MASSEY UNIVERSITY
TE KUNENGA KI PŪREHUROA
UNIVERSITY OF NEW ZEALAND

159.201 Algorithms & Data Structures
Summer 2014

Assignment 5

Write a C++ program that creates and populate ***a tree for an arithmetic expression***. Then it should perform an *in-order* and a *post-order* traversal on the tree. The input of the program will be a *text file* with the arithmetic expressions in RPN. To make it easier for the I/Os, each number has a single digit, so you can use `char` as data for the tree nodes. The RPN*.txt files should have one number or one operator per line.

The output of the program will be a printout of the arithmetic expression written out in both **in-fix** and **post-fix** (Reverse Polish) notations. The program does not need to compute the value of the expressions, *only print them out*. You should comment any other printouts (comment the printfs for "reading a number:..."). The output should look like this:

```
In-fix:
((1+5)*2)
Post-fix:
1 5 + 2 *
```

Notes: - The in-fix notation **must have all parenthesis** in the right places, i.e. the in-order traversal must be modified to achieve this. Note that there are no spaces between numbers and parenthesis. There is an additional pair of parenthesis that covers the entire expression (again, this will make it easier).

- The post-fix print out is useful for debugging purposes, as it should look like the original text file. There is a space after each character, including the last one.

- You need to use a **stack** that is able to store *Tree pointers*. You can modify the Stack class we showed during the lectures, or use STL.

Remember that you can use the following algorithm to build the tree out of a file with the post-fix notation. Assume the the RPN file is logically correct (i.e., no need to check "too many numbers/operators").

<u>An algorithm to construct an arithmetic tree</u>

Read in an expression that is ***already in post-fix notation*** (from the RPN*.txt files)

```
Tree *T1, *T2, *T;  Stack S;  //note that S is a stack of pointers to trees
while (expression continues) {
     x = next item from the expression
     if (x is a number) { S.Push(new Tree(x, NULL, NULL)); }
     if (x is an operator) {
        T1 = S.Top();  S.Pop();
        T2 = S.Top();  S.Pop();
        S.Push(new Tree(x, T2, T1));  //note order of T2 and T1
     }
   }
T = S.Top();
```

Use our virtual machine to mark your submissions (host name **vm000296**). The input/output requirements are essential, please follow them carefully to avoid losing marks. Spaces matter and text is case sensitive.

After you are satisfied with the performance of your code as tested in the virtual machine, submit a **one source file** code on Stream by **Friday 16th of January 2015.** Your **name** and **ID number** must appear **on top of the file as comments.** If you are working in a group, then ***all*** names and IDs must appear on top of the file as comments, but you still need to **submit individually** in both the virtual machine and Stream.