CS 515 Programming Languages and Compilers I
Problem Set 4
Fall 2018
Homeworks are not graded

Problem 1 - LR(1) canonical collection

1. Compute the canonical collection of sets of LR(1) items
2. Construct the LR(1) parse table
3. Is the grammar LR(1) or not? Justify your answer.
4. If the grammar is LR(1), show the behavior of the LR(1) parser on input TRUE or TRUE and FALSE, i.e., show stack content, current input, and selected action for each move of the machine.

Problem 2 - LR vs. SLR

Show that the following grammar is LR(1) but not SLR(1):

\[ S ::= Aa \mid bAc \mid dc \mid bda \]
\[ A ::= d \]
Problem 3 - Syntax-Directed Translation (STD) Schemes

The following context-free grammar describes a simple imperative programming language. As a matter of notation, non-terminals are in **CAPITALS**, and terminals are in **lower case**. “id” represents an identifier and “const” represents an integer constant.

```
PROGRAM ::= procedure STMT_LIST
STMT_LIST ::= STMT ; STMT_LIST
   | STMT
STMT ::= FOR_STMT
   | A_STMT
   | READ_STMT
   | WRITE_STMT
FOR_STMT ::= for id := const to const begin STMT_LIST end
A_STMT ::= id := EXPR
READ_STMT ::= read(id)
WRITE_STMT ::= write(EXPR)
EXPR ::= EXPR + EXPR
   | EXPR * EXPR
   | id
   | const
```

The token **const** has the attribute **val** which is a (synthesized) attribute and assigned by the scanner. It contains the integer value of the constant.

The number of times that the body of a **for-loop** executes can be computed from the two constants that specify the range of the loop variable.

**Example:**

```
procedure
  read(a);       // references: a
  a := b + 5;   // references: a, b
  c := a + b;   // references: a, b, c
  write(c);     // references: c
for i := 1 to 10 // references: 10 * i, 30 * a
  begin
    a := a + 1;   // references: a (twice)
    write (a)     // references: a
  end
```

The **for** statement executes its body exactly ten times. Overall, the program references variable “a” 33 times, variables “b” and “c” twice, and variable “i” 10 times (induction variable).

Define a syntax-directed translation scheme (including appropriate attribute assignments). Use a YACC-like notation to access the attribute instances.