
- Geef de te genereren (pseudo-)instructies voor de vier genummerde regels. U hoeft geen arraybound checks te doen!

3. (40 minuten)

Gegeven zijn de volgende produkties:

Prog	→	*Pdecls Pcall*
Pdecls	→	*Pdecl Pdecls* | *Empty*
Pdecl	→	*procsym id lpar Parlst rpar Result semicolon*
Pcall	→	*callsym id lpar Arglst rpar*
Parlst	→	*Par Rstpars*
Rstpars	→	*comma Parlst* | *Empty*
Arglst	→	*Arg Rstargs*
Rstargs	→	*comma Arglst* | *Empty*
Par	→	*intsym* | *charsym*
Result	→	*Par* | *voidsym*
Arg	→	*Pcall* | *int* | *character*
Empty	→	

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)))
```

Deze grammatica is topdown parseerbaar.

Een recursive descent parser bestaat onder andere uit een aantal procedures voor nonterminals van de grammatica.

a) Laat zien dat deze grammatica topdoen (LL(1)) parseerbaar is.

b) Geef de recursive descent code voor de nonterminals `Pdecls`, `Pcall`, `Result` en `Arg`.
De parser hoeft alleen errors te signaleren (geen recovery). Hierbij mag je gebruik maken van het volgende:

```
Type tsymbol = ...; (* terminals *)
Var sym: tsymbol;   (* current symbol *)
procedure nextsym;  (* zet sym op eerstvolgende symbol *)
procedure match(fs: tsymbol); (* match een terminal *)
```