CS3104 Compiler Lab, Autumn 2025
Assignment 1
Date: 01 August 2025

Instructions: Write your name and roll no on top of your source code as comments. Source code file name must be "Assignment1.c".
Submit your code in the following link: http://10.22.10.100/~abyaym/CS3104/submission/

Develop a naive C implementation based on your programming experience so far (PDS, Algo etc.). You will later see how a formal approach helps you simplify your life considerably.

Your task in this assignment is to build a simple calculator involving only positive integer operands and the arithmetic operators + and *. The usual rules of associativity and precedence apply to these expressions. The user is allowed to supply disambiguating parentheses in the expression. Here is an example of a type of expressions that your program will be able to handle.

4 + (14 * (6 + 12 + 2 * 16) + 9) * (11*3 + 15*8*13) * 19 + 5 * 18 + (1+10*(17 + 7)) + 20

This expression evaluates to 21459658.

Notice the following points.

- At the beginning of your program, the user enters an expression that is to be read as a string.
- At the end, your program should print the final numeric value of the expression, and exit.
- The integers in the expression may contain multiple digits.
- The user may use any number of spaces (including none) between an operator and an operand.
- The user may add unnecessary parentheses, like in the case of (1+10*(17 + 7)).
- Your program should detect whenever the user supplies an invalid arithmetic expression.

For the algorithm, you may use the following idea. Write two mutually recursive functions *evalsum*() and *evalterm*(). The first of these functions assumes that the expression is a sum of one or more terms. It attempts to locate the summands (terms) by identifying the outermost + signs (marked in red in the above example). The second function is called for each summand. This function in turn attempts to find the outermost * signs (marked in green for one term in the example above). Each factor in the term is again an expression (possibly parenthesized).

That's it! Simple as it may sound, writing a naive code even for a small problem like this would be rather involved. Let it be. You will later learn to ease that process anyway.

Let's start our journey.