

**NAME:**

**UQBA GULZAR**

**REG NO:**

**2023-BSE-067**

**SECTION:**

**BSE IIIB**

**COURSE TITLE:**

**DSA Lab**

**Task 1 :**

Give answers to the following.

|  |  |
| --- | --- |
| 1. | **Convert (manually) the following expressions to postfix.**  **(A+B\*D)/(E-F)+G**  ABD\*+EF-/G+  **A\*(B+D)/E-F\*(G+H/K) :**  ABD+\*E/FGHK/+\*- |
| 2. | Evaluate the given Postfix expression and trace the contents of the Stack at each step using the standard evaluation algorithm.  **2 7 3 - / 2 1 5 + \* +**  **Symbol Stack Contents**  **2** [2]  **7**  [2,7]  **3** [2,7,3]  **-** 7 & 3 pop (7-3=4)-> push 4 onto stack [2,4]  **/** 2 & 4 pop (2/4=0.5)-> push 0.5 onto stack [0.5]  **2** [0.5,2]  **1**  [0.5,2,1]  **5** [0.5,2,1,5]  **+** 1 & 5 pop (1+5=6)-> push 6 onto stack [0.5,2,6] |

|  |  |
| --- | --- |
|  | **\*** 2 & 6 pop (2\*6=12)-> push 12 onto stack [0.5,12]  **+** 0.5 & 12 pop (0.5+12=12.5)-> push onto stack [12.5]  **Result : \_\_\_\_\_\_\_\_12.5\_\_\_\_\_\_\_\_\_\_\_** |
| 3. | Convert the following expression from infix to postfix and show the contents of Stack and the output expression at each step.  (A+B) \* C – D+F\*G   |  |  |  | | --- | --- | --- | | **symbol** | **Stack contents** | **Output expression** | | **(** | **(** |  | | **A** | **(** | **A** | | **+** | **( +** | **A** | | **B** | **( +** | **AB** | | **)** |  | **AB+** | | **\*** | **\*** | **AB+** | | **C** | **\*** | **AB+C** | | **-** | **-** | **AB+C\*** | | **D** | **-** | **AB+C\*D** | | **+** | **+** | **AB+C\*D-** | | **F** | **+** | **AB+C\*D-F** | | **\*** | **+\*** | **AB+C\*D-F** | | **G** | **+\*** | **AB+C\*D-FG** | | **end** |  | **AB+C\*D-FG\*+** | |

**Code Task # 01**

**Implement the algo to evaluate the postfix expression using a Stack and display the result. (For simplicitiy, assume single digit numbers in the expression.)**

**23+5\*6+**

**Note: Use existing stack class #include<stack>**

#include <iostream>

#include <stack>

using namespace std;

int evaluatePostfix(string expression) {

stack<int> s;

for (int i = 0; i < expression.length(); i++) {

char ch = expression[i];

if (ch >= '0' && ch <= '9') {

s.push(ch - '0');

}

else {

int val2 = s.top();

s.pop();

int val1 = s.top();

s.pop();

if (ch == '+') {

s.push(val1 + val2);

}

else if (ch == '-') {

s.push(val1 - val2);

}

else if (ch == '\*') {

s.push(val1 \* val2);

}

else if (ch == '/') {

s.push(val1 / val2);

}

}

}

return s.top();

}

int main() {

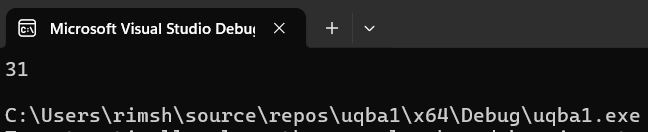
string postfixExpression = "23+5\*6+";

cout << evaluatePostfix(postfixExpression) << endl;

return 0;

}

**OUTPUT:**

****

**Code Task # 02**

**Implement the algo to covert the infix expression to postfix and display the result on screen Note: Use existing stack class #include<stack>**

#include <iostream>

#include <stack>

using namespace std;

int precedence(char op) {

if (op == '+' || op == '-') {

return 1;

}

if (op == '\*' || op == '/') {

return 2;

}

return 0;

}

string infixToPostfix(string expression) {

stack<char> s;

string postfix = "";

for (int i = 0; i < expression.length(); i++) {

char ch = expression[i];

if (ch >= '0' && ch <= '9') {

postfix += ch;

}

else if (ch == '(') {

s.push(ch);

}

else if (ch == ')') {

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

postfix += s.top();

s.pop();

}

s.pop();

}

else {

while (!s.empty() && precedence(s.top()) >= precedence(ch)) {

postfix += s.top();

s.pop();

}

s.push(ch);

}

}

while (!s.empty()) {

postfix += s.top();

s.pop();

}

return postfix;

}

int main() {

string infixExpression = "2+3\*(5-2)";

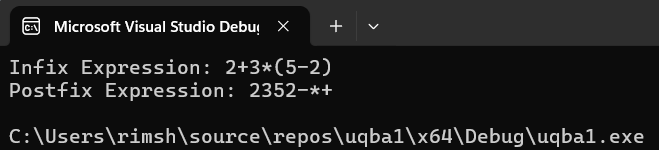
cout << "Infix Expression: " << infixExpression << endl;

cout << "Postfix Expression: " << infixToPostfix(infixExpression) << endl;

return 0;

}

**OUTPUT:**

****