Author: Nghia Duc Hong

Student Number: 292119

Email: duc.hong@tuni.fi

1. Lexical analysis is the first stage in the process that compiler uses to understand the input of the program. In this phase, the lexical analyzer splits the source code into small parts call tokens. By defining the lexical grammar, the rules are defined as lexical specification. By this lexical specification, the compiler can realize and scan what type of tokens and their order in the source code. If there are some characters/words/symbols are not defined in the rules, then the lexical analyzer would raise an error that the sequence is invalid or unexpected.
2. There are three parts in making a lexical analyzer. The first part is the declaration list of the tokens. In this part, the names of the tokens are defined by a string. The second part makes definition for each token, which is expressed by regular expression to capture the tokens which is predetermined by the language. Finally, the analyzer is compiled to be used.

3. How tokens are recognized:

- Keywords: keywords are firstly recognized, the form of the keywords only overlaps the form of IDENT (identifier) then if it is not a keyword, then it is an identifier.
- Comments: comments are eliminated by using re.sub() before recognizing any tokens.
- Whitespaces between tokens: are skipped.
- Operator & delimiters: are recognized as the short letter tokens.
- Decimal literal: contains a sequence of digits followed by a dot and a number.
- String literal: between the exclamation mark (!).
- Function name: A uppercase letter followed by numbers or lowercase letters or underscore.

4. How can lexer distinguish between:

- Function names and variables name: Function name has first letter uppercase and variables name does not.
- Keywords and variable names: keywords are pioritised to be analyzed first, if it is not, then it is a variables name (only keywords and variable names can start with a lowercase char).
- Operator > and =>: The string containing => has length of 2 and string containing > has length of 1 so it is automatically to recognize => before >.
- Comment and other code: The pair of “...” will nest a comment. Then anything between these pairs are skipped.
- Integer and decimals: the decimals are analyzed before integers as the decimals has a dot and integers do not.

5. Extra feature:

- Comments can be performed in multiple line
- Remove comments from inside string literals.
- Remove exclamation mark in the literal string tokens.
- Try to convert decimal tokens have value of float.

6. Reflection:

The assignment was not hard but required to read a lot. The documentation of the PLY is quite hard to follow in my opinion. The source on Internet is not much so it is hard to fix a bug and time consuming to find an appropriate way to code, for example, the function re.sub() is not mentioned in anywhere to be used and it took time looking for it. There was a struggling part is trying to prioritised how tokens are ordered that the lexical analyzer did anything stupid also. This require number of tests and check which is laborious. Overall, the assignment gave me to an overview how to make lexical analyzer in the first stage of writing a compiler/interpreter.