CS1020E | Lab 7 | Exercise 2

Infix Evaluator

Objectives

The main objective of this exercise is to practice using **recursion** to solve problem.

Problem Description

In this exercise, you are to write a program to evaluate a given infix arithmetic expression using recursion.

To simplify the problem, we assume there is no operator precedence and associativity rules, and every **subexpression** must be surrounded by a pair of parentheses. For example, "`1 + 2 + 3`" is not allowed, instead it must be given as "`( 1 + 2 ) + 3`" or "`1 + ( 2 + 3 )`". Moreover, because all operators have the same precedence, the intended order of evaluation must be explicitly specified using parentheses. For example, instead of "`1 + 2 * 3`", it must be given as "`1 + ( 2 * 3 )`".

You would only **get at most 50% of the marks** if **recursion** is not used, in a **correct** and **meaningful** way, to evaluate the arithmetic expression. You are also **not allowed to use stacks**. You are also required to **use the STL `queue`** to store the input data.

Add your code only to the parts of the files indicated. Do not modify any other part of the given code, and do not add new files.

Inputs

The input is a single line that contains an infix expression. The input ends when the end-of-file is reached (which can be entered from the keyboard using CTR-D).

The input infix expression can contain only the following *tokens*:

- positive integer numbers,
- the operators: `+`, `-`, `*` and `/`,
- the left and right parentheses: `(` and `)`.

The operator "/" is assumed to be the **integer division** operator. In the input line, every two tokens are separated by a space. You can assume that every input infix expression is valid.

Moreover, the input expression is allowed to contain redundant parentheses. For example, "`( 34 )`", "`( 1 + 2 )`", and "`( ( 1 ) + ( ( 2 * 3 ) ) )`" are all valid input expressions.

Outputs

A single integer number (may be negative) that is the result of evaluating the input expression.

Sample Input

```
( 1 + ( ( 7 * 3 ) / 2 ) ) * ( 5 )
```

Sample Output

```
55
```

Submission

You need to submit **ALL** your completed skeleton **`*.cpp`** and **`*.h`** files to CodeCrunch (https://codecrunch.comp.nus.edu.sg/) before the specified deadline. We will take only your latest submission.

Late submissions will not be accepted. The submission system in CodeCrunch will automatically close at the deadline.