CSCI3136
Assignment 5

Instructor: Alex Brodsky

Due: 9:00am, Friday, June 28, 2019

1. **[40 marks]** Write a recursive descent parser for the language generated by the grammar:

$$
\begin{array}{rcl}
S & \rightarrow & \epsilon \\
 & \rightarrow & E\_LIST \\
E\_LIST & \rightarrow & EXPR\ E\_TAIL \\
E\_TAIL & \rightarrow & \epsilon \\
 & \rightarrow & E\_LIST \\
EXPR & \rightarrow & S\_EXPR \\
S\_EXPR & \rightarrow & ANDOP\ S\_TAIL \\
S\_TAIL & \rightarrow & \epsilon \\
 & \rightarrow & '|' S\_EXPR \\
ANDOP & \rightarrow & RELOP\ A\_TAIL \\
A\_TAIL & \rightarrow & \epsilon \\
 & \rightarrow & '\&'\ ANDOP \\
RELOP & \rightarrow & TERM\ R\_TAIL \\
R\_TAIL & \rightarrow & \epsilon \\
 & \rightarrow & '<'\ RELOP \\
 & \rightarrow & '>'\ RELOP \\
 & \rightarrow & '='\ RELOP \\
 & \rightarrow & '\#'\ RELOP \\
TERM & \rightarrow & FACT\ T\_TAIL \\
T\_TAIL & \rightarrow & \epsilon \\
 & \rightarrow & '+'\ TERM \\
 & \rightarrow & '-'\ TERM
\end{array}
\qquad
\begin{array}{rcl}
FACT & \rightarrow & VALUE\ F\_TAIL \\
F\_TAIL & \rightarrow & \epsilon \\
 & \rightarrow & '*'\ FACT \\
 & \rightarrow & '/'\ FACT \\
VALUE & \rightarrow & LIST \\
 & \rightarrow & UNARY \\
 & \rightarrow & LITERAL \\
 & \rightarrow & '('\ EXPR\ ')' \\
 & \rightarrow & SYMBOL \\
LIST & \rightarrow & '['\ ARGS\ ']' \\
UNARY & \rightarrow & '-'\ VALUE \\
 & \rightarrow & '!'\ VALUE \\
ARGS & \rightarrow & \epsilon \\
 & \rightarrow & EXPR\ A\_LIST \\
A\_LIST & \rightarrow & \epsilon \\
 & \rightarrow & ','\ EXPRA\_LIST \\
SYMBOL & \rightarrow & symbol \\
LITERAL & \rightarrow & integer \\
 & \rightarrow & string \\
 & \rightarrow & 'true' \\
 & \rightarrow & 'false' \\
 & \rightarrow & 'nil'
\end{array}
$$

Figure 1: A grammar for the Splat language.

The terminal `int` denotes an integer, `string` denotes a double quoted string, e.g., `"hello world"` and the terminal `symbol` denotes a symbol, e.g., *myVar*.

You should use the scanner that you built (or the solution provided) as the front end to the parser. That is, the scanner will take input from `stdin` and generate tokens, which the parser will then use. See the provided test cases for examples of input.

The output of your parser should be the list of productions being applied (in order) as the parser parses the input. If the token stream to your parser represents a valid program, the parser should terminate after all productions have been applied (and printed). If the parser cannot finish parsing (the token stream does not represent a valid program, the last line of output should be: `Syntax Error`

The format of the productions that are to printed by the parser as they are being applied are of the form: `LHS -> RHS` where the LHS is a variable as shown in Figure 1 and the RHS consists of terminals and and variables, where terminals are depicted in single quotes, except for integers, strings, and symbols.

`symbol` is denoted by `symbol(`x`)` where x is the symbol name, e.g., `symbol(myVar)`

`integer` is denoted by `int(`v`)` where v is the integer value, e.g., `int(42)`

`string` is denoted by `string(`s`)` where s is the string in double quotes, e.g., `string("Hello World")`

For example, the outputs for the following the program

Program: `"Hello " + myVar`	Program: `"42 +`
`S -> E_LIST`	`S -> E_LIST`
`E_LIST -> EXPR E_TAIL`	`E_LIST -> EXPR E_TAIL`
`EXPR -> SEXPR`	`EXPR -> SEXPR`
`SEXPR -> ANDOP S_TAIL`	`SEXPR -> ANDOP S_TAIL`
`AMDOP -> RELOP A_TAIL`	`AMDOP -> RELOP A_TAIL`
`RELOP -> TERM R_TAIL`	`RELOP -> TERM R_TAIL`
`TERM -> FACTOR T_TAIL`	`TERM -> FACTOR T_TAIL`
`FACTOR -> VALUE F_TAIL`	`FACTOR -> VALUE F_TAIL`
`VALUE -> LITERAL`	`VALUE -> LITERAL`
`LITERAL -> string("Hello")`	`LITERAL -> int(42)`
`F_TAIL -> epsilon`	`F_TAIL -> epsilon`
`T_TAIL -> '+' TERM`	`T_TAIL -> '+' TERM`
`TERM -> FACTOR T_TAIL`	`TERM -> FACTOR T_TAIL`
`FACTOR -> VALUE F_TAIL`	`FACTOR -> VALUE F_TAIL`
`VALUE -> SYMBOL`	`Syntax Error`
`SYMBOL -> symbol(myVar)`	
`F_TAIL -> epsilon`	
`T_TAIL -> epsilon`	
`R_TAIL -> epsilon`	
`A_TAIL -> epsilon`	
`S_TAIL -> epsilon`	
`E_TAIL -> epsilon`	

A set of test cases is provided on Brightspace in the file `tests_5.zip` The sample solution takes about 160 lines of code (not counting the scanner).

You may implement your parser in any language that you wish, but it must be runnable on the `bluenose` server. Since the choice of language is up to you, you must provide a standard script called `runme.sh` to run your parser. For example the script for a Java solution looks like:

```
#!/bin/sh
# if your implementation requires compilation include the command here
javac SplatParser.java

# Command to run your program
java SplatParser
```

Please submit your code, along with a `runme.sh` script via brightspace.

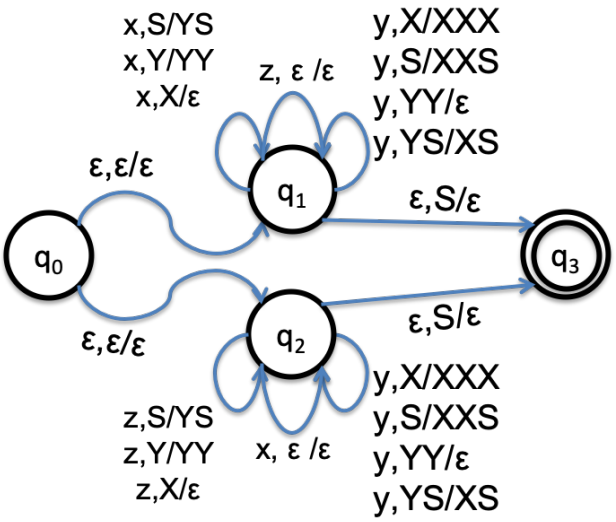
Please see solution posted as starter code for Assignment 7.

2. **[10 marks]** Give a PDA (Pushdown Automata) that recognizes the language

$$L = \{\sigma \in \{x, y, z\}^* \mid 2|\sigma|_x = |\sigma|_y \vee 2|\sigma|_y = |\sigma|_z\}$$

You can choose whether your PDA accepts by empty stack or final state, but make sure you clearly note, which acceptance is assumed.

The following PDA accepts both with empty stack and final state. The initial symbol on the stack is S, and the stack alphabet comprises $\{S, X, Y\}$. The PDA nondeterministically decides whether it is to check if the input has twice as many x's as y's or vice versa. Without loss of generallity assume the former. It keeps track of twice the number of x's minus number of y's. It uses Xs to denote a positive number and Y to denote a negative number.



CSCI3136: Assignment 5

Summer 2019

Student Name	Login ID	Student Number	Student Signature

	Mark	
Question 1		/40
Functionality	/20	
Structure	/20	
Question 2		/10
Total		**/50**

Comments:

Assignments are due by 9:00am on the due date. Assignments *must* be submitted electronically via Brightspace. Please submit zip file of the code and a PDF for the written work. You can do your work on paper and then scan in and submit the assignment.

Plagiarism in assignment answers will not be tolerated. By submitting their answers to this assignment, the authors named above declare that its content is their original work and that they did not use any sources for its preparation other than the class notes, the textbook, and ones explicitly acknowledged in the answers. Any suspected act of plagiarism will be reported to the Facultys Academic Integrity Officer and possibly to the Senate Discipline Committee. The penalty for academic dishonesty may range from failing the course to expulsion from the university, in accordance with Dalhousie Universitys regulations regarding academic integrity.