Stacks

Problem Description

**Objective**

This practical will test your knowledge on stacks.

**Polish prefix**

Polish notation, also known as Polish prefix notation or simply prefix notation, is a form of notation for logic, arithmetic, and algebra. Its distinguishing feature is that it places operators to the left of their operands.

The expression for adding the numbers 1 and 2, in prefix notation, is written as “+ 1 2” rather than “1 + 2”. In more complex expressions, the operands may be nontrivial expressions including operators of their own. For instance, the expression that would be written in conventional infix notation as (5 + 6) \* 7 can be written in prefix as \* (+ 5 6) 7

In this assignment, we only care about binary operators: + - \* and /.

For binary operators, prefix representation is unambiguous and bracketing the prefix expression is unnecessary. As such, the previous expression can be further simplified to \* + 5 6 7

The processing of the product is deferred until its two operands are available (i.e., 5 plus 6, then multiplies the result with 7). As with any notation, the innermost expressions are evaluated first, but in prefix notation this ”innermost-ness” is conveyed by order rather than bracketing.

Your task is to create a function that convert a prefix expression to a standard form (otherwise known as infix) and compute the prefix expression.

Create a main function that takes in one line. The line contains a list of operators and operands separated by spaces. The input doesn’t contain parenthesis. An operator is a character from +, -, \*, and /. An operand is a nonnegative integer from 0 to 99. You are asked to convert the prefix expression to an infix form and output its value as well. If the expression is not a valid prefix expression, your program should output “Error”.

Sample input: \* - 5 6 7

Sample output: (5-6)\*7=-7

Sample input: \* 5

Sample output: Error

Sample input: \* 5 6 7

Sample output: Error

Sample input: 5

Sample output: 5=5

Sample input: \*

Sample output: Error

Marking Scheme (10 marks total)

* + Passing public test cases (5 mark)
  + Passing hidden test cases (5 mark)

Supplementary material:

算法介绍及注意事项：Algorithm for Prefix to Infix Conversion.doc

Sample code: PolishPrefixToInfix\_sample.cpp