CISC2200 Fall 2018
Final Project 15 points Due Date: December 14, 2018

In this project, you will implement a calculator which could do calculation.

The input of this calculator should be in the format of infix notation:

$$(6 + 13) * 2 – (5 + 1)/3$$

To evaluate an infix expression, the calculator first converts the expression to postfix notation, and then evaluate it. The format of postfix notation of the above input is:

$$6\ 13\ +\ 2\ *\ 5\ \ 1\ +\ 3\ /\ -$$

Only four arithmetic operations are allowed in an expression: +, -, *, /.

The program should read the infix input from the standard input, print the expression in the postfix format on the screen, and then print the evaluated value of the expression.

You should check errors of the input string in case it is not syntactically and semantically correct. In other words, there may be input such as "1+2*", "(1+2))" or "1/0". You should create your own **Exception classes** to handle these errors.

Algorithm to convert from infix to postfix (with a stack)

1. Examine the next element in the input.
2. If it is an operand, output it.
3. If it is opening parenthesis, push it on stack.
4. If it is an operator, then
    a. If stack is empty, push operator on stack.
    b. If the top of the stack is opening parenthesis, push operator on stack.
    c. If it has higher priority than the top of stack, push operator on stack.
    d. Else pop the operator from the stack and output it, repeat step 4.
5. If it is a closing parenthesis, pop operators from the stack and output them until an opening parenthesis is encountered. Pop and discard the opening parenthesis.
6. If there is more input go to step 1
7. If there is no more input, unstack the remaining operators to output.

Algorithm to evaluate postfix expression (with a stack)

1. Scan the given postfix expression from left to right
2. for each token in the input postfix expression
    a. if the token is an operand
        i. push it (its value) onto a stack
    b. else if the token is an operator
        i. get operand2 by popping up the stack
        ii. get operand1 by popping up the stack
        iii. compute operand1 operator operand2
        iv. push result onto the stack
3. return top of the stack as result

Please use Stack template class from C++ STL to finish this project.

Submit: finalProject.cpp and any other class files.