```
#include <iostream>
#include <stack>
#include <string>
using namespace std;

struct Node {
    char value;
    Node* left;
    Node* right;
    Node(char val) : value(val), left(nullptr), right(nullptr) {}
};

Node* constructPostfixTree(const string& postfix) {
    stack<Node*> s;
    for (char c : postfix) {
        if (isalnum(c)) {
            s.push(new Node(c));
        } else {
            Node* operatorNode = new Node(c);
            operatorNode->right = s.top(); s.pop();
            operatorNode->left = s.top(); s.pop();
            s.push(operatorNode);
        }
    }
    return s.top();
}

Node* constructPrefixTree(const string& prefix) {
    stack<Node*> s;
    for (int i = prefix.length() - 1; i >= 0; i--) {
        char c = prefix[i];
        if (isalnum(c)) {
            s.push(new Node(c));
        } else {
            Node* operatorNode = new Node(c);
            operatorNode->left = s.top(); s.pop();
            operatorNode->right = s.top(); s.pop();
            s.push(operatorNode);
        }
    }
    return s.top();
}

void inOrder(Node* root) {
    if (!root) return;
    inOrder(root->left);
    cout << root->value << " ";
    inOrder(root->right);
}

void preOrder(Node* root) {
    if (!root) return;
    cout << root->value << " ";
    preOrder(root->left);
```

```
    preOrder(root->right);
}

void postOrder(Node* root) {
    if (!root) return;
    postOrder(root->left);
    postOrder(root->right);
    cout << root->value << " ";
}

void nonRecursiveInOrder(Node* root) {
    stack<Node*> s;
    Node* current = root;
    while (current || !s.empty()) {
        while (current) {
            s.push(current);
            current = current->left;
        }
        current = s.top(); s.pop();
        cout << current->value << " ";
        current = current->right;
    }
}

void nonRecursivePreOrder(Node* root) {
    if (!root) return;
    stack<Node*> s;
    s.push(root);
    while (!s.empty()) {
        Node* current = s.top(); s.pop();
        cout << current->value << " ";
        if (current->right) s.push(current->right);
        if (current->left) s.push(current->left);
    }
}

void nonRecursivePostOrder(Node* root) {
    if (!root) return;
    stack<Node*> s1, s2;
    s1.push(root);
    while (!s1.empty()) {
        Node* current = s1.top(); s1.pop();
        s2.push(current);
        if (current->left) s1.push(current->left);
        if (current->right) s1.push(current->right);
    }
    while (!s2.empty()) {
        cout << s2.top()->value << " ";
        s2.pop();
    }
}

void takeInput(string& postfix, string& prefix) {
    cout << "Enter Postfix Expression: ";
```

```
    cin >> postfix;
    cout << "Enter Prefix Expression: ";
    cin >> prefix;
}

int main() {
    string postfix, prefix;
    takeInput(postfix, prefix);

    Node* postfixRoot = constructPostfixTree(postfix);
    Node* prefixRoot = constructPrefixTree(prefix);

    cout << "\nPostfix Expression Tree Traversals (" << postfix <<
"):\n";
    cout << "In-order (Recursive): "; inOrder(postfixRoot); cout << endl;
    cout << "Pre-order (Recursive): "; preOrder(postfixRoot); cout <<
endl;
    cout << "Post-order (Recursive): "; postOrder(postfixRoot); cout <<
endl;
    cout << "In-order (Non-Recursive): ";
nonRecursiveInOrder(postfixRoot); cout << endl;
    cout << "Pre-order (Non-Recursive): ";
nonRecursivePreOrder(postfixRoot); cout << endl;
    cout << "Post-order (Non-Recursive): ";
nonRecursivePostOrder(postfixRoot); cout << endl;

    cout << "\nPrefix Expression Tree Traversals (" << prefix << "):\n";
    cout << "In-order (Recursive): "; inOrder(prefixRoot); cout << endl;
    cout << "Pre-order (Recursive): "; preOrder(prefixRoot); cout <<
endl;
    cout << "Post-order (Recursive): "; postOrder(prefixRoot); cout <<
endl;
    cout << "In-order (Non-Recursive): ";
nonRecursiveInOrder(prefixRoot); cout << endl;
    cout << "Pre-order (Non-Recursive): ";
nonRecursivePreOrder(prefixRoot); cout << endl;
    cout << "Post-order (Non-Recursive): ";
nonRecursivePostOrder(prefixRoot); cout << endl;

    return 0;
}
```