Lab-3

Write a program to convert a given valid parenthesized infix arithmetic expression to postfix expression. The expression consists of single character operands and the binary operators +, -, *, /.

```
#include <stdio.h>
#include <string.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'; }

int main()
{ char infix[20];
  char postfix[20];
  printf("Enter valid infix\n");
  scanf("%s", infix);
  infix-postfix(infix, postfix);
  printf("The postfix express is %s\n", postfix);
  return 0; }
```

Output sample:

Enter valid infix:

(a+b)*(c-d)*(e/f)

The postfix expression is ab+cd-*ef/*