LAB-2 PROGRAM

WAP to convert a given valid expression 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 <string.h>
#include <stdlib.h>

int F (char symbol)
{
 switch (symbol)
{
 case '+':
 case '-': return 2;
 case '*':
 case '/': return 4;
 case '^':
case '$': return 5;
case '(' : return 0;
case '#' : return -1;
 default  : return 8;
 }
}
 int G (char symbol)
{
 switch (symbol)
{
  case '+':
 case '-': return 1;
```

```
        case '*':
        case '/' : return 3;
        case '^':
        case '$': return 6;
        case '(' : return 9;
        case ')' : return 0;
        default : return 7;
    }
}
void infix-postfix (char infix[], char postfix[])
{
    int top, i, j;
    char s[30], symbol;
    top = -1;
    s[++top] = '#';
    j = 0;
    for (i=0; i<strlen (infix); i++)
    {
        symbol = infix[i];
        while (F(s[top]) > G(symbol))
        {
            postfix[j] = s[top--];
            j++;
        }
        if (F(s[top]) != G(symbol))
            s[++top] = symbol;
        else
            top--;
    }
    while(s[top] != '#')
    {   postfix[j++] = s[top--];
    }
```

```
postfix[j] = '\0';
}
void main()
{
  char infix[20], postfix[20];
  int c1=0, c2=0;
 printf("Enter the valid infix expression:\n");
 scanf("%s", infix);
 for( int k=0; k<strlen(infix);k++)
{
   if (infix[k] == '(')
   c1++;
   else if (infix[k] == ')')
   c2++;
   else
   continue;
}
  if (c1!= c2)
{  printf("Invalid infix expression!");
   exit(0);
}
 infix-postfix (infix, postfix);
printf(" The postfix expression is :\n");
 printf ("%s\n", postfix);
}
```