CSCI3136
Assignment 5

Instructor: Alex Brodsky

Due: 3:00pm, Friday, March 3, 2014

1. **[30 marks]** Write a recursive descent parser for the language generated by the grammar:

$$
\begin{array}{rcl}
S & \rightarrow & Atoms \\
Atoms & \rightarrow & \epsilon \\
Atoms & \rightarrow & Atom\ Atoms \\
Atom & \rightarrow & Quote \\
Atom & \rightarrow & List \\
Atom & \rightarrow & \texttt{id} \\
Atom & \rightarrow & \texttt{int} \\
Quote & \rightarrow & \texttt{'(}\ Atoms\ \texttt{)} \\
List & \rightarrow & \texttt{(}\ Proc\ Atoms\ \texttt{)} \\
Proc & \rightarrow & \texttt{id} \\
Proc & \rightarrow & List
\end{array}
$$

Figure 1: A simplified grammar for Scheme.

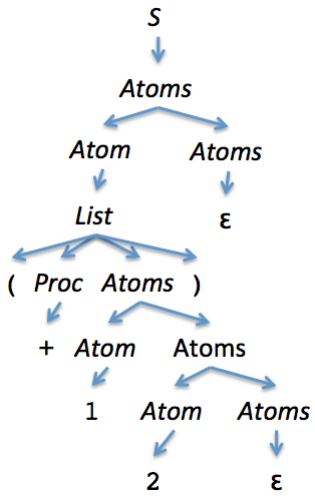

Figure 2: Parse tree for sentence `( + 1 2 )`

where the terminal `int` denotes an integer and the terminal `id` denotes an identifier. The only other terminals in this grammar are the quoted left parenthesis `'(`, the left parenthesis `(`, and the right parenthesis `)`.

You may implement your parser in any language that you wish, but it must be runnable on the `bluenose` server. The parser should take input from `stdin` and output to `stdout`. The input to the parser will comprise a stream of tokens, with the tokens being separated by white-space. Each token will be one of

Token (RE)	Description
`'(`	a *quoted left parenthesis*
`(`	a *left parenthesis*
`)`	a *right parenthesis*
`[0-9].*`	an *integer* if the first character is a digit
`..*`	an *identifier* if it is not any of the other tokens

For example, the following would be a possible input to your parser.

```
( define swap ( lambda ( tuple )
                ( list ( car ( cdr tuple ) ) ( car tuple ) )
              )
)
```

If the token stream input to your parser represents an invalid program, the output of your parser should be a single line message: `Syntax Error`

If the token stream input to your parser represents a valid program, the output of your parser should be the parse tree, represented as a *post-order* traversal of the parse tree, where each node has the following format: "[*Symbol*, *Children*]" where *Symbol* denotes the terminal, ϵ, or nonterminal corresponding to the node and *Children* is an integer denoting the number of children that the node has. For example, on input `( + 1 2)` the output is a post-order traversal of the tree in Figure 2 and should be:

```
[(,0] [+,0] [Proc,1] [1,0] [Atom,1] [2,0] [Atom,1] ["",0] [Atoms,1] [Atoms,2]
[Atoms,2] [),0] [List,4] [Atom,1] ["",0] [Atoms,1] [Atoms,2] [S,1]
```

Note: ϵ is represented by the empty string `""`. A test solution (in the form of a Java class file: `SchemeParser.class`) is provided for you to test your parser. The sample solution takes about 100 lines of Java code.

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 the sample solution looks like:

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

# Command to run your program
java SchemeParser
```

To submit this part of the assignment please use SVN as well as in hard-copy. On the hard-copy please note the login id of the person who submitted the programming portion of the assignment for the group. Please see the Resource or Assignment page of the course website for how to use SVN. Remember, you need to include a script called `runme.sh` that will let the marker run your code.

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

$$L = \{\sigma \in \{a, b, c, d\}^* \mid |\sigma|_a + |\sigma|_b = |\sigma|_c + |\sigma|_d\}$$

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

3. [**10 marks**] As mentioned in class, PDAs can accept in two different ways: empty stack or final state. Show that these two approaches are equivalent, i.e., that given a PDA that accepts with an empty stack, we can construct a PDA that accepts with a final state and vice versa.

CSCI3136: Assignment 5

Winter 2014

Student Name	Login ID	Student Number	Student Signature

	Mark	
Question 1		/30
Functionality	/15	
Structure	/15	
Question 2		/10
Question 3		/10
Total		**/50**

Comments:

Assignments are due by 3:00pm on the due date before class and must include this cover page. Assignment *must* be submitted into the assignment boxes on the second floor of the Goldberg CS Building (by the elevators).

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.