**PRACTICAL NO:10**

**INPUT:**

#include <iostream>

#include <cstring>

#include <stack>

#include <cctype>

using namespace std;

int getWeight(char ch) {

switch (ch) {

case '/':

case '\*': return 2;

case '+':

case '-': return 1;

default: return 0;

}

}

void infix2postfix(char infix[], char postfix[], int size) {

stack<char> s;

int k = 0;

char ch;

for (int i = 0; i < size; i++) {

ch = infix[i];

if (ch == '(') {

s.push(ch);

} else if (ch == ')') {

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

postfix[k++] = s.top();

s.pop();

}

if (!s.empty()) {

s.pop();

}

} else if (getWeight(ch) == 0) {

postfix[k++] = ch;

} else {

while (!s.empty() && s.top() != '(' && getWeight(ch) <= getWeight(s.top())) {

postfix[k++] = s.top();

s.pop();

}

s.push(ch);

}

}

while (!s.empty()) {

postfix[k++] = s.top();

s.pop();

}

postfix[k] = '\0';

}

int evaluate(char postfix[]) {

stack<double> s;

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

char ch = postfix[i];

if (isdigit(ch)) {

s.push(ch - '0');

} else {

double val2 = s.top(); s.pop();

double val1 = s.top(); s.pop();

switch (ch) {

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

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

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

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

}

}

}

return s.top();

}

int main() {

char infix[100];

cout << "\nEnter infix operation: ";

cin >> infix;

int size = strlen(infix);

char postfix[size + 1];

infix2postfix(infix, postfix, size);

cout << "\nInfix Expression: " << infix;

cout << "\nPostfix Expression: " << postfix;

int result = evaluate(postfix);

cout << "\nEvaluation Result: " << result << endl;

return 0;

}

**OUTPUT:**

Enter infix operation: 4\*8+3-5

Infix Expression: 4\*8+3-5

Postfix Expression: 48\*3+5-

Evaluation Result: 30