'''

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;
}

bool isOperator(char ch) {
    return (ch == '+' || ch == '-' || ch == '*' || ch == '/');
}
```

```
bool isOperand(char ch) {
    return isalnum(ch);
}

string infixToPostfix(const string& infix) {
    stack<char> s;
    string postfix = "";

    for (char ch : infix) {
        if (isOperand(ch)) {
            postfix += ch;
        } else if (ch == '(') {
            s.push(ch);
        } else if (ch == ')') {
            while (!s.empty() && s.top() != '(') {
                postfix += s.top();
                s.pop();
            }
            s.pop(); // Pop '('
        } else if (isOperator(ch)) {
            while (!s.empty() && precedence(s.top()) >= precedence(ch)) {
                postfix += s.top();
                s.pop();
            }
```

```
            s.push(ch);
        }
    }

    while (!s.empty()) {
        postfix += s.top();
        s.pop();
    }

    return postfix;
}

int evaluatePostfix(const string& postfix) {
    stack<int> s;

    for (char ch : postfix) {
        if (isOperand(ch)) {
            s.push(ch - '0'); // Convert char to int
        } else if (isOperator(ch)) {
            int operand2 = s.top();
            s.pop();
            int operand1 = s.top();
            s.pop();

            switch (ch) {
```

```
            case '+': s.push(operand1 + operand2); break;
            case '-': s.push(operand1 - operand2); break;
            case '*': s.push(operand1 * operand2); break;
            case '/': s.push(operand1 / operand2); break;
        }
      }
    }

    return s.top();
}

int main() {
    string infixExpression;
    cout << "Enter infix expression: ";
    cin >> infixExpression;

    string postfixExpression = infixToPostfix(infixExpression);
    cout << "Postfix expression: " << postfixExpression << endl;

    int result = evaluatePostfix(postfixExpression);
    cout << "Result after evaluation: " << result << endl;

    return 0;
}
```