#include <iostream>

using namespace std;

class Node

{

public:

char value;

Node \*left;

Node \*right;

Node \*next;

Node(char c)

{

this->value = c;

left = nullptr;

right = nullptr;

next = nullptr;

}

Node()

{

left = nullptr;

right = nullptr;

}

friend class Stack;

friend class ExpressionTree;

};

class Stack

{

Node \*head;

public:

Stack() : head(nullptr) {}

void push(Node \*x)

{

if (head == nullptr)

{

head = x;

}

else

{

x->next = head;

head = x;

}

}

Node \*pop()

{

if (head == nullptr)

return nullptr;

Node \*p = head;

head = head->next;

return p;

}

bool isEmpty()

{

return head == nullptr;

}

friend class ExpressionTree;

};

class ExpressionTree

{

public:

void inorder(Node \*x)

{

if (x == nullptr)

return;

inorder(x->left);

cout << x->value << " ";

inorder(x->right);

}

void preorder(Node \*x)

{

if (x == nullptr)

return;

cout << x->value << " ";

preorder(x->left);

preorder(x->right);

}

void postorder(Node \*x)

{

if (x == nullptr)

return;

postorder(x->left);

postorder(x->right);

cout << x->value << " ";

}

};

int main()

{

string s = "ABC\*+D/";

Stack e;

ExpressionTree tree;

Node \*x, \*y, \*z;

int l = s.length();

for (int i = 0; i < l; i++)

{

if (s[i] == '+' || s[i] == '-' || s[i] == '\*' || s[i] == '/' || s[i] == '^')

{

z = new Node(s[i]);

x = e.pop();

y = e.pop();

z->left = y;

z->right = x;

e.push(z);

}

else

{

z = new Node(s[i]);

e.push(z);

}

}

Node \*root = e.pop();

cout << "In-order Traversal: ";

tree.inorder(root);

cout << endl;

cout << "Pre-order Traversal: ";

tree.preorder(root);

cout << endl;

cout << "Post-order Traversal: ";

tree.postorder(root);

cout << endl;

return 0;

}