**INPUT CODE:**

#include <iostream>

using namespace std;

#define MAX 20

class stack {

int top, topeval;

char infix[MAX], postfix[MAX], stk[MAX];

float stkeval[MAX];

public:

stack();

void push(char);

void pusheval(float);

void read();

int IsEmpty();

int IsEmptyeval();

char pop();

float popeval();

void convert\_infix\_postfix();

int priority(char x);

float evaluatepostfix();

};

stack::stack() {

top = -1;

topeval = -1;

}

int stack::IsEmpty() {

if (top == -1)

return (-1);

else

return (1);

}

int stack::IsEmptyeval() {

if (topeval == -1)

return (-1);

else

return (1);

}

void stack::push(char temp) {

if (top >= MAX - 1)

cout << "Stack is Full";

else {

stk[++top] = temp;

}

}

void stack::pusheval(float temp) {

if (topeval >= MAX - 1)

cout << "Stack is Full";

else {

stkeval[++topeval] = temp;

}

}

char stack::pop() {

if (IsEmpty() == -1) {

return ('@');

} else {

return stk[top--];

}

}

float stack::popeval() {

if (IsEmptyeval() == -1) {

return (-999);

} else {

return stkeval[topeval--];

}

}

void stack::read() {

cout << "Enter infix expression: ";

cin >> infix;

}

void stack::convert\_infix\_postfix() {

int i, k = 0;

char sop;

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

if (infix[i] >= '0' && infix[i] <= '9')

postfix[k++] = infix[i];

else if (infix[i] == '(')

push(infix[i]);

else if (infix[i] == ')') {

while ((sop = pop()) != '(')

postfix[k++] = sop;

} else {

while (priority(infix[i]) <= priority(sop = pop())) {

postfix[k++] = sop;

if (top == -1)

break;

}

if (priority(infix[i]) > priority(sop)) {

push(sop);

}

push(infix[i]);

}

}

while (top != -1) {

sop = pop();

postfix[k++] = sop;

}

postfix[k] = '\0'; // Corrected to terminate the string properly

cout << endl << "The postfix is: " << postfix;

}

int stack::priority(char x) {

switch (x) {

case '@':

return -1;

case '(':

return 0;

case '+':

case '-':

return 1;

case '\*':

case '/':

return 2;

default:

return -1; // Handle any unknown operators

}

}

float stack::evaluatepostfix() {

float value, operand1, operand2, result;

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

if (postfix[i] >= '0' && postfix[i] <= '9') {

value = postfix[i] - '0';

pusheval(value);

} else {

operand2 = popeval();

operand1 = popeval();

switch (postfix[i]) {

case '+':

result = operand1 + operand2;

break;

case '\*':

result = operand1 \* operand2;

break;

case '-':

result = operand1 - operand2;

break;

case '/':

result = operand1 / operand2;

break;

}

pusheval(result);

}

}

return popeval();

}

int main() {

stack s;

float answer;

s.read();

s.convert\_infix\_postfix();

answer = s.evaluatepostfix();

cout << endl << "The answer is: " << answer;

return 0;

}

--------------------------------------------------------------------------------------------------------------------------------------

**OUTPUT:**

Enter infix expression: (3+4)\*5

The postfix is: 34+5\*

The answer is: 35