#include <iostream>

#include <string>

#include<stack>

using namespace std;

int OperatorPrecedence(char cOperator){

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

return 1;

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

return 2;

else if(cOperator == '^')

return 3;

else

return -1;

}

string infixToPostfix(string infixExp){

string postfixExp;

stack<char> expStack;

cout<<"infixExp = "<<infixExp<<endl;

//a+b\*(c^d-e)^(f+g\*h)-i

for(int i=0; i< infixExp.length(); i++){

//cout<<infixExp[i];

if( (infixExp[i] >= 'A' &&

infixExp[i] <= 'Z' )||

(infixExp[i] >= 'a' &&

infixExp[i] <= 'z') ){

postfixExp+=infixExp[i];

cout<<"pushed opd " << infixExp[i]<<endl;

}

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

expStack.push(infixExp[i]);

cout<<"pushed ( "<<endl;

}

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

while(!expStack.empty() && expStack.top() != '(')

{

char c = expStack.top();

cout<<"pop ) " <<c<<endl;

expStack.pop();

postfixExp += c;

}

if(expStack.top() == '(')

{

char c = expStack.top();

cout<<"pop ) " <<endl;

expStack.pop();

}

}

else{

while(!expStack.empty() &&

OperatorPrecedence(infixExp[i]) <= OperatorPrecedence(expStack.top()) ){

char op = expStack.top();

cout<<"opr " << op<<endl;

expStack.pop();

postfixExp+=op;

}

expStack.push(infixExp[i]);

cout<<"pushed op " << infixExp[i]<<endl;

}

}

//Pop all the remaining elements from the stack

while(!expStack.empty() )

{

char c = expStack.top();

expStack.pop();

postfixExp += c;

}

cout<<endl;

//postfixExp.assign(infixExp);

return postfixExp;

}

int main(){

string infixExp = "a+b\*(c^d-e)^(f+g\*h)-i";

string postfixExp;

postfixExp.assign(infixToPostfix(infixExp));

cout<<"postfixExp = "<<postfixExp<<endl;

return 0;

}

//abcd^e-fgh\*+^\*+i-

// abcd^e-fgh\*+^\*+i-