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

#include <iostream>

#include <stack>

#include <cctype>

using namespace std;

// Tree node

struct Node {

char data;

Node\* left;

Node\* right;

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

};

// Check if character is operator

bool isOperator(char c) {

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

}

// Construct expression tree from prefix expression

Node\* constructTreeFromPrefix(const string& prefix) {

stack<Node\*> st;

// Traverse from right to left

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

char c = prefix[i];

Node\* node = new Node(c);

if (isOperator(c)) {

// Pop two operands

Node\* left = st.top(); st.pop();

Node\* right = st.top(); st.pop();

node->left = left;

node->right = right;

}

st.push(node);

}

return st.top();

}

// Non-recursive postorder traversal using two stacks

void postorderTraversal(Node\* root) {

if (!root) return;

stack<Node\*> s1, s2;

s1.push(root);

while (!s1.empty()) {

Node\* current = s1.top(); s1.pop();

s2.push(current);

if (current->left) s1.push(current->left);

if (current->right) s1.push(current->right);

}

// Print postorder

cout << "Postorder Traversal (non-recursive): ";

while (!s2.empty()) {

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

s2.pop();

}

cout << endl;

}

// Delete entire tree using postorder traversal

void deleteTree(Node\* root) {

if (!root) return;

stack<Node\*> s1, s2;

s1.push(root);

while (!s1.empty()) {

Node\* current = s1.top(); s1.pop();

s2.push(current);

if (current->left) s1.push(current->left);

if (current->right) s1.push(current->right);

}

// Delete nodes in postorder

while (!s2.empty()) {

Node\* node = s2.top(); s2.pop();

delete node;

}

cout << "Tree deleted successfully." << endl;

}

int main() {

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

Node\* root = constructTreeFromPrefix(prefix);

postorderTraversal(root); // Expect: a b c \* - d e f / - +

deleteTree(root);

return 0;

}

/\*OUTPUT

Postorder Traversal (non-recursive): a b c \* - d e f / - +

Tree deleted successfully.\*/