#include <bits/stdc++.h>

#include <stack>

using namespace std;

struct Node {

int data;

Node\* next;

Node(int d) {

data = d;

next = nullptr;

}

};

struct Stack {

private:

Node\* top;

public:

Stack() {

top = nullptr;

}

bool isEmpty() {

return top == nullptr;

}

void push(int data) {

Node\* newNode = new Node(data);

newNode->next = top;

top = newNode;

}

int pop() {

if (isEmpty()) {

cerr << "Stack is empty. Cannot pop." << endl;

return -1;

}

int data = top->data;

Node\* temp = top;

top = top->next;

delete temp;

return data;

}

int peek() {

if (isEmpty()) {

cerr << "Stack is empty. Cannot peek." << endl;

return -1;

}

return top->data;

}

};

bool isOperator(char c) {

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

}

int prec(char c) {

if (c == '^')

return 3;

else if (c == '\*' || c == '/')

return 2;

else if (c == '+' || c == '-')

return 1;

else

return -1;

}

string infixToPostfix(string infix) {

int l = infix.size();

Stack char\_stack;

string output;

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

if (isalpha(infix[i]) || isdigit(infix[i]))

output += infix[i];

else if (infix[i] == '(')

char\_stack.push('(');

else if (infix[i] == ')') {

while (!char\_stack.isEmpty() && char\_stack.peek() != '(') {

output += char\_stack.pop();

}

char\_stack.pop();

} else {

while (!char\_stack.isEmpty() && prec(infix[i]) <= prec(char\_stack.peek())) {

output += char\_stack.pop();

}

char\_stack.push(infix[i]);

}

}

while (!char\_stack.isEmpty()) {

output += char\_stack.pop();

}

return output;

}

string infixToPrefix(string infix) {

int k = infix.size();

reverse(infix.begin(), infix.end());

for (int i = 0; i < k; i++) {

if (infix[i] == '(') {

infix[i] = ')';

} else if (infix[i] == ')') {

infix[i] = '(';

}

}

string prefix = infixToPostfix(infix);

reverse(prefix.begin(), prefix.end());

return prefix;

}

int evaluatePostfix(string postfix) {

int l = postfix.size();

Stack num\_stack;

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

if (isdigit(postfix[i])) {

num\_stack.push(postfix[i] - '0');

} else if (isOperator(postfix[i])) {

int operand2 = num\_stack.pop();

int operand1 = num\_stack.pop();

int result;

switch (postfix[i]) {

case '+':

result = operand1 + operand2;

break;

case '-':

result = operand1 - operand2;

break;

case '\*':

result = operand1 \* operand2;

break;

case '/':

result = operand1 / operand2;

break;

case '^':

result = pow(operand1, operand2);

break;

default:

result = 0;

}

num\_stack.push(result);

}

}

return num\_stack.pop();

}

int evaluatePrefix(string prefix) {

int l = prefix.size();

Stack num\_stack;

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

if (isdigit(prefix[i])) {

num\_stack.push(prefix[i] - '0');

} else if (isOperator(prefix[i])) {

int operand1 = num\_stack.pop();

int operand2 = num\_stack.pop();

int result;

switch (prefix[i]) {

case '+':

result = operand1 + operand2;

break;

case '-':

result = operand1 - operand2;

break;

case '\*':

result = operand1 \* operand2;

break;

case '/':

result = operand1 / operand2;

break;

case '^':

result = pow(operand1, operand2);

break;

default:

result = 0;

}

num\_stack.push(result);

}

}

return num\_stack.pop();

}

int main() {

int a;

do {

cout << "1. Infix to Postfix\n2. Infix to Prefix\n3. Evaluate Postfix\n4. Evaluate Prefix" << endl;

int choice;

cin >> choice;

cin.ignore();

switch (hoice) {

case 1: {

string s;

cout << "Enter the infix expression: ";

getline(cin, s);

cout << "Postfix: " << infixToPostfix(s) << endl;

break;

}

case 2: {

string s;

cout << "Enter the infix expression: ";

getline(cin, s);

cout << "Prefix: " << infixToPrefix(s) << endl;

break;

}

case 3: {

string s;

cout << "Enter the postfix expression: ";

getline(cin, s);

cout << "Result: " << evaluatePostfix(s) << endl;

break;

}

case 4: {

string s;

cout << "Enter the prefix expression: ";

getline(cin, s);

cout << "Result: " << evaluatePrefix(s) << endl;

break;

}

default:

cout << "Invalid choice. Please try again." << endl;

}

cout << "Do you want to continue? Press 1: ";

cin >> a;

} while (a == 1);

return 0;

}