COMS W4115
Programming Languages and Translators

Programming Assignment 1

Due Date: 17th October 2024, 11:59 PM EST

Background

The working of a scanner—sometimes called the tokenizer or lexical analyzer—forms the first phase of the compilation process. A scanner is a program that takes a sequence of characters (the program's source file) and produces a sequence of tokens that will be used to feed the compiler's parser, the next phase of the compilation process. So, for example, the input
A := B + 4
This would translate into the following tokens:

IDENTIFIER (Value = "A")
OPERATOR (Value = ":=")
IDENTIFIER (Value = "B")
OPERATOR (Value = "+")
INTLITERAL (Value = "4")

We define tokens in a programming language using regular expressions.

For example, a regular expression that defines an integer literal token looks like
[0-9]+ (read: "1 or more digits")

Deliverables

You will create a scanner that processes an input source code written in the programming language you designed and outputs a list of tokens. The program should show state transitions with a finite automate

1. **Define the lexical grammar** for your language, ensuring each token and its rules are documented in a README file. Design the lexical grammar that has at least **5** different token types.
2. **Develop a scanning algorithm** to

    a. Scan the input programs written in your language and output a list of tokens in the following format: `<Token Type, Token Value>.`
    b. The scanner should show state transitions with finite automata, as discussed in the class, rather than calling some library routine (e.g., regular expression library). Please look at the class lecture where the instructor describes what an acceptable program will look like for this purpose.
    c. Handle lexical errors present in the input program
3. Provide **5 sample input programs** with their expected token outputs for testing. Write the input programs such that you can
    a. demonstrate different types of token parsing
    b. Error handling capabilities of your parser
    c. The input program should do something more meaningful than some random input string.
4. **Write a shell script** to execute your lexer. The script should either include the installation steps or provide them separately in the README.
5. Provide a detailed description of each step in the README.
6. **Add names of the teammates and UNI in the README**. 0 points will be awarded if it is absent.

It is fine if one of the teammates submits. As long as deliverable 6 is done, we will credit all the teammates.

Submission

Submit the URL to your GitHub repository on Courseworks. The repository should be public for the TAs to access and grade. If the TAs are unable to access the repository, you will receive a ‘0’ grade for the assessment. The repository will be monitored to see the contribution from each team member, so it might be a good idea to create Pull Requests on the repository for the work that each teammate is doing on the assessment.

Late Submission Policy:

There is 0 tolerance for late submissions. You will instantly get a 0 for any late submission/no submissions. For any assignment going forward, we will do a code pull for the deadline and will not accept any code submitted after the grading deadline.

For any issues or doubts around the project, contact the TAs during their office hours as listed here. If the doubt can be answered publicly and would benefit everyone in the class, please prefer posting on EdStem.