Ruby Project
401-Programming Languages

1 Ruby Installation

- Windows: Go to `http://rubyinstaller.org/`, and follow the instructions. For a quick and easy install, click the <Download> button at the top of the page, and choose desired version under <RubyInstallers>
- OS X: preinstalled
- Linux/Unix
  - Install from package manager
  - Go to `www.ruby-lang.org/en/downloads/`, and follow the instructions.
- CIS: installed on vulcans

2 Assignment

Consider the following BNF for the Sim Programming Language (SimPL).

```
<program>   ::= <stmts>
<stmts>     ::= <stmt> ';'
              | <stmt> ';' <stmts>
<stmt>      ::= identifier ':=' <addop>
              | 'if' <lexpr> 'then' <stmts> 'end'
              | 'if' <lexpr> 'then' <stmts> 'else' <stmts> 'end'
              | 'for' identifier 'from' <addop> 'to' <addop> 'do' <stmts> 'end'
              | 'for' identifier 'from' <addop> 'to' <addop> 'by' <addop> 'do' <stmts> 'end'
<addop>     ::= <mulop> '+' <addop>
              | <mulop> '-' <addop>
              | <mulop>
<mulop>     ::= <factor> '*' <mulop>
              | <factor> '/' <mulop>
              | <factor>
<factor>    ::= integer
              | identifier
              | '(' <addop> ')'
<lexpr>     ::= <lterm> 'and' <lexpr>
              | <lterm>
<lterm>     ::= 'not' <lfactor>
              | <lfactor>
<lfactor>   ::= 'true'
              | 'false'
              | <relop>
<relop>     ::= <addop> '<=' <addop>
              | <addop> '<' <addop>
              | <addop> '=' <addop>
```

- Write a lexer for the program language definition. The lexer should have a `getTokenKind()` function that, when called, returns the symbol for the next sequential token (this function will be utilized in the next section). A symbol is a numeric value indicating the kind of token that was parsed. (e.g., one value for each keyword, operator, one value for all identifiers, one value for all integers, and one value for EOF.) In addition, the lexer should have a function `getTokenText` that returns the textual representation of the token. `nextToken` consumes the current token. The Lexer also needs to tack an 'EOF' (end of file) token to the end of the list of tokens (this will also be used later).
  - separating by whitespace alone will not suffice as there may be instances such as "x:=5", which consists of three tokens, but will only be counted as one.
  - A SimPL identifier consists of letters (only alphabetic characters), digits, and underscores (_) with the restrictions that it must begin with a letter. SimPL comments are indicated by being preceded by (//) and terminated by the end of the line.
- Create a recursive-descent syntax analyzer for the above language. Use the `getTokenKind()` and `getTokenText()` methods from the previous section to retrieve the tokens at each step. If the 'EOF' token is reach with no issue, the program should report that the file is syntactically correct. Otherwise, if an error is found, an appropriate error message should be displayed.

Assignment report: Turn in the source code of your project together with an assignment report. The assignment report should contain: (1) names of two team members including a short statement of work, (2) a description of your design, (3) what was difficult, (4) what did you like about Ruby, (5) what did you dislike about Ruby.

The following code should be a syntacticly correct SimPL program.

```
prev := 0;
curr := 1;

for iter from 0 to N-1 do // iterative fibonacci
  tmp := prev + curr;
  prev := curr;
  curr := tmp;
end
;
```

The following is not a valid SimPL program. It contains a syntax error in every line.

```
prev := 0
curr := -1;
/ not a comment
for iter from 1 by 2 do
  tmp := prev (+ curr);
  prev = curr;
  curr := ;
end
```