**CS 4280, Section 001, Academic Integrity Statement**

For this project, AI resources can be freely used, as long as they are fully disclosed as described herein. Additional non-AI internet resources can be utilized, as long as they are fully disclosed. Furthermore, code written by UMSL students in ***previous*** semesters can be looked at, but never copied.

**IMPORTANT**: Clearly indicate all outside resources utilized and sign below. Failure to cite the use of outside resources will be reported for appropriate disciplinary actions. Note that discussions with other students are encouraged; looking at each other’s code and/or copying – with or without modifications – are unacceptable and will be reported.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

I hereby certify that all outside resources utilized, other than suggested textbooks and class materials, are clearly noted in my source code and in the following. The start and finish lines of the affected code are shown using ‘start ORx’ and ‘end ORx’, respectively, where ‘x’ is a unique number. Each value of ‘x’ has a corresponding explanation in this Academic Integrity Statement.

All internet resources include the web address and the date accessed. For each usage of AI, I also include the prompt and code submitted and the output returned or a link to the interaction is included herein.

All other materials I provide for this project submission are my own original work. I hereby certify that I am responsible for each and every line of code that I submit in my source files and I thoroughly understand how the code works to produce the output. I understand that I wll be required to meet with the instructor to answer detailed questions about my submitted code.

OR1:

<https://www.programiz.com/cpp-programming/library-function/cctype/isalpha>

* Looking for letter -> token 3 validation
* if (!isalpha(str[0]))
* return false;

OR2:

* <https://docs.vultr.com/cpp/standard-library/cctype/isdigit>
* Looking for digit -> token3 validation
* if (!isdigit(str[i])) *// checking if the rest are digits*
* return false;

OR3:

* https://cplusplus.com/reference/string/string/
* *// function to remove comments*

string removeComments(string str)

{

size\_t startOfComment = str.find('\*');

while (startOfComment != string::npos) *// if another '\*' is found*

{

size\_t endOfComment = str.find('\*', startOfComment + 1);

if (endOfComment == string::npos) *// there is no other '\*'*

break;

return str.substr(0, startOfComment) + str.substr(endOfComment + 1); *// removing the comment and everything in-between*

}

return str;

}

OR4:

* <https://www.w3schools.com/cpp/cpp_enum.asp>
* Use enums to access the type of token
* */\**
* *Author: Grant Hughes*
* *Created: February 23, 2025*
* *token.hpp:*
* *- header file for token*
* *\*/*
* #ifndef \_TOKEN\_HPP\_
* #define \_TOKEN\_HPP\_
* #include <string>
* enum TokenID {
* t1\_tk,
* t2\_tk,
* t3\_tk,
* EOFTk
* };
* struct Token {
* TokenID tokenID;
* std::string tokenInstance;
* int lineNumber;
* Token(TokenID id, const std::string &instance, int line): tokenID(id), tokenInstance(instance), lineNumber(line) {}
* *// Method to get the string representation of the token type instead of int value*
* std::string getTokenAsString() const {
* switch (tokenID) {
* case t1\_tk: return "t1\_tk";
* case t2\_tk: return "t2\_tk";
* case t3\_tk: return "t3\_tk";
* case EOFTk: return "EOFTk";
* default: return "tk\_unkown";
* }
* }
* };
* #endif

*Project #*: **\_\_P1\_\_**

Grant Hughes February 25, 2025

*Signature (typed name is fine) Date*