Practical 8

Mayur Zope Comp A 75

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 <string>

#include <cctype>

#include <sstream>


using namespace std;


// Function to return the precedence of operators

int precedence(char op) {

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

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

   return 0;

}


// Function to check if the character is an operator

bool isOperator(char c) {

   return c == '+' || c == '-' || c == '*' || c == '/';

}


// Function to convert infix expression to postfix

string infixToPostfix(const string &infix) {

   stack<char> operators;

   string postfix;
```

```
    for (char c : infix) {
        if (isalnum(c)) {  // If the character is an operand
            postfix += c;
        } else if (isOperator(c)) {  // If the character is an operator
            while (!operators.empty() && precedence(operators.top())
>= precedence(c)) {
                postfix += operators.top();
                operators.pop();
            }
            operators.push(c);
        }
    }

    // Pop all remaining operators from the stack
    while (!operators.empty()) {
        postfix += operators.top();
        operators.pop();
    }

    return postfix;
}

// Function to evaluate postfix expression
int evaluatePostfix(const string &postfix) {
    stack<int> operands;

    for (char c : postfix) {
```

```
        if (isdigit(c)) {  // If the character is an operand
            operands.push(c - '0');  // Convert char to int
        } else if (isOperator(c)) {  // If the character is an operator
            int right = operands.top(); operands.pop();
            int left = operands.top(); operands.pop();
            switch (c) {
                case '+': operands.push(left + right); break;
                case '-': operands.push(left - right); break;
                case '*': operands.push(left * right); break;
                case '/':
                    if (right != 0) {
                        operands.push(left / right);
                    } else {
                        cerr << "Error: Division by zero." << endl;
                        return 0;
                    }
                    break;
            }
        }
    }

    return operands.top();  // The result is the last remaining item on
the stack
}

int main() {
    string infix;
```

```
    cout << "Enter an infix expression (single-character operands
and operators only): ";
    cin >> infix;

    string postfix = infixToPostfix(infix);
    cout << "Postfix expression: " << postfix << endl;

    int result = evaluatePostfix(postfix);
    cout << "Evaluation of Postfix expression: " << result << endl;

    return 0;
}
// OUTPUT
```