/* Expression Tree: Construct an Expression Tree from postfix and prefix expression. Perform recursive and non- recursive In-order.*/

```
#include <iostream>
#include <stack>
#include <cctype>

using namespace std;

// Define the expression tree node
struct Node {
    char data;
    Node* left;
    Node* right;
    Node(char value) : data(value), left(nullptr), right(nullptr) {}
};

// Function to check if a character is an operator
bool isOperator(char c) {
    return (c == '+' || c == '-' || c == '*' || c == '/');
}

// Function to construct an expression tree from postfix expression
Node* constructExpressionTreeFromPostfix(string postfix) {
    stack<Node*> s;

    for (char c : postfix) {
        if (isalnum(c)) {
            Node* newNode = new Node(c);
            s.push(newNode);
```

```
        } else if (isOperator(c)) {
            Node* rightNode = s.top();
            s.pop();
            Node* leftNode = s.top();
            s.pop();
            Node* newNode = new Node(c);
            newNode->left = leftNode;
            newNode->right = rightNode;
            s.push(newNode);
        }
    }

    return s.top();
}

// Recursive In-order traversal
void recursiveInOrder(Node* root) {
    if (root) {
        recursiveInOrder(root->left);
        cout << root->data << " ";
        recursiveInOrder(root->right);
    }
}

// Non-recursive In-order traversal
void nonRecursiveInOrder(Node* root) {
    stack<Node*> s;
    Node* current = root;
```

```
    while (current != nullptr || !s.empty()) {
        while (current != nullptr) {
            s.push(current);
            current = current->left;
        }
        current = s.top();
        s.pop();
        cout << current->data << " ";
        current = current->right;
    }
}

int main() {
    string postfix;
    cout << "Enter postfix expression: ";
    cin >> postfix;

    Node* root = constructExpressionTreeFromPostfix(postfix);

    cout << "Recursive In-order traversal: ";
    recursiveInOrder(root);
    cout << endl;

    cout << "Non-Recursive In-order traversal: ";
    nonRecursiveInOrder(root);
    cout << endl;

    return 0;
}
```