**PRACTICAL NO:5**

**PROGRAM:**

#include <iostream>

#include <stack>

#include <string>

using namespace std;

struct Node

{

char value;

Node\* left;

Node\* right;

Node(char val) : value(val), left(NULL), right(NULL) {}

};

bool isOperator(char c) {

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

}

Node\* constructTree(string prefix) {

stack<Node\*> stk;

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

if (isOperator(prefix[i])) {

Node\* op1 = stk.top(); stk.pop();

Node\* op2 = stk.top(); stk.pop();

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

node->left = op1;

node->right = op2;

stk.push(node);

} else {

stk.push(new Node(prefix[i]));

}

}

return stk.top();

}

void deleteTree(Node\* node) {

if (node == NULL)

return;

deleteTree(node->left);

deleteTree(node->right);

delete node;

}

void postOrderTraversal(Node\* root) {

if (!root) return;

stack<Node\*> s1, s2;

s1.push(root);

while (!s1.empty()) {

Node\* node = s1.top();

s1.pop();

s2.push(node);

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

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

}

while (!s2.empty()) {

Node\* node = s2.top();

s2.pop();

cout << node->value << ' ';

}

}

int main() {

string prefix;

cout << "Enter the prefix expression: ";

cin >> prefix;

Node\* root = constructTree(prefix);

cout << "Post-order traversal: ";

postOrderTraversal(root);

cout << endl;

deleteTree(root);

return 0;

}OUTPUT:

Enter the prefix expression: +--a\*bc/def

Post-order traversal: a b c \* - d e / - f +