**Homework Six Project Report**

In this project, I learned how to implement stacks and vectors to create an evaluator of expressions as well as the reformatting of expression notation. In HW6, we built upon our HW3 code, creating a method for postfix notation which reformats infix input notation to make the expression easier to manipulate arithmetically in code. Postfix notation uses a stack to store the user input expressions in a specific order and push the expression into a vector of tokens named postfix. Postfix is then used for prefix notation, fully parenthesizing, and arithmetic evaluation of integers and identifiers since 2 integers/identifiers followed by an operator defines an operation in postfix. In this assignment, I also learned to make a functioning user input menu and error (syntax) checking for the specific input possibilities such as a divide by 0 error and undefined variable error.

A key portion of this assignment was learning how to differ the types and validity of inputs since some could be arithmetic, assignment, or illegal. For my program, I created a set\_type method that defines whether the type of the input is arithmetic, assignment, or illegal dependent on many conditional statements. This definition is used to determine whether an expression is valid or not so that we can call other methods such as postfix, prefix, fully parenthesize, and evaluate without running into errors. Some future edits we could make to the code is testing bound errors as well as improving the menu to make user input and testing simpler.

Test cases are extremely important for finding program functionality and diagnosing errors; thus, this homework assignment contained many user inputs to cover a wide range of possible outputs for the varying actions and error handling.

**My Test Cases:**

=== expression evaluation program starts ===

input:(7-7 \*5+1)\*(3-2+5\*6)/2;

action:=

(7-7\*5+1)\*(3-2+5\*6)/2 = -418

action:<

postfix of (7-7\*5+1)\*(3-2+5\*6)/2 is: 7 7 5 \* - 1 + 3 2 - 5 6 \* + \* 2 /

action:>

prefix of (7-7\*5+1)\*(3-2+5\*6)/2 is: / \* + - 7 \* 7 5 1 + - 3 2 \* 5 6 2

action:f

fully parenthesizing (7-7\*5+1)\*(3-2+5\*6)/2 is: ((((7-(7\*5))+1)\*((3-2)+(5\*6)))/2)

action:s

input:69/0;

action:=

69/0 = no result, divide by 0 error!

action:a+b;

Wrong input for the action! Please type one of =, <, >, f(F), q(Q), c(C), s(S)

action:=

69/0 = no result, divide by 0 error!

action:s

input:a+b;a=4;b=9;

action:=

a+b = 13

cannot evaluate a=4 which is not an arithmetic expression, but assignment.

cannot evaluate b=9 which is not an arithmetic expression, but assignment.

action:c

input:b=100;

action:=

a+b = 104

cannot evaluate a=4 which is not an arithmetic expression, but assignment.

cannot evaluate b=9 which is not an arithmetic expression, but assignment.

cannot evaluate b=100 which is not an arithmetic expression, but assignment.

action:s

input:7++4;

action:=

cannot evaluate 7++4 which is an invalid expression.

action:s

input:((3);

action:=

cannot evaluate ((3) which is an invalid expression.

action:s

input:)2+2(;

action:=

cannot evaluate )2+2( which is an invalid expression.

action:q

=== expression evaluation program ends ===

**Given Test Cases:**

