```
#include <iostream>

#include <stack>

#include <string>

using namespace std;



struct Node {

char data;

Node* left;

Node* right;

Node(char val) : data(val), left(nullptr), right(nullptr) {}

};



class ExpressionTree {

public:



Node* constructFromPostfix(const string& postfix) {

stack<Node*> st;
```

```
for (char ch : postfix) {

if (isalnum(ch)) {

st.push(new Node(ch));

} else {

Node* node = new Node(ch);

node->right = st.top(); st.pop();

node->left = st.top(); st.pop();

st.push(node);

}

}

return st.top();

}
```

```
Node* constructFromPrefix(const string& prefix) {

stack<Node*> st;
```

```
for (int i = prefix.size() - 1; i >= 0; --i) {

char ch = prefix[i];

if (isalnum(ch)) {

st.push(new Node(ch));

} else {

Node* node = new Node(ch);

node->left = st.top(); st.pop();

node->right = st.top(); st.pop();

st.push(node);

}

}

return st.top();

}



// Recursive Traversals

void inOrder(Node* root) {

if (root) {

inOrder(root->left);
```

```
cout << root->data << " ";

inOrder(root->right);

}

}



void preOrder(Node* root) {

if (root) {

cout << root->data << " ";

preOrder(root->left);

preOrder(root->right);

}

}



void postOrder(Node* root) {

if (root) {

postOrder(root->left);

postOrder(root->right);
```

```
cout << root->data << " ";

}

}

};



int main() {

ExpressionTree tree;

string postfix, prefix;



cout << "Enter postfix expression: ";

cin >> postfix;

Node* postfixRoot = tree.constructFromPostfix(postfix);



cout << "\nIn-order traversal: ";

tree.inOrder(postfixRoot);

cout << "\nPre-order traversal: ";
```

```
tree.preOrder(postfixRoot);

cout << "\nPost-order traversal: ";

tree.postOrder(postfixRoot);




cout << "\n\nEnter prefix expression: ";

cin >> prefix;

Node* prefixRoot = tree.constructFromPrefix(prefix);




cout << "\nIn-order traversal: ";

tree.inOrder(prefixRoot);

cout << "\nPre-order traversal: ";

tree.preOrder(prefixRoot);

cout << "\nPost-order traversal: ";

tree.postOrder(prefixRoot);




return 0;

}
```