CSCI3136
Assignment 6

Instructor: Alex Brodsky

Due: 9:00am, Friday, July 5, 2019

Consider the grammar for the Splat programming language in Figure 1, where the terminal `int` denotes an integer, the terminal `symbol`, and the terminal `string` denotes a quoted string.

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

Figure 1: A grammar for the Splat language.

Splat, is a functional language where all programs consist of one or more expressions. For now, an expression can be one of:

arithmetic expressions such as `1 + 2 * 3`
boolean expressions such as `1 < 2 & 3 > 2`
string expressions such as `"Hello " + "world!"`
list expressions such as `["a", "b", "c"]` and `[3, 1, 2] + [4, 5, 6]`

Expressions in Splat are evaluated in the standard way. For example,

`1 + 2 * 3` evaluates to 7
`1 < 2 & 3 > 2` evaluates to *true*
`"Hello " + "world!"` evaluates to `"Hello world"`
`[3, 1, 2] + [4, 5, 6]` evaluates to `[3, 1, 2, 4, 5, 6]`

Formally, an expression is either a literal

integer such as 42,
boolean such as *true*
string such as *"Hello"*
list of constants such as `[1,2,3]`

or a composition of subexpressions and operators such as above. The expression is evaluated by applying the operators just like in most programming languages. The operators in Splat are:

Operator	Description
`+`	add two integers concatenate two strings concatenate two lists
`-`	subtract one integer expression from another integer expression, e.g., `3 - 2` negate an integer expression, e.g., `-42`
`*, /`	multiply and divide integer expressions
`<, >, =, #`	compare two expressions: less than, greater than, equals, not equals
`&, |, !`	combine boolean expressions with *and*, *or*, and *not*

Table 1: Operators in Splat.

All expressions are evaluated in the same way as in most other languages such as Java or Python. The *list* expression is evaluated by evaluating each of the elements of the list. E.g., the evaluation of `[1+2, 3+4, "Hello"]` is `[3, 7 "Hello"]`.

Using a parse tree of a Splat expression, we want to evaluate the expression.

Note: This assignment feeds directly into Assignment 7, which is released at the same time as assignment 6. In Assignment 7, you will be asked to implement the attribute grammar you develop for this assignment.

1. [**5 marks**] Assume that the *S_EXPR* symbol has a synthesized attribute called *val*, which stores the evaluation of an expression. **Give an attribute grammar based on the grammar above that will perform the evaluation of a list of expressions.** I.e., evaluate the expression represented by the parse tree whose root is S, assuming that everything below *S_EXPR* has already been evaluated, and that *S_EXPR.val* stores the result of the evaluation. I.e., assume that the leaves of the parse tree are *S_EXPR* symbols.

Note: All attributes are synthesized.

Symbol	Attribute
S	*list*
E_LIST	*list*
E_TAIL	*list*
$EXPR$	*val*
S_EXPR	*val*

Production			Semantic Rule
S	$\rightarrow$	ϵ	$\triangleright$ $S.list = []$
	$\rightarrow$	E_LIST_1	$\triangleright$ $S.list = E\_LIST_1.list$
E_LIST	$\rightarrow$	$EXPR_1\ E\_TAIL_1$	$\triangleright$ $E\_LIST.list = EXPR_1.val + E\_TAIL_1.list$
E_TAIL	$\rightarrow$	ϵ	$\triangleright$ $E\_TAIL.list = []$
	$\rightarrow$	E_LIST_1	$\triangleright$ $E\_TAIL.list = E\_LIST_1.list$
$EXPR$	$\rightarrow$	S_EXPR_1	$\triangleright$ $EXPR.val = S\_EXPR_1.val$

2. **[15 marks]** Assume that the *EXPR* symbol has a synthesized attribute called *val*, which stores the evaluation of an expression. Furthermore, assume that *VALUE* also has a synthesized attribute called *val*. **Give an attribute grammar based on the grammar above that will perform the evaluation of a** *VALUE***.** I.e., if *VALUE* is the start symbol, evaluate the expression represented by the parse tree whose root is *VALUE*.

   Note: You can use pseudocode in your semantic rules. For example, if L is a list, you can add to the list using pseudocode such as $L.add(e)$. I.e., feel free to make use of standard data structures and operations in your semantic rules.

Note: All attributes are synthesized.

Symbol	Attribute
$VALUE$	*val*
$LIST$	*list*
$UNARY$	*val*
$ARGS$	*list*
A_LIST	*list*
$SYMBOL$	*val*
$LITERAL$	*val*

		Production	Semantic Rule
$VALUE$	$\rightarrow$	$LIST_1$	$\triangleright$ $VALUE.val = LIST_1.list$
	$\rightarrow$	$UNARY_1$	$\triangleright$ $VALUE.val = UNARY_1.val$
	$\rightarrow$	$LITERAL_1$	$\triangleright$ $VALUE.val = LITERAL_1.val$
	$\rightarrow$	$'(' \ EXPR_1 \ ')'$	$\triangleright$ $VALUE.val = EXPR_1.val$
	$\rightarrow$	$SYMBOL_1$	$\triangleright$ $VALUE.val = SYMBOL_1.val$
$LIST$	$\rightarrow$	$'[' \ ARGS_1 \ ']'_1$	$\triangleright$ $LIST.list = ARGS_1.list$
$UNARY$	$\rightarrow$	$'-' \ VALUE_1$	$\triangleright$ $UNARY.val = -VALUE_1.val$
	$\rightarrow$	$'!' \ VALUE_1$	$\triangleright$ $UNARY.val = !VALUE_1.val$
$ARGS$	$\rightarrow$	ϵ	$\triangleright$ $ARGS.list = []$
	$\rightarrow$	$EXPR_1 \ A\_LIST_1$	$\triangleright$ $ARGS.list = EXPR_1.val + A\_LIST_1.list$
A_LIST	$\rightarrow$	ϵ	$\triangleright$ $A\_LIST.list = []$
	$\rightarrow$	$',' \ EXPR_1 A\_LIST_1$	$\triangleright$ $A\_LIST.list = EXPR_1.val + A\_LIST_1.list$
$SYMBOL$	$\rightarrow$	$symbol_1$	$\triangleright$ $SYMBOL.val = symbol(s)$
$LITERAL$	$\rightarrow$	$integer_1$	$\triangleright$ $LITERAL.val = integer(s)$
	$\rightarrow$	$string_1$	$\triangleright$ $LITERAL.val = string(s)$
	$\rightarrow$	$'true'_1$	$\triangleright$ $LITERAL.val = 'true'$
	$\rightarrow$	$'false'_1$	$\triangleright$ $LITERAL.val = 'false'$
	$\rightarrow$	$'nil'_1$	$\triangleright$ $LITERAL.val = 'nil'$

3. **[30 marks]** Assume that the *VALUE* symbol has a synthesized attribute called *val*, which stores the evaluation of an expression. Furthermore, assume that *S_EXPR* also has a synthesized attribute called *val*. **Give an attribute grammar based on the grammar above that will perform the evaluation of a *S_EXPR*.** I.e., if *S_EXPR* is the start symbol, evaluate the expression represented by the parse tree whose root is *S_EXPR*, assuming that the leaves of the tree are *VALUE* symbols.

   Hint: You will need to use inherited attributes to do this. You can use the operators from Table 1 directly in your semantic rules.

Note: The I in the following two tables indicates inherited attributes/semantic rules.

Symbol	Attribute
EXPR	*val*
S_EXPR	*tmp*(**I**)
	val
S_TAIL	*tmp*(**I**)
	val
ANDOP	*tmp*(**I**)
	val
A_TAIL	*tmp*(**I**)
	val
RELOP	*tmp*(**I**)
	op(**I**)
	val

Symbol	Attribute
R_TAIL	*tmp*(**I**)
	op(**I**)
	val
TERM	*tmp*(**I**)
	val
T_TAIL	*tmp*(**I**)
	val
FACT	*tmp*(**I**)
	op(**I**)
	val
F_TAIL	*tmp*(**I**)
	val
VALUE	*val*

Production				Semantic Rule
$EXPR$	$\rightarrow$	S_EXPR_1		$\triangleright$ $EXPR.val = S\_EXPR_1.val$
S_EXPR	$\rightarrow$	$ANDOP_1\ S\_TAIL_1$	I	$\triangleright$ $S\_TAIL_1.tmp = S\_EXPR.tmp \mid ANDOP_1.val$
				$\triangleright$ $S\_EXPR.val = S\_TAIL_1.val$
S_TAIL	$\rightarrow$	ϵ		$\triangleright$ $S\_TAIL.val = S\_TAIL.tmp$
	$\rightarrow$	$'\mid' S\_EXPR_1$	I	$\triangleright$ $S\_EXPR_1.tmp = S\_TAIL.tmp$
				$\triangleright$ $S\_TAIL.val = EXPR_1.val$
$ANDOP$	$\rightarrow$	$RELOP_1\ A\_TAIL_1$	I	$\triangleright$ $A\_TAIL_1.tmp = ANDOP.tmp \mid RELOP_1.val$
				$\triangleright$ $ANDOP.val = A\_TAIL_1.val$
A_TAIL	$\rightarrow$	ϵ		$\triangleright$ $A\_TAIL.val = A\_TAIL.tmp$
	$\rightarrow$	$'\&' ANDOP_1$	I	$\triangleright$ $ANDOP_1.tmp = A\_TAIL.tmp$
				$\triangleright$ $A\_TAIL.val = ANDOP_1.val$
$RELOP$	$\rightarrow$	$TERM_1\ R\_TAIL_1$	I	$\triangleright$ $R\_TAIL_1.tmp = RELOP.tmp\ RELOP.op\ TERM_1.val$
				$\triangleright$ $RELOP.val = R\_TAIL_1.val$
R_TAIL	$\rightarrow$	ϵ		$\triangleright$ $R\_TAIL.val = R\_TAIL.tmp$
	$\rightarrow$	$' <'\ RELOP_1$	I	$\triangleright$ $RELOP_1.tmp = R\_TAIL.tmp$
			I	$\triangleright$ $RELOP_1.op = ' <'$
				$\triangleright$ $R\_TAIL.val = RELOP_1.val$
	$\rightarrow$	$' >'\ RELOP_1$	I	$\triangleright$ $RELOP_1.tmp = R\_TAIL.tmp$
			I	$\triangleright$ $RELOP_1.op = ' >'$
				$\triangleright$ $R\_TAIL.val = RELOP_1.val$
	$\rightarrow$	$' ='\ RELOP_1$	I	$\triangleright$ $RELOP_1.tmp = R\_TAIL.tmp$
			I	$\triangleright$ $RELOP_1.op = ' ='$
				$\triangleright$ $R\_TAIL.val = RELOP_1.val$
	$\rightarrow$	$'\#'\ RELOP_1$	I	$\triangleright$ $RELOP_1.tmp = R\_TAIL.tmp$
			I	$\triangleright$ $RELOP_1.op = '\#'$
				$\triangleright$ $R\_TAIL.val = RELOP_1.val$
$TERM$	$\rightarrow$	$FACT\ T\_TAIL_1$		$\triangleright$ $T\_TAIL_1.tmp = TERM.tmp\ TERM.op\ FACT_1.val$
				$\triangleright$ $TERM.val = T\_TAIL_1.val$
T_TAIL	$\rightarrow$	ϵ		$\triangleright$ $T\_TAIL.val = T\_TAIL.tmp$
	$\rightarrow$	$'+'\ TERM_1$	I	$\triangleright$ $TERM_1.tmp = T\_TAIL.tmp$
			I	$\triangleright$ $TERM_1.op = '+'$
				$\triangleright$ $T\_TAIL.val = TERM_1.val$
	$\rightarrow$	$'-'\ TERM_1$	I	$\triangleright$ $TERM_1.tmp = T\_TAIL.tmp$
			I	$\triangleright$ $TERM_1.op = '-'$
				$\triangleright$ $T\_TAIL.val = TERM_1.val$
$FACT$	$\rightarrow$	$VALUE_1\ F\_TAIL_1$	I	$\triangleright$ $F\_TAIL_1.tmp = FACT.tmp\ FACT.op\ VALUE_1.val$
				$\triangleright$ $FACT.val = F\_TAIL_1.val$
F_TAIL	$\rightarrow$	$epsilon$		$\triangleright$ $F\_TAIL.val = F\_TAIL.tmp$
	$\rightarrow$	$'*'\ FACT_1$	I	$\triangleright$ $FACT_1.tmp = F\_TAIL.tmp$
			I	$\triangleright$ $FACT_1.op = '*'$
				$\triangleright$ $F\_TAIL.val = FACT_1.val$
	$\rightarrow$	$'/'\ FACT_1$	I	$\triangleright$ $FACT_1.tmp = F\_TAIL.tmp$
			I	$\triangleright$ $FACT_1.op = '/'$
				$\triangleright$ $F\_TAIL.val = FACT_1.val$

CSCI3136: Assignment 6

Summer 2019

Student Name	Login ID	Student Number	Student Signature

	Mark
Question 1	/5
Question 2	/15
Question 3	/30
Total	**/50**

Comments:

Assignments are due by 9:00am on the due date. Assignments *must* be submitted electronically via Brightspace. Please submit 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.