Tentamen Vertalerbouw—4 februari 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.

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

2. (45 minuten)

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

   $$\begin{array}{cccl} , & S & \rightarrow & (\ E\ )\ R \\ , & E & \rightarrow & E\ +\ T \\ , & E & \rightarrow & T \\ , & T & \rightarrow & T\ *\ F \\ , & T & \rightarrow & F \\ , & F & \rightarrow & N \\ , & F & \rightarrow & (\ E\ )\ R \\ , & N & \rightarrow & N\ dig \\ , & N & \rightarrow & dig \\ , & R & \rightarrow & \#\ dig \\ , & R & \rightarrow & \end{array}$$

   Hierbij is `dig` een terminalsymbol met attribuut `val` van type integer (de waarde van de digit). Ziehier driee (korrekte) voorbeelden uit dit taaltje:

```
(3*17 + 76)
(3*17 + 76) # 8
(101 * (12+1) # 3) # 2
```

Gevraagd wordt deze syntaxregels te voorzien van attributen en rekenvoorschriften. De volgende restricties dienen daarbij opgelegd te worden:

- `R` geeft de radix aan; default gebruiken wij radix 10 (het 10-tallig stelsel); in het derde voorbeeld worden het 2-tallig resp. 3-tallig stelsel gebruikt
- in de expressie `E` komen geen digits voor die gelijk of groter zijn dan de bijbehorende radix.

We willen de numerieke waarde van de expressie, gegeven de radix, via de attributen en rekenvoorschriften bepalen. Het symbol `S` heeft dus tenminste een attribuut `val` van type integer nodig. In de voorbeelden van zonet moet `S.val` dus de waarde 127 resp. 107 resp. 30 krijgen.

Geef ook bij elk grammatica symbool aan welke attributen daarbij horen, en of dat attribuut inherited danwel synthesized is.

3. (50 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 (a: rij; var b: rij);
VAR i: integer;

   PROCEDURE p (var c: rij);
   BEGIN c[i] := 10                    (* 1 *)
   END;

   FUNCTION q (c: rij): integer;
   BEGIN q := 2 * c[i]                 (* 2 *)
   END;

BEGIN FOR i := 1 TO 3 DO BEGIN
          p(a);                        (* 3 *)
          b[i] := q(b)                 (* 4 *)
      END
END (* set_all *)

BEGIN ...
      set_all(a,a)                     (* 5 *)
      ...
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.

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

4. (40 minuten)

Gegeven zijn de volgende grammaticaregels.

$$
\begin{array}{lrcl}
P = & \{ & & \\
 & S & \rightarrow & E \\
, & E & \rightarrow & F\ X \\
, & X & \rightarrow & \Leftarrow F\ X \\
, & X & \rightarrow & \Rightarrow F\ X \\
, & X & \rightarrow & \\
, & F & \rightarrow & a \\
, & F & \rightarrow & (S) \\
 & \} & &
\end{array}
$$

Schrijf een topdown parser met expliciete stapel, die de taal accepteert als beschreven door deze grammatica. De parser dient bij fouten aan error-recovery te doen.

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

```
TYPE
  symbol    = (S,E,X,F,
               lpar, rpar, leftarrow, rightarrow, asym, eofs);
  tsymbol   = lpar..eofs;

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 *)

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

Verder is gegeven een abstract data type (ADT) stack:

```
PROCEDURE initstack;
  (* Creeert een lege stack *)

FUNCTION isempty: boolean;
  (* checks if stack is empty *)

FUNCTION top: symbol;
  (* returns the symbol on top of the stack *)

PROCEDURE push (s: symbol);
  (* pushes s onto the stack *)

PROCEDURE pop;
  (* pops one symbol from the stack *)

PROCEDURE copystack;
  (* Copieert de inhoud van de stack naar een veilige plaats *)

PROCEDURE resetstack;
  (* Plaatst de inhoud van de stack terug vanuit de veilige plaats *)
```