**Team Project 2**

Project 1A – Infix Expression Parser

**Bryce Thornton**

**Robert Blocker**

**november 13rd, 2022**

**Table of Contents**

[1.](#_gjdgxs) System Design 2

[2.](#_30j0zll) UML Diagram 2

[3.](#_1fob9te) Test Cases 2

[4.](#_2et92p0) Team Member Contribution 2

[5.](#_tyjcwt) Possible Improvements 2

# System Design

The Infix\_To\_Postfix files have the Boolean is\_operator(), precedence() returning an int, and the infix\_to\_postfix that returns the string of the converted infix expression. Is\_operator() was used to return if the char was an operator. Precedence was created to manually change the precedence of the operators, returning the order in which the operator was to be evaluated in the expression. The infix\_to\_postfix() converted the infix expression string to a postfix string. There also is a header file to attach to the main function.

The Postfix\_Eval files calculated a for a single operator with the calc() function and evaluated the previously converted postfix expression and returned evaluation.

The main function opens an input and output file, reads from an input file and calls the infix\_to\_postfix and evaluate functions from the two previously mentioned files to write the evaluation into the output file.

The system was based on using stacks to push operators and operands onto a stack and pop the top operator and operand to append to our strings.

.

# UML Diagram

Diagram

Description automatically generated

# Test Cases

Text

Description automatically generatedText

Description automatically generatedGraphical user interface, text, application

Description automatically generated

For our test cases, we wanted to display the infix expressions to the console to show that our tests work. Then we can see our input file filled with the infix expressions and we can see the output file that has the evaluated expressions.

For each expression we received our expected output at the same line of the evaluated expression.

# Team Member Contribution

Bryce Thornton – Bryce was responsible for code development for the project. Through logic and testing, Bryce helped make sure his code functioned properly. Bryce was responsible for the header files and development of the main function.

Robert Blocker – Robert was responsible for code development and documentation of the system. Through logic and testing, Bryce helped make sure his code functioned properly. Robert developed the Infix\_To\_Postfix and Postfix\_Eval cpp files.

# Possible Improvements

* The design of the system could be improved by trying to develop a system that was more conscious of the time complexities.
* More tests could have been run to ensure any possible bugs were fixed.
* The code could have been simplified to ensure a more efficient program.