

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.**

CODE:-

```
#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 (isdigit(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 (isdigit(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;
}

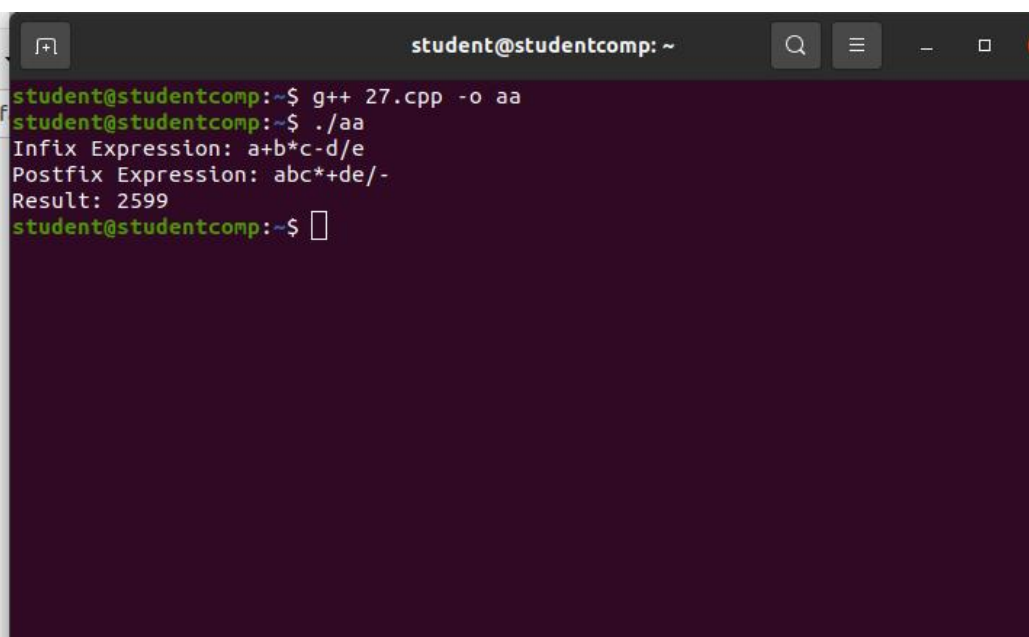
operandStack.push(result);
}
}
return operandStack.top();
}
int main() {
    string infixExpression = "a+b*c-d/e";
    string postfixExpression = infixToPostfix(infixExpression);
    cout << "Infix Expression: " << infixExpression << endl;
    cout << "Postfix Expression: " << postfixExpression << endl;

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

    return 0;
}

```

OUTPUT:-



```

student@studentcomp: ~
student@studentcomp:~$ g++ 27.cpp -o aa
student@studentcomp:~$ ./aa
Infix Expression: a+b*c-d/e
Postfix Expression: abc*+de/-
Result: 2599
student@studentcomp:~$ 

```