DS LAB PROGRAM 2

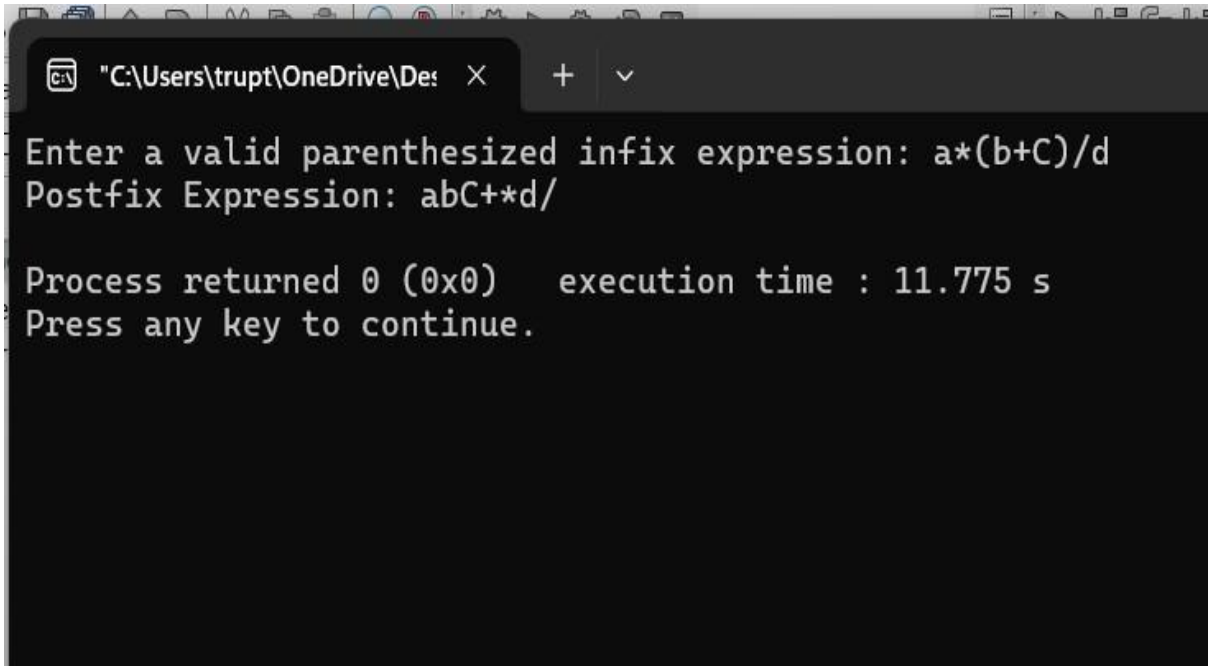
WAP to convert a given valid parenthesized infix arithmetic expression to postfix expression. The expression consists of single character operands and the binary operators + (plus), - (minus), * (multiply) and / (divide)

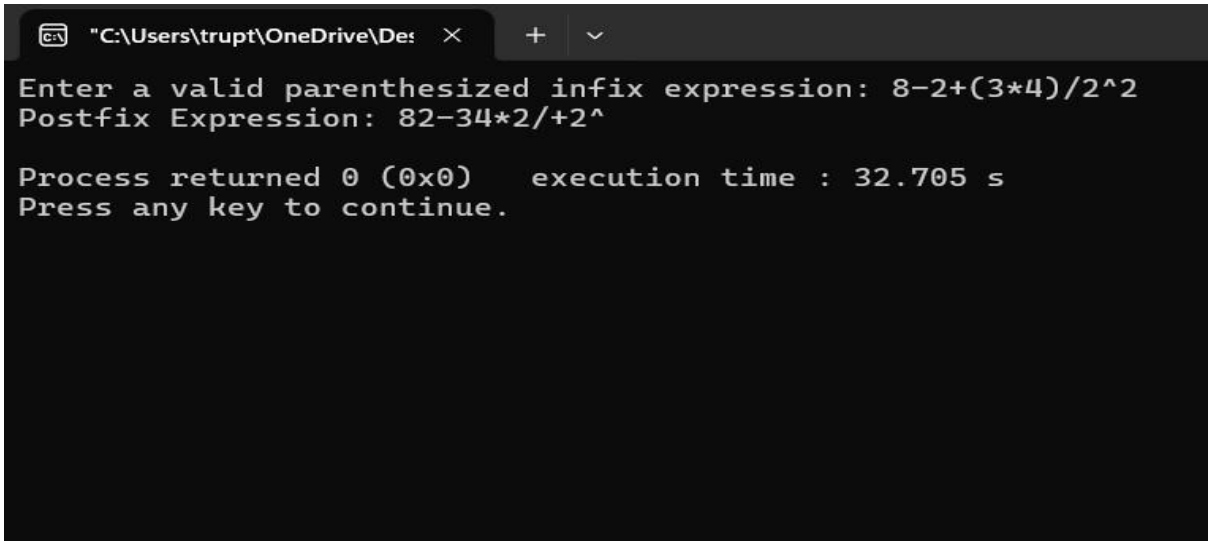
Start here | Labprogram2.c

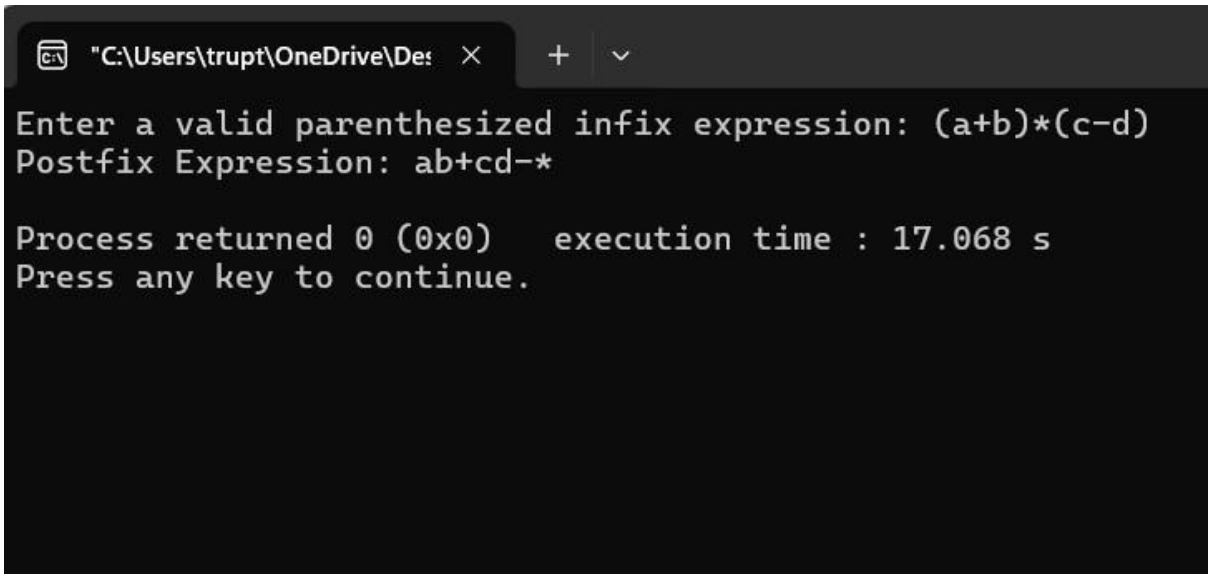
```c
#include <stdio.h>
#include <ctype.h> // for isalnum()
#include <string.h> // for strlen()
#define MAX 100
char stack[MAX];
int top = -1;
// Function to push into stack
void push(char c) {
if (top == MAX - 1) {
printf("Stack Overflow\n");
} else {
top = top + 1;
stack[top] = c;
}
}
// Function to pop from stack
char pop() {
char val;
if (top == -1) {
printf("Stack Underflow\n");
return -1;
} else {
val = stack[top];
top = top - 1;
return val;
}
}
// Function to peek stack top
char peek() {
if (top == -1)
return '\0';
return stack[top];
}
```

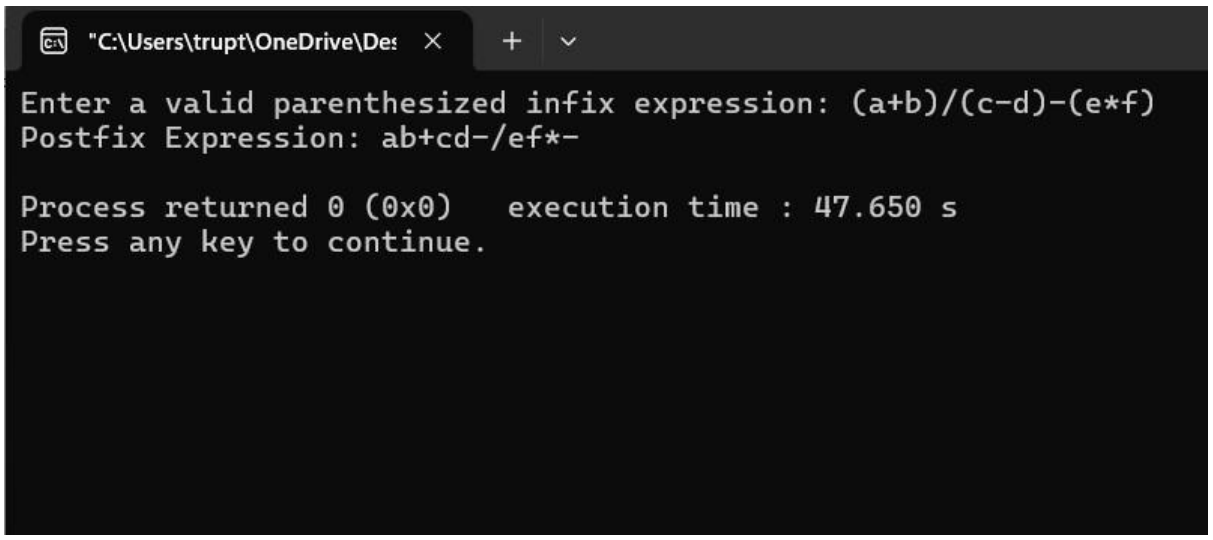
```
29  char peek() {
30  if (top == -1)
31  return '\0';
32  return stack[top];
33  }
34  // Function to check precedence of operators
35  int precedence(char c) {
36  if (c == '+' || c == '-') return 1;
37  if (c == '*' || c == '/') return 2;
38  return 0;
39  }
40  // Function to convert infix to postfix
41  void infixToPostfix(char infix[], char postfix[]) {
42  int i, k = 0;
43  char c;for (i = 0; infix[i] != '\0'; i++) {
44  c = infix[i];
45  // If operand, add to postfix expression
46  if (isalnum(c)) {
47  postfix[k] = c;
48  k = k + 1;
49  }
50  // If '(', push to stack
51  else if (c == '(') {
52  push(c);
53  }
54  // If ')', pop until '('
55  else if (c == ')') {
56  while (top != -1 && peek() != '(') {
57  postfix[k] = pop();
58  k = k + 1;
59  }
60  pop(); // remove '('
61  }
```

```
55      else if (c == ')') {
56      while (top != -1 && peek() != '(') {
57       postfix[k] = pop();
58       k = k + 1;
59      }
60       pop(); // remove '('
61      }
62       // If operator
63      else {
64      while (top != -1 && precedence(peek()) >= precedence(c)) {
65       postfix[k] = pop();
66       k = k + 1;
67      }
68       push(c);
69      }
70      }
71       // Pop all remaining operators
72      while (top != -1) {
73       postfix[k] = pop();
74       k = k + 1;
75      }
76       postfix[k] = '\0';
77      }
78      int main() {
79       char infix[MAX], postfix[MAX];
80       printf("Enter a valid parenthesized infix expression: ");
81       scanf("%s", infix);
82       infixToPostfix(infix, postfix);
83       printf("Postfix Expression: %s\n", postfix);
84       return 0;
85       }
86
```




"C:\Users\trupt\OneDrive\Des
Enter a valid parenthesized infix expression: 8-2+(3*4)/2^2
Postfix Expression: 82-34*2/+2^
Process returned 0 (0x0)   execution time : 32.705 s
Press any key to continue.


"C:\Users\trupt\OneDrive\Des
Enter a valid parenthesized infix expression: (a+b)*(c-d)
Postfix Expression: ab+cd-*
Process returned 0 (0x0)   execution time : 17.068 s
Press any key to continue.


"C:\Users\trupt\OneDrive\Des
Enter a valid parenthesized infix expression: (a+b)/(c-d)-(e*f)
Postfix Expression: ab+cd-/ef*-
Process returned 0 (0x0)   execution time : 47.650 s
Press any key to continue.