**ASSIGNMENT NO 5**

**TITLE : PREFIX AND POSTFIX EXPRESSIONS**

CODE:

#include <iostream>

#include <cstring>

#include <cctype> // For isalpha()

#include <stack>

using namespace std;

struct Node {

char data;

Node \*left, \*right;

Node(char val) {

data = val;

left = right = nullptr;

}

};

class Tree {

public:

Node \*root;

Tree() {

root = nullptr;

}

// Build expression tree from prefix expression

void buildExpressionTree(const char\* prefix) {

stack<Node\*> st;

int n = strlen(prefix);

// Traverse from right to left (since it's prefix)

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

char ch = prefix[i];

// If operand, create node and push

if (isalpha(ch)) {

Node\* newNode = new Node(ch);

st.push(newNode);

}

// If operator, pop two nodes and make them children

else {

Node\* newNode = new Node(ch);

if (!st.empty()) {

newNode->left = st.top();

st.pop();

}

if (!st.empty()) {

newNode->right = st.top();

st.pop();

}

st.push(newNode);

}

}

if (!st.empty())

root = st.top();

cout << "Expression tree built successfully.\n";

}

// Non-recursive postorder traversal using 2 stacks

void displayPostfixNonRecursive(Node\* node) {

if (!node) {

cout << "Tree is empty.\n";

return;

}

stack<Node\*> s1, s2;

s1.push(node);

while (!s1.empty()) {

Node\* curr = s1.top();

s1.pop();

s2.push(curr);

if (curr->left)

s1.push(curr->left);

if (curr->right)

s1.push(curr->right);

}

cout << "Postfix Expression: ";

while (!s2.empty()) {

cout << s2.top()->data;

s2.pop();

}

cout << endl;

}

// Delete tree using postorder

void deleteTree(Node\* node) {

if (!node) return;

deleteTree(node->left);

deleteTree(node->right);

cout << "Deleting node: " << node->data << endl;

delete node;

}

};

int main() {

Tree tree;

char expression[50];

int choice;

do {

cout << "\n=== MENU ===\n";

cout << "1. Enter prefix expression\n";

cout << "2. Display postfix expression (non-recursive)\n";

cout << "3. Delete expression tree\n";

cout << "4. Exit\n";

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1:

cout << "Enter the prefix expression (e.g., +--a\*bc/def): ";

cin >> expression;

tree.buildExpressionTree(expression);

break;

case 2:

tree.displayPostfixNonRecursive(tree.root);

break;

case 3:

if (tree.root) {

tree.deleteTree(tree.root);

tree.root = nullptr;

cout << "Tree deleted successfully.\n";

} else {

cout << "Tree is already empty.\n";

}

break;

case 4:

cout << "Exiting...\n";

break;

default:

cout << "Invalid choice. Please choose between 1 and 4.\n";

}

} while (choice != 4);

return 0;

}  
OUTPUT:  
