CS324 – LANGUAGE PROCESSORS

MID-SEMESTER EXAMINATION

DATE: 22nd FEBRUARY 2002 **Weightage: 15%**
Maximum Marks: 30 **Closed Notes/Books Exam**

---

1. Consider a variant of FORTRAN defined using the grammar below:

   $sts \rightarrow sts\ \ st; \mid st;$
   $st \rightarrow assgn \mid do \mid if \mid labassgn \mid goto$
   $term \rightarrow \texttt{id} \mid \texttt{num} \mid \texttt{real} \mid term + term$
   $if \rightarrow \texttt{IF}\ (cond)\ goto$
   $do \rightarrow \texttt{DO}\ \texttt{num}\ \texttt{id} = \texttt{num}\ ,\ \texttt{num}$
   $assgn \rightarrow \texttt{id} = term \mid \texttt{id}\ (\ \texttt{id}\ ) = term$
   $labassgn \rightarrow \texttt{num}\ assgn$
   $cond \rightarrow \texttt{id} \mid term\ \texttt{.GT.}\ term$
   $goto \rightarrow \texttt{GOTO}\ \texttt{num}$

   `id` is defined in the usual way. A `num` is a natural number, and a `real` is, in general, two `num`s separated by a decimal point. Only one of the two `num`s may be absent.

   Note that in the earlier versions of FORTRAN, white spaces did not play any role. Thus a preprocessing stage would strip all the white spaces from the grammar and pass the result to the scanner. Further, keywords can be used as identifiers. Thus:

   (a) `DO10I=1,25;` and `DO10I=1.25;` are both valid statements. The first is a *do* and the second is an *assgn*.

   (b) `IF(A)GOTO30;` and `IF(A)=3.4;` are both valid statements, the first is an *if*, the second an *assgn*.

   Write a JFlex script just for detecting tokens (no declarations, no actions) in the language described above. The script should use the lookahead operator and not use states. (7 Marks)

2. For the grammar shown below, answer the following questions with explanations:

   $S \rightarrow \texttt{id}\ [\ E\ ]\ := E$
   $E \rightarrow E + T \mid T$
   $T \rightarrow T * F \mid F$
   $F \rightarrow (\ E\ ) \mid \texttt{id}$

   (a) In the automaton of the SLR(1) parser for this grammar, there is a state with a self loop? Identify the state through its shortest viable prefix.

   (b) Is the behavior of the SLR(1) parser and the LR(1) parser identical for *all strings*? No explanation is required if answer is yes. However, if the 'yes' answer is wrong, it will attract penalty.

(c) Is the behavior of the LALR(1) parser and the LR(1) parser identical for *all strings*? No explanation is required if answer is yes, but a wrong answer will attract penalty. $(3 \times 2 = 6$ Marks)

3. Consider the following grammar:

$$S \rightarrow A_1\ b_1 \mid A_2\ b_2 \mid A_3\ b_3$$
$$A_1 \rightarrow a_2\ A_1 \mid a_3\ A_1 \mid a_2 \mid a_3$$
$$A_2 \rightarrow a_1\ A_2 \mid a_3\ A_2 \mid a_1 \mid a_3$$
$$A_3 \rightarrow a_1\ A_3 \mid a_2\ A_3 \mid a_1 \mid a_2$$

Now answer the following questions with explanations. No credit if the explanation is wrong.

(a) Is the grammar ambiguous?

(b) Is the grammar $LL(n)$ for some value of n?

(c) Is the grammar $SLR(n)$ for some value of n?

(d) Is the grammar $LALR(n)$ for some value of n? $(1 + 3 \times 2 = 7$ Marks)

4. Given below is the grammar that generates binary numbers (including those with a fractional part):

$$S \rightarrow L_1\ .\ L_2 \mid L$$
$$L \rightarrow L_1\ B \mid B$$
$$B \rightarrow 0 \mid 1$$

Write a L-attributed definition to calculate the decimal value of S in an attribute called *val*. Clearly explain the meaning of the attributes and functions that you will be using. (5 Marks)

5. Consider a grammar to generate C-style expressions.

$$S \rightarrow E$$
$$E \rightarrow E_1 = E_2$$
$$E \rightarrow E_1 + E_2$$
$$E \rightarrow \texttt{id}\ [\ E_1\ ]$$
$$E \rightarrow (\ E_1\ )$$
$$E \rightarrow \texttt{id}$$

I am assuming that you know what it means to say that an expression has a *l-value*. An assignment is `valid` if the expression to the left of `=` has a l-value. And an expression is valid if all assignments within it are valid. Thus `a = b = c` is valid, only if it is parsed as `a = (b = c)` and not as `(a = b) = c`.

Write a L-attributed definition to determine an attribute *valid* of S which will evaluate to true only when the sentence generated from it is valid. You might like to do this in terms of an attribute *has_l* of E which determines whether the expression E has a l-value. Use any other attributes if necessary, however explain what they mean.

(5 Marks)