CS 3520 - Programming Language Structures

Program: 3
Points: 50
Due Date: **November 15, 11:59 pm**

You are going to implement a lexical analyzer in Ruby for the Dot language. Please refer to *dotguide.pdf* for more information about Dot. The Lexer reads in an input stream of characters and recognizes tokens in that stream.

Notes:

1. You need to create two ruby files: **dot_lexer.rb** where you have all the token recognition logic and **token.rb** where you have a Token class that uses **text** and **type** to describe a token. In class DotLexer, you must implement the method **next_token** that returns recognized tokens. **prog3.rb** is provided.
2. All supported token types are listed below. All other tokens are considered as `Token.INVALID`**.** There is another token **EOF** that should also be defined in class Token. Please refer to the sample output for "Ca#t" in the sample input to get an idea how to deal with an illegal character.
3. The name of a node can be an ID, an INT, or a STRING. An ID starts with a letter followed by letters or digits. An INT contains only digits. A STRING is recognized by double quotes within which all characters are valid.
4. There are only two keywords: `digraph` (or `DIGRAPH`) and `subgraph` (or `SUBGRAPH`).
5. For whitespaces, such as ' ', '\r', '\t', and '\n', recognize and skip them without printing out.
6. Comment your code whenever necessary.
7. This program MUST be done individually. Overly similar programs will result in 0 for all involved.
8. Fully test your program before submission.

All valid token types:

```
ID = 1
INT = 2
STRING = 3
LCURLY = 4
RCURLY = 5
SEMI = 6
LBRACK = 7
RBRACK = 8
ARROW = 9
EQUALS = 10
DIGRAPH = 11
SUBGRAPH = 12
COMMA = 13
WS = 14
```

Sample Input:

```
digraph trees {
  subgraph t {
    0 -> "1" [label = "A"];
    0 -> "2" [label = "B"];
  }
  SUBGRAPH u {
    Animal -> Ca#t [label = "feline"];
    Animal -> Dog1 [label = "canine"];
  }
}
```

Sample Output:

```
[digraph:11]
[trees:1]
[{:4]
[subgraph:12]
[t:1]
[{:4]
[0:2]
[->:9]
["1":3]
[[:7]
[label:1]
[=:10]
["A":3]
[]:8]
[;:6]
[0:2]
[->:9]
["2":3]
[[:7]
[label:1]
[=:10]
["B":3]
[]:8]
[;:6]
[}:5]
[SUBGRAPH:12]
[u:1]
[{:4]
[Animal:1]
[->:9]
[Ca:1]
illegal char: #
[t:1]
[[:7]
[label:1]
[=:10]
["feline":3]
[]:8]
[;:6]
[Animal:1]
[->:9]
[Dog1:1]
[[:7]
[label:1]
[=:10]
["canine":3]
[]:8]
[;:6]
[}:5]
Lexical analysis is finished!
```