

## CS 405/505 ASSIGNMENT #2

Due Wednesday, January 31, 2007

Consider the following extended BNF grammar for a subset of the Perl programming language, called MicroPerl.

program	::=	<b>use strict ;</b> <b>require “list.pl” ;</b> <u>{ sub-definition }</u> <u>{ { declaration ; } { statement } }</u>
sub-definition	::=	<b>sub</b> sub-identifier <u>{</u> <b>my</b> variable-identifier = @_ ; <u>{ declaration ; } { statement }</u> <b>return</b> ( expression ) ; <u>}</u>
declaration	::=	<b>my</b> variable-identifier
statement	::=	variable-identifier = expression ;   <b>if</b> ( expression ) statement-block [ <b>else</b> statement-block ]   <b>while</b> ( expression ) statement-block   <b>print</b> variable-identifier ;
statement-block	::=	<u>{ statement { statement } }</u>
expression	::=	term   expression binary-op expression
term	::=	primary-expression   unary-op term
primary-expression	::=	variable-identifier   integer   list-expression   sub-identifier list-expression   <b>hd</b> list-expression   <b>tl</b> list-expression   <b>isNil</b> list-expression
list-expression	::=	( [ expression { , expression } ] )

**Syntactic and Semantic Conventions** The keywords and the token symbols in MicroPerl are in bold. Note that MicroPerl has symbols { and }, which are distinguished from grammar metasymbols { and }, respectively, by underlining. The unary operators are +, - and ! (negation). Binary operators obey the customary precedence rules, from highest to lowest:

multiplicative	*, /
binary additive	+, -
relational	==, !=, <, >, <=, >=
boolean	&&,

A MicroPerl sub identifier is a bare name, which consists of letters (only alphabetic characters), digits, and underscores (\_) with the restrictions that it must begin with a letter, cannot end with an underscore and cannot have two consecutive underscores. For example, give\_2\_Joe, tell\_me and A45Asm3 are valid bare names, but 6gh, two\_bad, and no\_end\_ are not. A MicroPerl variable may be a scalar variable or a list variable. A scalar variable identifier begins with a \$, followed by a bare name. A list variable identifier begins with a @, followed by a bare name. integer is an unsigned integer. MicroPerl comments are indicated by being preceded by # and terminated by the end of the line.

1. Determine the set of tokens which a lexical analyzer would need to recognize.
2. Design and implement a lexical analyzer procedure to read a source program in the above language and print the next token in the input stream. If the token detected is a valueless token, such as a keyword, then it is sufficient to print only the keyword. If it has a value, then both the token type and lexeme should be printed.
3. You will be given several MicroPerl programs with which to test your lexical analyzer. These will be located on the class WWW page and will be of the form test1.pl, test2.pl, etc.

**Suggestions:**

1. Construct a token class to represent the token data structure, including a method to print a token.
2. Use the JLex tool to automatically construct a lexical analyzer in Java from a set of regular expressions specifying tokens. This can interface with your token class to return a token to the main program which then prints the token.