**CS 4280, Project P4, 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:

* Used AI to convert to C to C++ (will check/test it before using it)
  + Just letting you know

// inline function for termination:

inline void fatal(const char\* string) {fprintf(stderr,"\nFatal: %s\n\n",string); exit(1); }

// prototypes:

char\* t2\_2\_pID (char\* t2);

int t3\_2\_intVal(char\* t2ID);

// function definitions:

char\* t2\_2\_pID (char\* t2) {

if(t2[0] != '+')

fatal("Expected a plus sign for the first character of a t2 token.");

t2[0] = 'p';

return t2;

}

int t3\_2\_intVal(char\* t3) {

int intVal; // will hold the integer value of the t3 token

int caseVal; // set to +1 if upper-case (positive value) and set to -1 if lower-case (negative value)

if((t3[0] < 65) || (t3[0] > 122)) // checking whole range from A to Z (ascii 65 - 122)

fatal("Expected a letter for the first character of a t3 token.");

if((t3[0] > 90) && (t3[0] < 97)) // checking for ascii non-alphabetical characters between Z and a (ascii 91 - 96)

fatal("Expected a letter for the first character of a t3 token.");

if((t3[0] > 64) && (t3[0] < 91)) // upper-case letter, so integer will be positive

caseVal = 1;

if((t3[0] > 96) && (t3[0] < 123)) // lower-case letter, so integer will be negative

caseVal = -1;

t3[0] = '0'; // set to zero so that it will go away when convert string to integer

intVal = atoi(t3); // convert to integer

intVal = intVal \* caseVal; // set to negative if needed

return intVal;

}

#include <iostream>

#include <cstdlib>

#include <string>

using namespace std;

// inline function for termination:

inline void fatal(const string& message) {

cerr << "\nFatal: " << message << "\n\n";

exit(1);

}

// prototypes:

string t2\_2\_pID(string t2);

int t3\_2\_intVal(const string& t3);

// function definitions:

string t2\_2\_pID(string t2) {

if(t2[0] != '+')

fatal("Expected a plus sign for the first character of a t2 token.");

t2[0] = 'p';

return t2;

}

int t3\_2\_intVal(const string& t3) {

int intVal; // will hold the integer value of the t3 token

int caseVal; // set to +1 if upper-case (positive value) and set to -1 if lower-case (negative value)

if((t3[0] < 65) || (t3[0] > 122)) // checking whole range from A to Z (ascii 65 - 122)

fatal("Expected a letter for the first character of a t3 token.");

if((t3[0] > 90) && (t3[0] < 97)) // checking for ascii non-alphabetical characters between Z and a (ascii 91 - 96)

fatal("Expected a letter for the first character of a t3 token.");

if((t3[0] > 64) && (t3[0] < 91)) // upper-case letter, so integer will be positive

caseVal = 1;

if((t3[0] > 96) && (t3[0] < 123)) // lower-case letter, so integer will be negative

caseVal = -1;

// Create a copy of t3 with first character replaced by '0'

string t3Copy = t3;

t3Copy[0] = '0';

intVal = stoi(t3Copy); // convert to integer

intVal = intVal \* caseVal; // set to negative if needed

return intVal;

}

OR2:

-protypes

<https://cplusplus.com/articles/yAqpX9L8/>

*// prototypes*

void processEachNodeInTree(node\* root);

OR3:

Copilot fix on vscode

"ASMCode::ASMCode(std::\_\_1::string file, SymbolTable &symbolTable)" provides no initializer for:C/C++(366)

This

table = symbolTable;

countForTempVariables = 0;

to

: table(symbolTable), countForTempVariables(0), output(file)

The part after the colon (:) is called an **initializer list**. It initializes class member variables before the constructor body executes.

OR3:

*Project #*: **\_\_P4\_\_**

Grant Hughes 5/10/2025

*Signature (typed name is fine) Date*