#include<iostream>

#include<stack>

#include<string>

using namespace std;

bool isOperator(char c){

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

}

int precedence(char op){

if(op=='+'||op=='-')

return 1;

if(op=='\*'||op=='/')

return 2;

return 0;

}

string infixToPostfix(const string & infix){

string postfix;

stack<char>operatorStack;

for(char c:infix){

if(isalnum(c)){

postfix+=c;

}else if(c=='('){

operatorStack.push(c);

}else if(c==')'){

while(!operatorStack.empty()&&operatorStack.top()!='('){

postfix+=operatorStack.top();

operatorStack.pop();

}

if(!operatorStack.empty()&&operatorStack.top()=='('){

operatorStack.pop();

}

}

else if(isOperator(c)){

while(!operatorStack.empty()&&precedence(operatorStack.top())>=precedence(c)){

postfix+=operatorStack.top();

operatorStack.pop();

}

operatorStack.push(c);

}

}

while(!operatorStack.empty()){

postfix+=operatorStack.top();

operatorStack.pop();

}

return postfix;

}

int evaluatePostfix(const string&postfix){

stack<int>operandStack;

for(char c:postfix){

if(isalnum(c)){

operandStack.push(c-'0');

}else if(isOperator(c)){

int operand2=operandStack.top();

operandStack.pop();

int operand1=operandStack.top();

operandStack.pop();

int result;

switch(c){

case'+':

result=operand1+operand2;

break;

case'-':

result=operand1-operand2;

break;

case'\*':

result=operand1\*operand2;

break;

case'/':

result=operand1/operand2;

break;

}

operandStack.push(result);

}

}

return operandStack.top();

}

int main(){

string infixExpression="a+b\*c-d/e";

string postfixExpression=infixToPostfix(infixExpression);

cout<<"InfixExpression:"<<infixExpression<<endl;

cout<<"PostfixExpression:"<<postfixExpression<<endl;

int result=evaluatePostfix(postfixExpression);

cout<<"Result:"<<result<<endl;

return 0;

}