**Report on Evaluation of Expression Tree**

**Introduction**

An expression tree is a binary tree data structure where each internal node represents an operator, and each leaf node represents an operand. Expression trees are commonly used to represent mathematical expressions in a hierarchical manner. The evaluation of an expression tree involves traversing the tree and performing the corresponding operations to obtain the final result of the expression. This report provides an overview of the evaluation process for expression trees and includes a sample C code implementation.

**Expression Tree Structure**

In an expression tree, nodes are categorized into two types: operators and operands. Operators include addition (+), subtraction (-), multiplication (\*), and division (/). Operands are the values that participate in these operations. The tree is constructed in such a way that each internal node represents an operator, and its children represent the operands or subexpressions.

**Evaluation Algorithm**

The evaluation of an expression tree is typically done through recursive traversal. Starting from the root node, the algorithm checks whether the current node is an operator or an operand. If it's an operand (leaf node), the value is returned. If it's an operator, the algorithm recursively evaluates the left and right subtrees and performs the corresponding operation.

The following steps summarize the evaluation algorithm:

1. If the current node is a leaf (operand), return its value.
2. Recursively evaluate the left subtree.
3. Recursively evaluate the right subtree.
4. Apply the operator at the current node to the results of the left and right subtrees.
5. Return the final result.

**Sample C Code Implementation**

The provided C code demonstrates the evaluation of an expression tree. It defines a structure for tree nodes, includes functions to create nodes and evaluate the expression tree, and presents a sample expression tree: ((2+3)\*(5-4)). The code handles basic arithmetic operators (+, -, \*, /) and includes error checking, such as division by zero.

// [Code snippet included in the previous response]

**Conclusion**

Expression trees provide a structured way to represent mathematical expressions, and their evaluation is a fundamental operation in computer science and mathematics. The recursive nature of the evaluation algorithm allows for an elegant and efficient solution. Understanding the principles behind expression tree evaluation is essential for implementing mathematical expressions in various applications, including compilers, calculators, and symbolic mathematics systems. The provided C code serves as a practical example of how to implement expression tree evaluation in a programming language.