LAB PROGRAM 02

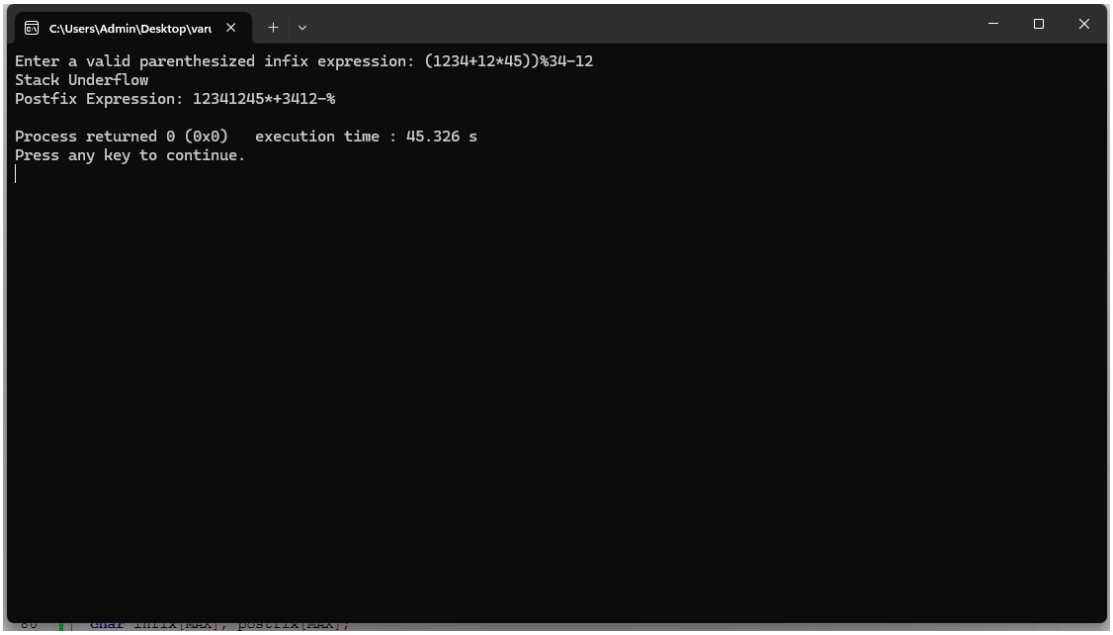
2. WAP to convert a given valid parenthesized infix arithmetic expression to postfix expression. The expression consists of single character operands and the binary operators + (plus), - (minus), * (multiply) and / (divide)

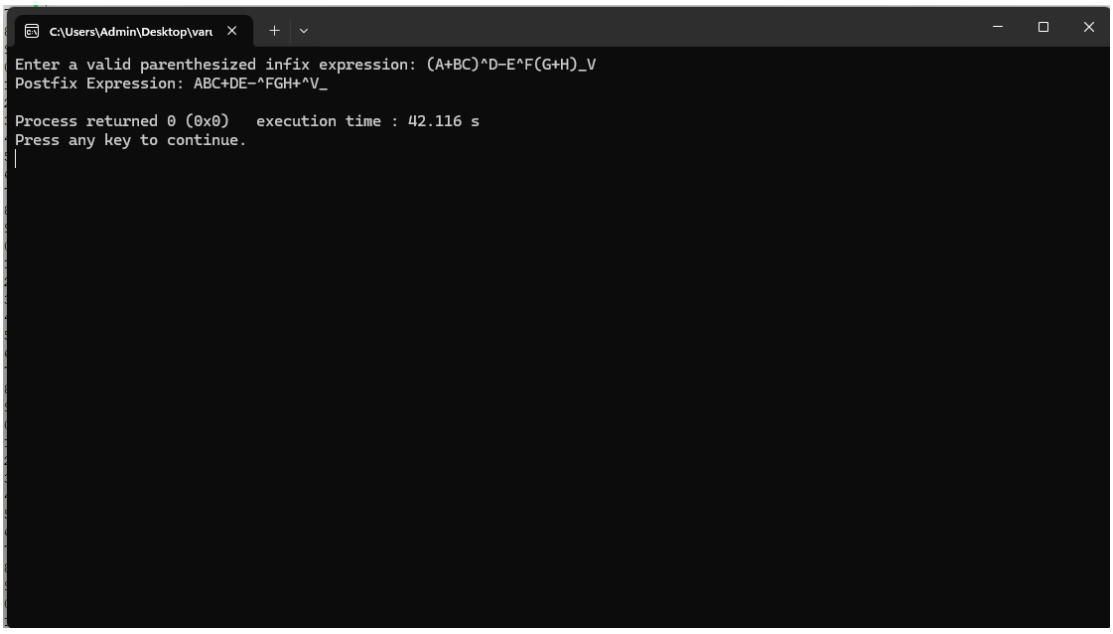
```
#include <stdio.h>
#include <ctype.h>   // for isalnum()
#include <string.h>  // for strlen()

#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];
}
```

```
// Function to check precedence of operators
int precedence(char c) {
   if (c == '+' || c == '-')
      return 1;
   if (c == '*' || c == '/')
      return 2;
   return 0;
}

// Function to convert infix to postfix
void infixToPostfix(char infix[], char postfix[]) {
   int i, k = 0;
   char c;

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

      // If operand, add to postfix expression
      if (isalnum(c)) {
         postfix[k] = c;
         k = k + 1;
      }

      // If '(', push to stack
      else if (c == '(') {
         push(c);
      }

      // If ')', pop until '('
      else if (c == ')') {
         while (top != -1 && peek() != '(') {
            postfix[k] = pop();
            k = k + 1;
         }
         pop(); // remove '('
      }

      // If operator
      else {
         while (top != -1 && precedence(peek()) >= precedence(c)) {
            postfix[k] = pop();
            k = k + 1;
         }
         push(c);
      }
   }
```

```
    // Pop all remaining operators
    while (top != -1) {
        postfix[k] = pop();
        k = k + 1;
    }

    postfix[k] = '\0';
}

int main() {
    char infix[MAX], postfix[MAX];

    printf("Enter a valid parenthesized infix expression: ");
    scanf("%s", infix);

    infixToPostfix(infix, postfix);

    printf("Postfix Expression: %s\n", postfix);

    return 0;
}
```

Output:


C:\Users\Admin\Desktop\var
Enter a valid parenthesized infix expression: (1234+12*45))%34-12
Stack Underflow
Postfix Expression: 12341245*+3412-%
Process returned 0 (0x0)   execution time : 45.326 s
Press any key to continue.


C:\Users\Admin\Desktop\var
Enter a valid parenthesized infix expression: (A+BC)^D-E^F(G+H)_V
Postfix Expression: ABC+DE-^FGH+^V_
Process returned 0 (0x0)   execution time : 42.116 s
Press any key to continue.