Milestone #1 Self Evaluation

Name: Cole Cathcart
UCID: 30043556
Email: cole.cathcart1@ucalgary.ca
Gitlab repo: https://gitlab.cpsc.ucalgary.ca/cole.cathcart1/cpsc411-term-project.git

One area I'd like feedback from the TA:

- I created a simple Token class to represent tokens. I am unsure if this was a necessary/good idea as I could have also defined them as a struct. I would like to know:
    - Should this implementation be changed to a struct?
    - Would it be better for the 'type' parameter to be changed to an enum or other type? It is currently a string

Self Assessment: **7/8**. I will break this mark down using the dimensions as outlined in the milestone 1 specification:

1.) Tool (milestone) properties: **2/2**
    - Error/Warning message output: I believe my errors and warnings are descriptive and accurate, and they provide the appropriate line number.
    - Predictable: I believe my tokenization works properly for all inputs, rejects erroneous inputs, and never crashes
    - Evidence: Please see the 'run.output' file located in my gitlab repository for errors, warnings, tokenization. Please see the README.md file (on the repo) for instructions on how to run the code to verify there are no crashes
2.) Development practices: **1.5/2**
    - Commiting regularly to the repo: I believe I did this well for the most part, although on at least 1 occasion I probably waited too long and in-between commits and had commits that were too large as a result
    - Good commit messages: I believe all my commit messages are descriptive and accurate
    - Evidence: Please see my repo's commit history
3.) Code qualities: **1.5/2**
    - My code is fast and easy to build and run
    - I believe my code is very well documented and maintains style consistency
    - I believe my code follows good coding practices and standard c++ conventions
    - One area my code could be improved: Although most of my implementation is properly object-oriented and modular, my scanner's 'lex()' function is quite large and should probably be broken into sub-functions for better readability and modularity
    - Evidence: Please see the *.cpp and *.hpp files for evidence of documentation and good coding practices. Please see the 'lex()' function in scanner.cpp for evidence of where my code could be improved. For a fresh build output please see the beginning of the 'run.output' file in my repo

4.) Relationship with the environment: **2/2**

- Errors and warnings printed to stderr, regular output printed to stdout, exit codes set to non-zero numbers for error exits
- No extraneous garbage in outputs
- I believe my code is easy to run, and my README.md is well documented with clear instructions
- Evidence: Please see the outputs in the ‘run.output’ file in my repo, and the ‘README.md’ file in my repo