**Name: Affan Shaikh**

**Roll no: COB227**

**Code :**

#include <iostream>

#include <cstring>

#include <cctype>

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 (inti = 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;

    }

};

intmain() {

    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:

                break;

            default:

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

        }

    } while (choice != 4);

    return 0;

}

**Output :**

1 -> Enter prefix expression

2 -> Display postfix expression

3 -> Delete tree

4 -> Exit

Choose an option (1-4): 1

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

1 -> Enter prefix expression

2 -> Display postfix expression

3 -> Delete tree

4 -> Exit

Choose an option (1-4): 2

abc\*-de/-f+

1 -> Enter prefix expression

2 -> Display postfix expression

3 -> Delete tree

4 -> Exit

Choose an option (1-4): 3

Deleting node: a

Deleting node: b

Deleting node: c

Deleting node: \*

Deleting node: -

Deleting node: d

Deleting node: e

Deleting node: /

Deleting node: -

Deleting node: f

Deleting node: +

1 -> Enter prefix expression

2 -> Display postfix expression

3 -> Delete tree

4 -> Exit

Choose an option (1-4): 4