CSCI/CMPE 3333 Assignment 1

Instructor: Zhixiang Chen

Problem (170 points): This assignment is about expression trees, which are special cases of parse trees or grammar trees. There are three parts with different scores.

- **Part 1 (100 points): Binary Expression Trees.** For this part, you need to implement a program that takes a binary arithmetic expression as input, builds a binary expression tree for the input and then calculates the value of the input. A binary expression is composed of double type operands, binary arithmetic operators +, -, *, and /, and parentheses (and). Your program shall detect whether there are any syntactic errors in the input expression. Once the binary expression tree is built for the input expression that has no syntactic errors, your program shall print prefix and postfix expressions of the input expression. You shall use stack data structures to help build the binary expression tree.

  Your program shall read input expressions from a text file.

  Use the following examples to test your program:

  1. (10.24 + 5.4 * 2.5) / 6.7 + (12.5 * 20.67 + 10) * 25
  2. (10.24 + 5.4 * 2.5) / 6.7 + (12.5 * 20.67 + 10) * 25)
  3. (10.24 + 5.4 * 2.5) / 6.7 + (12.5 * 20.67 +) * 25

  Note:
  - Expression 1 has no syntactic errors.
  - Expressions 2 and 3 have one error each.

- **Part 2 (50 points): Boolean Expression Trees with Binary Relational and Arithmetic Operators.** This part is an extension of Part 1. Here, we consider how to construct an expression tree for any given Boolean expression with binary logical operators, binary relational operators and binary arithmetic operators. Of courses, parentheses are included as well.

  Binary logical operators are && and | |. Binary relational operators are >, >=, <, <=, ==, and !=. Binary arithmetic operators are +, -, *, and /.

  The value of a Boolean express is either True or False. Recall that in C++, any nonzero value is interpreted as True, while zero is interpreted as False.

Below is the order of precedence of the operators and parentheses:

Operator	Precedence (or Priority)	Association Order
( )	0 (highest precedence)	left to right
!, +, - (unary operator)	1	right to left
*, /	2	left to right
+, -	3	left to right
<, <=, >=, >	4	left to right
==, !=	5	left to right
&&	6	left to right
||	7 (lowest precedence)	left to right

Use the following example to test your program:

4. (10.24 + 5.4 * 2.5) >= 6.7 && (12.5 * 20.67 + 10) != 25
5. (10.24 >= 5.4 * 2.5 | | 4.6 < 6.7) && (12.5 * 20.67 + 10) <= 25

- **Part 3 (20 points): Working with Unary Operators.** This part is an extension of Part 2. Here, we consider how to work with unary operators. In particular, we would like to consider the unary logical operator ! in Boolean expressions. Because ! is a unary operator, a tree node constructed for it has one child node, so the expression tree for any input Boolean expression is, in general, not binary. But, if you spend some time to think this operator over, there will be not so much difficult at all to work with it, using the same solution to Part 2.

  Use the following example to test your program:

  6. (10.24 + 5.4 * 2.5) >= 6.7 && !((12.5 * 20.67 + 10) != 25)
  7. (10.24 >= 5.4 * 2.5 | | !(4.6 < 6.7)) | | (12.5 * 20.67 + 10) >= 25

Due Date:
The due date will be given via Blackboard.

Warning:
Any submission one week after the due date will not be accepted.

How to submit your work?
Please upload your source program files and your test results to Blackboard.