Lab-2

Aim: Convert a given infix expression to Postfix and display it. Expression consists of binary operators (+, -, *, /)

Code:

```
#include <stdio.h>
#include <ctype.h>
#include <string.h>

define MAX
~~stack~~ infix [MAX], postfix [MAX]
top = -1

function precedence (op):
    If op is '+' or '-'
        return 1
    else if op is '*' or '/'
        return 2
    else if op is '^'
        return 3
    else if op is '('
        return 0

function associativity (op):
    If op is '^'
        return 1 (Right to left)
    else
        return 0 (Left to right)

function push (ch):
    If top == MAX-1
        print 'Stack overflow'
        return
    stack [++top] = ch
```

```
function pop():
    if top == -1
        print 'stack underflow'
    return stack[top--]
function peek():
    if top == -1
        print 'stack is empty'
    return stack[top]
function infixToPostfix (char infix[], char postfix[]):
    ~~for (int i=0; i~~
    int i, p=0
    char c
    for (int i=0; i<strlen(infix[]); i++)
        c = infix[i]
        if (~~infix~~ c is operand)
            push into postfix[]
        else if (c == '(')
            push into stack
        else if (c == ')')
            while (peek() != '(')
                ~~postfix[k]~~
                postfix[p++] = pop()
        else
            while (top != 1 and precedence(peek()) or
                (precedence(c) or precedence(peek()) ==
                precedence(c) and associativity == 0)
            {
                postfix[p++] = pop()
            }
        pop()
```

```
function main ( ):
    input infix [ ]
    * infixToPostfix [infix, postfix]
    print [postfix ( ])
    return
```

Output:

I. Enter a valid parenthecized infix expression: (a+b*c)
Postfix expression: abc *+

II. Enter a valid parenthecized infix expression: ((a+b)*c)
Postfix expression: ab+c*

III. Enter a valid parenthecized infix expression: (a^(b^c))
Postfix expression: abc^^

IV. Enter a valid parenthecized infix expression: ((a+b)*(c-d))
Postfix expression: ab+cd-*

V. Enter a valid parenthecized infix expression: (a+b*c-d/e)
Postfix expression: abc*+de/-

```
Enter a valid parenthesized infix expression : (a+b*c)
Postfix expression : abc*+
Process returned 0 (0x0)   execution time : 3.245 s
```

```
Enter a valid parenthesized infix expression : ((a+b)*c)
Postfix expression : ab+c*
Process returned 0 (0x0)   execution time : 9.120 s
```

```
Enter a valid parenthesized infix expression : (a^(b^c))
Postfix expression : abc^^
Process returned 0 (0x0)   execution time : 2.367 s
```

```
Enter a valid parenthesized infix expression : ((a+b)*(c-d))
Postfix expression : ab+cd-*
Process returned 0 (0x0)   execution time : 2.379 s
```

```
Enter a valid parenthesized infix expression : (a+b*c-d/e)
Postfix expression : abc*+de/-
Process returned 0 (0x0)   execution time : 2.220 s
```