PLC Test 2

Name: Ethan Tang

a.)

Tokens	Regular Expression
INT_LIT	(/d+)
VAR_IDENT	([A-Za-z_]{6, 8})
INT_TYPE_BYTE	(#b)
INT_TYPE_WORD	(#w)
INT_TYPE_DWORD	(#d)
INT_TYPE_QWORD	(#q)
IF_STMT	(\$i)
FOR_LOOP	(\$f)
WHILE_LOOP	(\$w)
BEGIN	(!b)
END	(!e)

b.)

Order of Operation: BDSMAO

Brackets first

Division and Subtraction (left to right)

Multiplication and Addition (left to right)

Order(Exponents)

<program> → !b <stmtlist> !e	Not Left Recursive
<stmtlist> → <stmt> | <stmt>; <stmtlist>	Not Left Recursive
<stmt> → <if_stmt> | <while> | <declr> | <for>	Not Left Recursive
<if_stmt> → `$i``(`<bool_expr>`)`<stmt>	Not Left Recursive
<while> → `$w``(`<bool_expr>`)`<stmt>	Not Left Recursive
<for> → `$f` `(` INT_TYPE `>>` INT_LIT `)``:`<stmt>	Not Left Recursive
<declr> → VAR_IDEN `=` <expr>	Not Left Recursive
<expr> → <term> { (`*`|`+`) <term>}	Not Left Recursive
<term> → <factor> { (`\`|`-`|`%`) <factor>}	Not Left Recursive
<factor> → VAR_IDEN | INT_LIT | `(` <expr> `)`	Not Left Recursive
<bool_expr> → <brel> { (`!=`|`==`) <brel>}	Not Left Recursive
<brel> → <bexpr> { (`>=`|`<=`|`<`|`>`) <bexpr>}	Not Left Recursive
<bexpr> → <bterm> { (`*`|`+`) <bterm>}	Not Left Recursive
<bterm> → <bfactor> { (`\`|`-`|`%`) <bfactor>}	Not Left Recursive
<bfactor> → VAR_IDEN | INT_LIT | `(` <expr> `)`	Not Left Recursive

c.)

None of the grammar rules' LHS is appearing at the beginning of the RHS so this language's rule sets are not left recursive.

This grammar rule set is also not ambiguous because the LHS of the rules does not appear in its RHS hence there is only one derivation tree.

This language conforms to the LL grammar because it is not left recursive and unambiguous.

d.)

This language is unambiguous as explained in part c.

e.)

Code written in Lexeme.java

f.)

Code written in Syntax.java

g.)

First file test1.in has 5 lexical errors. They are represented by error code 444.

Error 1: #, invalid because # had to be followed by b, w, d, or q to represent a valid number type.

Error 2: q, invalid identifier. Identifier has to consist of 6 to 8 letters or underscores

Error 3: $d, invalid keyword. $d lexeme does not exist

Error 4: notvr, invalid identifier. Identifier has to consist of 6 to 8 letters or underscores

Error 5: ><, invalid relational operator. >< does not exist as a lexeme.

Second file test2.in has 5 syntax errors. They are scanned by Syntax.java. Any errors detected will force the program to end early.

Error 1: Line 2, missing integer type on variable assignment.

Error 2: Line 3, missing semicolon at the end of the statement.

Error 3: Line 4, missing UNTIL token after the first identifier inside the for loop condition.

Error 4: Line 5, unidentified relational operator >> inside the while loop condition.

Error 5: line 6, missing identifier on variable declaration.

Third file test3.in and fourth file test4.in have no error.