#include <iostream>

using namespace std;

#define MAX 100

struct Node {

char data;

Node\* left;

Node\* right;

Node(char val) {

data = val;

left = right = nullptr;

}

};

class Stack {

Node\* arr[MAX];

int top;

public:

Stack() { top = -1; }

bool isEmpty() { return top == -1; }

bool isFull() { return top == MAX - 1; }

void push(Node\* node) {

if (isFull()) {

cout << "Stack Overflow\n";

return;

}

arr[++top] = node;

}

Node\* pop() {

if (isEmpty()) {

cout << "Stack Underflow\n";

return nullptr;

}

return arr[top--];

}

Node\* peek() {

if (isEmpty()) return nullptr;

return arr[top];

}

};

bool isOperator(char ch) {

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

}

Node\* constructTree(const char\* prefix) {

Stack st;

int len = 0;

while (prefix[len] != '\0') len++;

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

char ch = prefix[i];

Node\* node = new Node(ch);

if (isOperator(ch)) {

Node\* left = st.pop();

Node\* right = st.pop();

node->left = left;

node->right = right;

}

st.push(node);

}

return st.pop();

}

void postorderNonRecursive(Node\* root) {

if (root == nullptr) return;

Stack s1, s2;

s1.push(root);

while (!s1.isEmpty()) {

Node\* node = s1.pop();

s2.push(node);

if (node->left)

s1.push(node->left);

if (node->right)

s1.push(node->right);

}

cout << "Postorder traversal: ";

while (!s2.isEmpty()) {

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

}

cout << endl;

}

void deleteTree(Node\* root) {

if (root == nullptr) return;

Stack s1, s2;

s1.push(root);

while (!s1.isEmpty()) {

Node\* node = s1.pop();

s2.push(node);

if (node->left)

s1.push(node->left);

if (node->right)

s1.push(node->right);

}

while (!s2.isEmpty()) {

Node\* node = s2.pop();

delete node;

}

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

}

int main() {

char prefix[MAX];

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

cin >> prefix;

Node\* root = constructTree(prefix);

postorderNonRecursive(root);

deleteTree(root);

return 0;

}