//convert postfix to infrix expression

#include <iostream>

#include <stack>

using namespace std;

//compare the precedence of each operation

int input\_pred (char operator1)

{

if (operator1 == (char )'+' || operator1 == (char) '-')

{

return 1;

}

if (operator1 == (char) '\*' || operator1 == (char) '/' || operator1 == (char) '%')

{

return 2;

}

return 3;

}

//see if the string has the target value

bool has(string str, char target)

{

return str.find(target) != std::string::npos;

}

//see if string a has a operation stronger than string b

bool is\_stronger (string a, string b)

{

if ( has(a,'^') )

{

if ( has(b,'\*') || has(b,'/') || has (b,'%') )

{

return true;

}

if ( has(b,'+') || has(b,'-') )

{

return true;

}

}

if ( has(a,'\*') || has(a,'/') || has (a,'%') )

{

if ( has(b,'+') || has(b,'-') )

{

return true;

}

}

return false;

}

//convert postfix to infix

string postfix\_to\_infix(string postfix)

{

stack <string> s;

stack <char> r;

char operator1, operator2 = '\0';

string op1, op2;

for (int i = 0; postfix[i] != '\0'; i++)

{

if ((postfix[i] >= 48 && postfix[i] <= 57) || (postfix[i] >= 65 && postfix[i] <= 90) || (postfix[i] >= 97 && postfix[i] <= 122))

{

string op(1, postfix[i]);

s.push(op);

}

else

{

op1 = s.top();

s.pop();

op2 = s.top();

s.pop();

operator1 = postfix[i];

if (op1.length() >= 3 && op2.length() >= 3)

{

// equal precendence

if(is\_stronger(op1, op2) && is\_stronger(op2, op1))

{

s.push( op2 + postfix[i] + op1 );

}

else if (is\_stronger(op1, op2))

{

s.push( '(' + op2 + ')' + postfix[i] + op1 );

}

else

{

s.push( op2 + postfix[i] + '(' + op1 + ')');

}

}

else if (operator2 == '\0' || (op1.length() == 1 && op2.length() ==1 ))

{

s.push(op2 + postfix[i] + op1);

}

else if (input\_pred(operator1) > input\_pred(operator2))

{

if (op1.length() == 1)

{

s.push('(' + op2 + ')' + postfix[i] + op1);

}

else

{

s.push( op2 + postfix[i] + '(' + op1 + ')' );

}

}

else if (input\_pred(operator1) <= input\_pred(operator2))

{

s.push(op2 + postfix[i] + op1);

}

operator2 = operator1;

}

}

return s.top();

}

int main ()

{

string a = "12+3\*";

cout<<"Post Fix: "<<a<<endl;

cout<<"Infix: "<<postfix\_to\_infix (a)<<endl<<endl;

a = "31\*2-324^^\*";

cout<<"Post Fix: "<<a<<endl;

cout<<"Infix: "<<postfix\_to\_infix (a)<<endl<<endl;

a = "abcd\*+\*e/";

cout<<"Post Fix: "<<a<<endl;

cout<<"Infix: "<<postfix\_to\_infix (a)<<endl<<endl;

return 0;

}