**PRACTICAL-4**

SAI ANANT PATIL

COB208

#include <iostream>

#include <cstring>

#include <cctype> // For isalpha

using namespace std;

struct Node {

char data;

Node \*left, \*right;

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

};

class Tree {

public:

Node \*root;

Tree() : root(nullptr) {}

void buildExpressionTree(const char \*prefix) {

Node \*stack[50];

int top = -1;

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

if (isalpha(prefix[i])) {

stack[++top] = new Node(prefix[i]);

} else {

Node \*node = new Node(prefix[i]);

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

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

stack[++top] = node;

}

}

root = stack[top];

}

void displayPostfix(Node \*node) {

if (!node) return;

displayPostfix(node->left);

displayPostfix(node->right);

cout << node->data;

}

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 << "1 -> Enter prefix expression\n";

cout << "2 -> Display postfix expression\n";

cout << "3 -> Delete tree\n";

cout << "4 -> Exit\n";

cout << "Choose an option (1-4): ";

cin >> choice;

switch (choice) {

case 1:

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

cin >> expression;

tree.buildExpressionTree(expression);

break;

case 2:

if (tree.root) {

tree.displayPostfix(tree.root);

cout << endl;

} else {

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

}

break;

case 3:

if (tree.root) {

tree.deleteTree(tree.root);

tree.root = nullptr;

} else {

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

}

break;

case 4:

cout << "\n// END OF CODE\n";

break;

default:

cout << "Choose a valid option (1-4).\n";

}

} while (choice != 4);

return 0;

}

