Implement C++ program for expression conversion as infix to postfix and its evaluation using stack based on given conditions: 1. Operands and operator, both must be single character. 2. Input Postfix expression must be in a desired format. 3. Only '+', ' -', '\*' and '/ ' operators are expected  
  
#include <iostream>

#include <stack>

#include <cctype>

using namespace std;

int precedence(char op) {

if (op == '+' || op == '-') return 1;

if (op == '\*' || op == '/') return 2;

return 0;

}

string infixToPostfix(string infix) {

stack<char> st;

string postfix;

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

char ch = infix[i];

if (isalnum(ch)) {

postfix += ch;

}

else if (ch == '(') {

st.push(ch);

}

else if (ch == ')') {

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

postfix += st.top();

st.pop();

}

st.pop();

}

else {

while (!st.empty() && precedence(st.top()) >= precedence(ch)) {

postfix += st.top();

st.pop();

}

st.push(ch);

}

}

while (!st.empty()) {

postfix += st.top();

st.pop();

}

return postfix;

}

int evaluatePostfix(string postfix) {

stack<int> st;

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

char ch = postfix[i];

if (isdigit(ch)) {

st.push(ch - '0');

}

else {

int val2 = st.top(); st.pop();

int val1 = st.top(); st.pop();

switch (ch) {

case '+': st.push(val1 + val2); break;

case '-': st.push(val1 - val2); break;

case '\*': st.push(val1 \* val2); break;

case '/': st.push(val1 / val2); break;

}

}

}

return st.top();

}

int main() {

string infix;

cout << "Enter an infix expression: ";

cin >> infix;

string postfix = infixToPostfix(infix);

cout << "Postfix expression: " << postfix << endl;

int result = evaluatePostfix(postfix);

cout << "Evaluation result: " << result << endl;

return 0;

}