2016 Fall CIS 200 – Lab 10

**Stacks**

In this lab you are going to write reverse postfix calculator using a stack.  **Postfix notation** and is parenthesis-free as long as operator [arities](http://en.wikipedia.org/wiki/Arity) are fixed.

In postfix notation the [operators](http://en.wikipedia.org/wiki/Operation_%28mathematics%29) follow their [operands](http://en.wikipedia.org/wiki/Operands); for instance, to add 3 and 4, one would write "3 4 +" rather than "3 + 4". If there are multiple operations, the operator is given immediately after its second operand; so the expression written "3 − 4 + 5" in conventional notation would be written "3 4 − 5 +" in postfix: first subtract 4 from 3, then add 5 to that. An advantage of postfix is that it eliminates the need for parentheses that are required by infix. While "3 − 4 \* 5" can also be written "3 − (4 \* 5)", that means something quite different from "(3 − 4) \* 5". In postfix, the former could be written "3 4 5 \* −", which unambiguously means "3 (4 5 \*) −" which reduces to "3 20 −"; the latter could be written "3 4 - 5 \*" (or 5 3 4 - \*, if you wish to keep similar formatting), which unambiguously means "(3 4 -) 5 \*".

|  |  |  |
| --- | --- | --- |
| **Standard expression** | **Postfix Notation** | **Value** |
| 1 + 3 | 1 3 + | 4 |
| 10 / 5 | 10 5 / | 2 |
| 10 / (6 + 2 -3) | 10 6 2 + 3 - / | 2 |

**The program logic**

read in a string

while the string is not “stop”

if the string is +, pop the last 2 values from the stack and push back their sum

else if the string is \*,pop the last 2 values from the stack and push back their

product

else if the string is -, pop the last 2 values from the stack and push back the second

– the first

else if the string is /, pop the last 2 values from the stack and push back the second

/ the first

else if string is =, print the top of the stack and pop the stack

else ( //the string is a number), convert to a double and push it on the stack;

read the next string

**Example Input/Output**

**type in a postFix express or stop to stop**

**1 3 + =**

**4.00000**

**10 5 / =**

**2.00000**

**10 6 2 + 3 - / =**

**2.00000**

**1.1 2.2 \* =**

**2.42000**

**stop**

**Hints:**

To include stack ADT from the standard template library

**#include <stack>**

To declare a stack of Type

**stack<Type> nameOfStack;**

The stack ADT supports , pop(), push( value), top().

To convert a string **word** into a double use

**double num = atof(word.c\_str());**

For pretty output include **<iomanip>**

And in the string put **fixed<<showpoint<<setprecision(5)**

Submit a well commented program in a word document.

Include some examples of input/output (commented out at the bottom)