METROPOLITAN STATE UNIVERSITY OF DENVER
Department of Mathematical and Computer Sciences

CS 3210 - Principles of Programming Languages (Fall 2019)

Programming Assignment 01

Deadline: September 29th 11:59pm

1. Introduction

The goal of this assignment is to have you write a lexical and syntax analyzer for a hypothetical programming language (roughly based on Pascal). The input of your parser is a source code written in the programming language's grammar. If the source code is syntactically correct your parser should display the parse tree. An appropriate error message should be provided otherwise.

The expectation for this programming assignment is that you will implement a parser for the given PL using the bottom-up shift-reduce algorithm discussed in class. You are encouraged to use and modify the code used in class. The LR table for the parser can be generated from the PL's grammar using the online tool available here, also discussed in class.

2. Grammar

Below is the grammar for the programming language specified using EBNF notation. Special words and symbols are highlighted for easy identification. Assume that this PL is NOT case-sensitive.

```
1. <program> → program <identifier> <block> .
2. <identifier> → <letter> { <letter> | <digit> }
3. <block> → [ <var_declaration_section> ] <compound_statement>
4. <var_declaration_section> → var <var_declaration> {; <var_declaration> }
5. <var_declaration> → <identifier> { <identifier> } : <type>
6. <type> → Integer | Boolean
7. <compound_statement> → begin <statement> { ; <statement> } end
8. <statement> → <simple_statement> | <structured_statement>
9. <simple_statement> → <assignment_statement> | <read_statement> |
   <write_statement>
10.<assignment_statement> → <identifier> := <expression>
11.<read_statement> → read <identifier>
12.<write_statement> → write ( <identifier> | <literal> )
13.<structured_statement> → <if_statement> | <while_statement> |
   <compound_statement>
14.<if_statement> → if <boolean_expression> then <statement> [ else
```

```
   <statement> ]
15.<while_statemnet> → while <boolean_expression> do <statement>
16.<expression> → <arithmetic_expression> | <boolean_expression>
17.<arithmetic_expression> → <arithmetic_expression> ( + | - ) <term> |
   <term>
18.<term> → <term> * <factor> | <factor>
19.<factor> → <identifier> | <integer_literal>
20.<literal> → <integer_literal> | <boolean_literal>
21.<integer_literal> → <digit> { <digit> }
22.<boolean_literal> → true | false
23.<boolean_expression> → <boolean_literal> | <arithmetic_expression> ( > |
   >= | = | <= | < ) <arithmetic_expression>
```

3. Token Table

For this project you MUST use the following token table (codes and descriptions must match).

Token#	Description	Token #	Description	Token #	Description
1	ADDITION	11	GREATER	21	PROGRAM
2	ASSIGNMENT	12	GREATER_EQUAL	22	READ
3	BEGIN	13	IDENTIFIER	23	SEMICOLON
4	BOOLEAN_TYPE	14	IF	24	SUBTRACTION
5	COLON	15	INTEGER_LITERAL	25	THEN
6	DO	16	INTEGER_TYPE	26	TRUE
7	ELSE	17	LESS	27	VAR
8	END	18	LESS_EQUAL	28	WHILE
9	EQUAL	19	MULTIPLICATION	29	WRITE
10	FALSE	20	PERIOD	30	

4. Error Table

For this project you MUST use the following error table (codes and descriptions must match). Optionally, you can add more specific errors by creating new codes between 11 and 99.

Error#	Description
1	Source file missing
2	Couldn't open source file
3	Lexical error
4	Couldn't open grammar file
5	Couldn't open SLR table file
6	EOF expected
7	Identifier expected
8	Special word missing
9	Symbol missing
10	Data type expected
11	Identifier or literal value expected
99	Syntax error

5. Deliverables and Submission

Below is the list of minimum deliverables for this project:

- `parser.xxx` source code (e.g., `parser.py` or `source.java`)
- `grammar.txt` file, and
- `slr_table.txt` file.

If you are writing your parser in a PL other than Python or Java you MUST provide specific instructions on how to deploy your code, including IDE/compiler used (with version numbers) and how to compile/run your code. I should be able to test your code using MacOS. So if you are using a different platform I encourage you to contact me ahead of the deadline so I can properly set up my computer.

Your source code MUST have a comment section in the beginning with the name(s) of the author(s) of the project. You are allowed to work together with another classmate. Teams of more than two students will NOT be accepted (NO exceptions). Only one of the members of the team needs to submit on Blackboard.

Please use ZIP format when submitting your project (no Z, RAR, or any other format).

6. Running and Testing

Your parser should accept a source code from the command-line. Also, your code should assume that the grammar and the LR table are provided by files named `grammar.txt` and `slr_table.csv`, respectively. Those files are expected to be at the same folder where the parser code is located. The format of those files must match the one used in class.

For testing purposes, 15 source files are provided (download link here). The parser expects that source files should be at the same folder. Source files from 1-3 should be parsed without any errors. The table below summarizes the expected results when running the parser for each of the source files.

Source File	Expected Result
source1.pas	Success!
source2.pas	Success!
source3.pas	Success!
source4.pas	Error 07: identifier expected
source5.pas	Error 08: special word missing
source6.pas	Error 99: syntax error
source7.pas	Error 09: symbol missing
source8.pas	Error 10: data type expected
source9.pas	Error 99: syntax error
source10.pas	Error 09: symbol missing
source11.pas	Error 09: symbol missing
source12.pas	Error 11: identifier or literal value expected
source13.pas	Error 99: syntax error
source14.pas	Error 99: syntax error
source15.pas	Error 06: EOF expected

I should warn you that those tests are far from being comprehensive. The instructor reserves the right to run other tests on your code if deemed necessary. You are encouraged to create other tests on your own.

6. Rubric

This programming assignment is worth 100 points, distributed in the following way:

+1 command-line validation
+2 grammar is loaded correctly
+2 SLR table is loaded correctly
+30 lexical analyzer works as expected
    +5 token codes match specification
    +2 recognizes EOF
    +5 recognizes identifiers
    +3 recognizes integer literals
    +5 recognizes special words
    +2 recognizes assignment operator
    +3 recognizes relational operators
    +3 recognizes punctuators
    +2 raises exception with proper error message when it fails
+60 syntax analyzer works as expected
    +10 grammar used matches specification
    +10 SLR table is correct
    +8 shift operation is implemented correctly
    +12 reduce operation is implemented correctly
    +5 accept operation is implemented correctly
    + 15 parse tree is built correctly
+5 submission follows instructions (student names identified, zip format, source/grammar/slr table submitted, specific instructions provided when using different development platform etc.)

10 points will be deducted for each day of late submission. I will not accept submissions that are five days (or more) late.