Building a Lexical Analyser with Ruby

The grammar rules for the language “TINY” are listed below. In this assignment we will identify the tokens in this language and build a lexical analyser (lexer) for recognizing and outputting TINY tokens.

# [https://www.cs.rochester.edu/~brown/173/readings/05\_grammars.txt](http://www.cs.rochester.edu/%7Ebrown/173/readings/05_grammars.txt)

#

# "TINY" Grammar

#

# PGM --> STMT+

# STMT --> ASSIGN | "print" EXP

# ASSIGN --> ID "=" EXP

# EXP --> TERM ETAIL

# ETAIL --> "+" TERM ETAIL | "-" TERM ETAIL | EPSILON

# TERM --> FACTOR TTAIL

# TTAIL --> "\*" FACTOR TTAIL | "/" FACTOR TTAIL | EPSILON

# FACTOR --> "(" EXP ")" | INT | ID

#

# ID --> ALPHA+

# ALPHA --> a | b | … | z or

# A | B | … | Z

# INT --> DIGIT+

# DIGIT --> 0 | 1 | … | 9

# WHITESPACE --> Ruby Whitespace

Whenever a token is identified, we encapsulate it using the Token class below. Each token has a type and text. For example, if DOG was identified as a variable in this language, it could have type “id” and

text “DOG”. The add operator might have type “addOp” and text “+” or type “+” and text “+”. These values are somewhat arbitrary – choose values that make sense to you.

## Token Class

I have given you a partially complete Token class that you should modify for this assignment called “**TinyToken.rb**”. It has a few constants already defined. You will need to build constants for all the tokens.

It is important to test all the code you write. At the very least, use “puts” to test the class methods. You can build the tests in separate files and load them as needed: load “mytest.rb”. The code below tests the Token class methods.

# Test the Token class

tok = Token.new("atype","atext") puts "Token type: #{tok.type}" puts "Token text: #{tok.text}" tok.type = "btype"

tok.text = "btext"

puts "Token type: #{tok.type}" puts "Token text: #{tok.text}" puts "Token: #{tok}"

## The “Lexer”

The first goal is to build a Scanner (or Lexer) for TINY. I have sketched the basic structure in file named “**TinyScanner.rb**”. Here are a few points to consider:

1. The constructor is passed a file name which contains the source code for a TINY program. The constructor opens the file and reads the first character, storing it in class variable @c (which acts as a one-character look ahead).
2. Currently, the Scanner abends if it is passed a file that doesn’t exist. Modify the code so that it fails gracefully in this circumstance.
3. Method nextCh() updates @c with the next character and returns it, unless it has reached the end of the file, in which case it will return “**!eof!**”.
4. Method nextToken() returns the next token identified by the scanner. It is not complete, as it does not identify all Tokens in your grammar yet. You should complete this method.
5. Contiguous whitespace should be combined and emitted as a single token. (I already did this for you).
6. An end of file (EOF) token should be emitted when the file has been completely processed. (I already did this for you).
7. There are several helper methods defined at the bottom of “TinyScanner.rb” that you can use to help you to identify different types of characters (like numbers, letters, and whitespace).

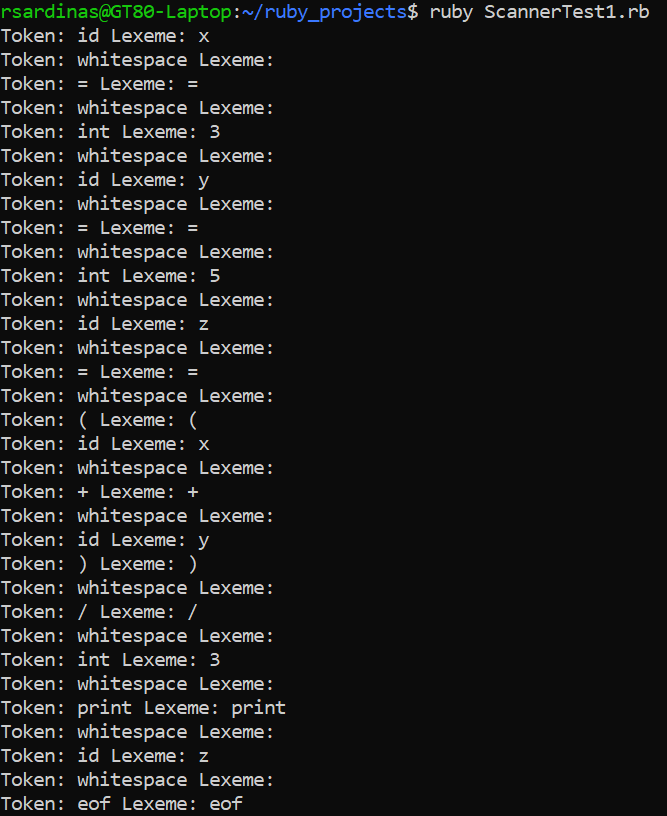
## Display and Print Tokens

I’ve included a 3rd file called “**TestTinyLexer.rb**” that will use your lexer (previous section) and display each token it encounters as it lexes your source code file. I would like for you to modify this file so that in addition to displaying the tokens to the console, it will also write them to a file (that we could potentially pass to a parser).

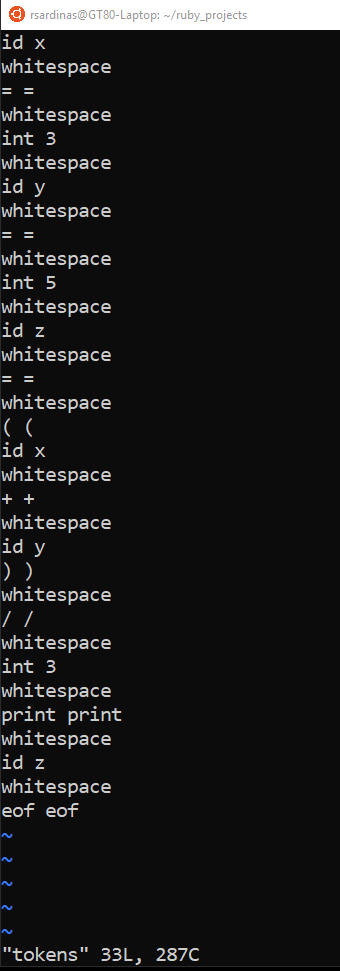
I’ve also included a sample input file (input.txt)that adheres to our TINY programming language. You should experiment with different inputs that both adhere to and don’t adhere to the grammar to verify the correctness of your lexer.

## Sample Program Output

Below is a screenshot of sample output that would result from using the included sample input file.



Below is a screenshot of the token file that was generated as a result of lexing the same input file (TINY source code file).



## Deliverables (THIS IS WHAT YOU SHOULD TURN IN)

1. TinyToken.rb
2. TinyScanner.rb
3. TestTinyLexer.rb