Project Title: Arithmetic Expression Evaluator

Overview: This project is a C++ console-based application that evaluates arithmetic expressions entered by the user. It supports basic mathematical operations, including addition, subtraction, multiplication, and division, as well as parentheses for grouping. The program ensures valid input and continues running until the user explicitly exits by pressing 'E'.

Features:

- Accepts mathematical expressions as input.
- Supports operations: +, -, *, /.
- Handles parentheses for order of operations.
- Uses a stack-based approach to evaluate expressions.
- Detects and prompts for valid input if the user enters an incorrect expression.
- Runs continuously until the user enters 'E' to exit.

Technology Used:

- Programming Language: C++
- Standard Libraries: <iostream>, <sstream>, <stack>, <cctype>, <cmath>

Implementation Details:

1. **Operator Precedence Handling:** The program determines precedence using a function and processes operations accordingly.

2. **Expression Parsing:** The input is read as a string, and tokens (numbers, operators, parentheses) are extracted using stringstream.
3. **Stack-Based Evaluation:** Two stacks (one for values and one for operators) are used to evaluate the expression.
4. **Error Handling:**
    - Division by zero is detected and handled.
    - Invalid expressions prompt the user for correct input instead of terminating the program.
5. **User Interaction:** The program continuously asks for new expressions until the user chooses to exit by entering 'E'.

Usage Instructions:

1. Run the program in a C++-supported environment.
2. Enter a valid arithmetic expression (e.g., 3 + 5 * (2 - 1)).
3. The program evaluates and displays the result.
4. To exit, enter 'E'. If an invalid expression is entered, the program will prompt for a valid one.

Future Enhancements:

- Support for exponentiation and additional mathematical functions.
- Improved error messages for more detailed debugging.
- Implementing user-defined functions for advanced calculations.

Conclusion: This project demonstrates the use of stacks and operator precedence in evaluating arithmetic expressions. It ensures robustness by validating inputs and providing a user-friendly interaction. The program can be extended further to support more complex mathematical operations.