**Assignment Number: 10**

**Subject: Data Structure and Algorithms**

**Name: Shrirang Mhalgi**

**Roll No.:222006**

**Class:S.E.**

**Division:B**

**Batch:B1**

**PROBLEM STATEMENT:-**

Implement C++ program for expression conversion as infix to postfix and its evaluation

using stack based on given conditions

i. Operands and operator, both must be single character.

ii. Input Postfix expression must be in a desired format.

iii. Only '+', '-', '\*' and '/ ' operators are expected

**CODE**

#include<iostream>

#include<stack>

#include<string>

using namespace std;

void postfixEvaluate(string s)

{

stack <char> st;

char t, op1, op2;

int i = 0, val, res;

t = s[i];

bool opnd = true; //true if operand, false if operator

while (t != '\0')

{

if (t >= '0' && t <= '9')

opnd = true;

else if (t == '+' || t == '-' || t == '\*' || t == '/')

opnd = false;

if (opnd)

{

val = t - 48;

st.push(val);

}

else if (!opnd)

{

op2 = st.top();

st.pop();

op1 = st.top();

st.pop();

switch (t)

{

case '+':

res = op1 + op2;

break;

case '-':

res = op1 - op2;

break;

case '\*':

res = op1 \* op2;

break;

case '/':

res = op1 / op2;

break;

}

st.push(res);

}

i += 1;

t = s[i];

}

cout << "The result is " << res << endl;

return;

}

int getweight(char ch)

{

switch (ch)

{

case '/':

case '\*':

return 2;

case '+':

case '-':

return 1;

default:

return 0;

}

}

void toPostfix(string infix, char postfix[])

{

stack <char> s;

int w, k;

k = 0;

unsigned int i = 0;

char ch;

while (i < infix.length())

{

ch = infix[i];

if (ch == '(')

{

s.push(ch);

i++;

continue;

}

if (ch == ')')

{

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

{

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

s.pop();

}

if (!s.empty())

{

s.pop();

}

i++;

continue;

}

w = getweight(ch);

if (w == 0)

{

postfix[k] = ch;

k++;

}

else

{

if (s.empty())

{

s.push(ch);

}

else

{

while (!s.empty() && s.top() != '(' && w <= getweight(s.top()))

{

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

s.pop();

}

s.push(ch);

}

}

i++;

}

while (!s.empty())

{

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

s.pop();

}

postfix[k] = 0;

cout << "The postfix expression is: ";

cout << postfix << endl;

postfixEvaluate(postfix);

return;

}

int main()

{

string infix;

char postfix[50];

int x = 1;

while (x == 1)

{

cout << "Enter the infix expression: ";

cin >> infix;

toPostfix(infix, postfix);

cout << endl << "Enter another expression? 1 or 0: ";

cin >> x;

cout << endl;

}

return 0;

}

**OUTPUT**

Enter the infix expression: (5+9+(5\*9))

The postfix expression is: 5 9 + 5 9 \* +

The result is 59