**Practical GroupB\_07**

**Aim:**

Construct an expression tree from the given prefix expression eg. +--a\*bc/def and traverse it using post order traversal (non recursive) and then delete the entire tree.

**Code:**

#include <iostream>

using namespace std;

struct Node {

char data;

Node\* left;

Node\* right;

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

};

class ExpressionTree {

public:

Node\* constructTree(const string& prefixExpression);

void postOrderTraversal(Node\* root);

void deleteTree(Node\* root);

private:

bool isOperator(char c);

};

Node\* ExpressionTree::constructTree(const string& prefixExpression) {

int len = prefixExpression.length();

Node\* stack[len];

int top = -1;

for (int i = len - 1; i >= 0; i--) {

char c = prefixExpression[i];

if (isOperator(c)) {

Node\* newNode = new Node(c);

newNode->left = stack[top--];

newNode->right = stack[top--];

stack[++top] = newNode;

} else {

stack[++top] = new Node(c);

}

}

return stack[top];

}

void ExpressionTree::postOrderTraversal(Node\* root) {

if (root == nullptr) {

return;

}

postOrderTraversal(root->left);

postOrderTraversal(root->right);

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

}

void ExpressionTree::deleteTree(Node\* root) {

if (root == nullptr) {

return;

}

deleteTree(root->left);

deleteTree(root->right);

delete root;

}

bool ExpressionTree::isOperator(char c) {

return (c == '+' || c == '-' || c == '\*' || c == '/');

}

int main() {

ExpressionTree expressionTree;

string prefixExpression = "+--a\*bc/def";

Node\* root = expressionTree.constructTree(prefixExpression);

cout << "Post-order traversal: ";

expressionTree.postOrderTraversal(root);

cout << endl;

expressionTree.deleteTree(root);

return 0;

}