

# 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:

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

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 be 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 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.

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.

## CSCI3136: Assignment 5

Summer 2019

Student Name	Login ID	Student Number	Student Signature

	Mark
<b>Question 1</b>	/40
Functionality	/20
Structure	/20
<b>Question 2</b>	/10
<b>Total</b>	<b>/50</b>

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 Faculty's 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 University's regulations regarding academic integrity.