Stack-based Postfix Expression Calculator

Due Tuesday by 11:59pm **Points** 70 **Submitting** an external tool

An arithmetic expresion written in postfix notation can be evaluated by pushing each operand onto a stack, popping these operations whenever an operation is encountered with the result pushed back on top of the stack. In this assignment a function that calculates a postfix expression with integer operands using a stack will be implemented. The expression is represented using an array of strings with each string being either an operand or operator symbol.

Download the starter code and implement the **calculatePostfixExpression()** function in the **calculatePostfixExpression.cpp** file which will be modified and must be submitted with the same filename. This function takes two parameters: the first is an array of strings for the expression, each string being either an (integer) operand or a single operator symbol; the second is an int indicating the length of this expression (#operands + #operator symbols). The expression is evaluated upon reaching the end of the expression and after performing the last operation the result of the expression should be the only value on the stack which is then popped and returned from the function as an int value.

Operator symbols:
"+" addition
"-" subtraction
"*" multiplication
"/" divison
"%" remainder

Example expression: "8", "5", "-" evaluates to a result of 3:
1: push 8
2: push 5
3: "-" -> pop top two values, earlier value is the left-side operand: 8 - 5
The result of the operation (8-5) is 3, push 3 onto stack

After evaluation of the expression, the final result should be the only value on the stack, return 0 if the stack is non-empty after popping the result. Additionally if the expression is zero-length (no operators or operands) or if there are insufficient operands for an operation, return 0. The STL Stack class can be used to easily convert a numeric string to an int you may use the **stoi()** function which takes a string as a parameter and returns an int. Only the **calculatePostfixExpression()** function needs to be implemented for the test cases but it will be helpful to use a main() function to

test the function before turning in the assignment. Submit **calculatePostfixExpression.cpp** with the implemented function.

Stack-based Postfix Expression Calculator Starter Code.zip
(https://clemson.instructure.com/courses/198274/files/17721202?wrap=1) 
(https://clemson.instructure.com/courses/198274/files/17721202/download?download_frd=1)

Postfix Calculator Sample Test Cases.zip
(https://clemson.instructure.com/courses/198274/files/17721216?wrap=1) 
(https://clemson.instructure.com/courses/198274/files/17721216/download?download_frd=1) - Main functions for a subset of the test cases on Gradescope: T#.cpp is the main function for test case #, returns 0 when test case passes, returns 1 when test case fails.

This tool needs to be loaded in a new browser window

Load Stack-based Postfix Expression Calculator in a new window