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).

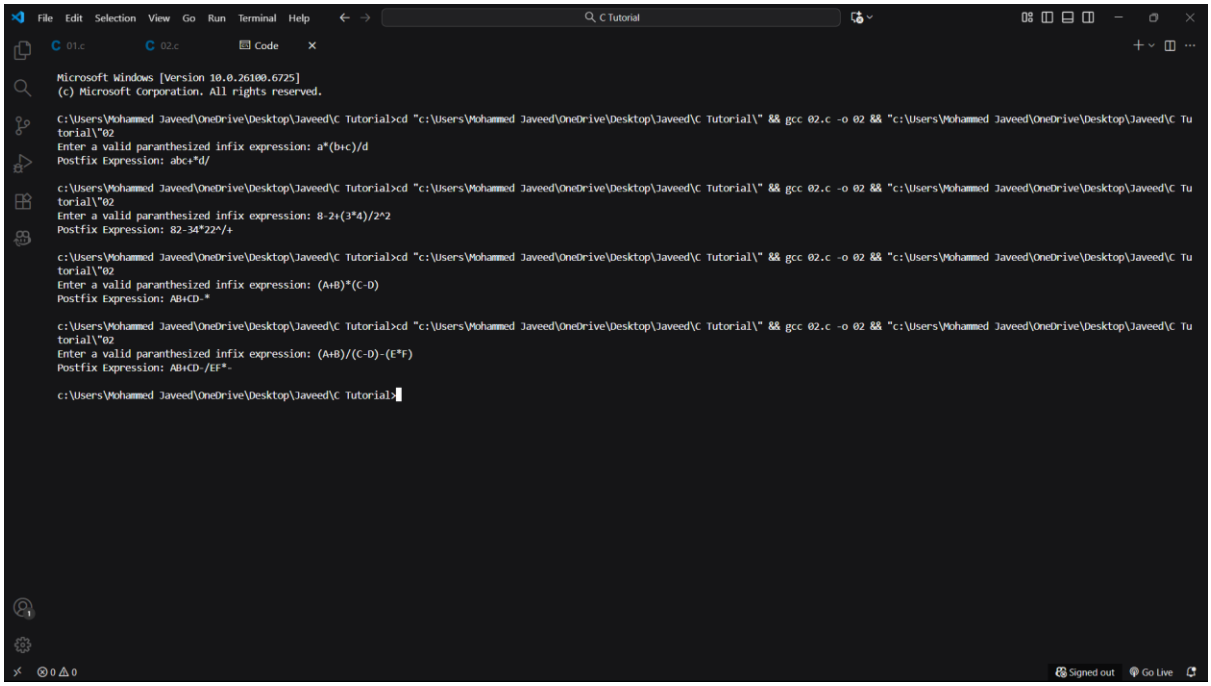
```
#include<stdio.h>
#include<ctype.h> //for isalnum()
#include<string.h> //for strlen()
#define MAX 100

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 += 1;
      stack[top] = c;
   }
}

//funtion to pop from stack
char pop()
{
   char val;
   if(top ==-1)
   {
      printf("Stack Underflow\n");
```

```
        return -1;

    }
    else{
        val = stack[top];
        top -= 1;
        return val;
    }
}

//function to get peek of stack
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;
    if(c == '^') return 3;
    return 0;
}

//funtion to convert infix to postfix
void infixToPosstfix(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+=1;
        }
        //if '(' ,pussh to stack
        else if(c == '(')
        {
            push(c);
        }
        //if ')', pop until '('
        else if(c == ')')
        {
            while(top != -1 && peek() != '(')
            {
                postfix[k] = pop();
                k+=1;
            }
            pop(); //remove '('
        }
        //if its a 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 += 1;
    }
    postfix[k] = '\0';

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

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

    infixToPosstfix(infix,postfix);

    printf("Postfix Expression: %s\n",postfix);
    return 0;
```

OUTPUT :


File Edit Selection View Go Run Terminal Help
C Tutorial
01.c
02.c
Code
Microsoft Windows [Version 10.0.26100.6725]
(c) Microsoft Corporation. All rights reserved.
C:\Users\Mohammed Javeed\OneDrive\Desktop\Javeed\C Tutorial>cd "c:\Users\Mohammed Javeed\OneDrive\Desktop\Javeed\C Tutorial\" && gcc 02.c -o 02 && "c:\Users\Mohammed Javeed\OneDrive\Desktop\Javeed\C Tutorial\"02
Enter a valid paranthesized infix expression: a*(b+c)/d
Postfix Expression: abc+*d/
c:\Users\Mohammed Javeed\OneDrive\Desktop\Javeed\C Tutorial>cd "c:\Users\Mohammed Javeed\OneDrive\Desktop\Javeed\C Tutorial\" && gcc 02.c -o 02 && "c:\Users\Mohammed Javeed\OneDrive\Desktop\Javeed\C Tutorial\"02
Enter a valid paranthesized infix expression: 8-2+(3*4)/2^2
Postfix Expression: 82-34*22^/+
c:\Users\Mohammed Javeed\OneDrive\Desktop\Javeed\C Tutorial>cd "c:\Users\Mohammed Javeed\OneDrive\Desktop\Javeed\C Tutorial\" && gcc 02.c -o 02 && "c:\Users\Mohammed Javeed\OneDrive\Desktop\Javeed\C Tutorial\"02
Enter a valid paranthesized infix expression: (A+B)*(C-D)
Postfix Expression: AB+CD-*
c:\Users\Mohammed Javeed\OneDrive\Desktop\Javeed\C Tutorial>cd "c:\Users\Mohammed Javeed\OneDrive\Desktop\Javeed\C Tutorial\" && gcc 02.c -o 02 && "c:\Users\Mohammed Javeed\OneDrive\Desktop\Javeed\C Tutorial\"02
Enter a valid paranthesized infix expression: (A+B)/(C-D)-(E*F)
Postfix Expression: AB+CD-/EF*-
c:\Users\Mohammed Javeed\OneDrive\Desktop\Javeed\C Tutorial>
0 0
Signed out
Go Live