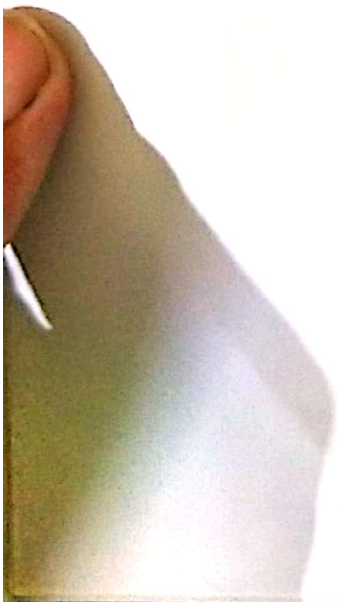
LAB 2

→ WAP to convert a given valid parenthesized infix arithmetic expression to postfix expression. The expression consists of single character operands & the binary operators +, −, * & /.

```
#include <stdio.h>
#include <string.h>
#include <process.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];
   char postfix[20];
  printf("enter the valid infix exp^n \n");
   scanf("%s", infix);
  infix_postfix(infix, postfix);
  printf("the postfix exp^n is \n");
  printf("%s\n", postfix);
   getch();
 }
```



art here X | ds lab 2.c X

```
#include<stdio.h>
#include<string.h>
#include<process.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))
```

gs & others

HUN\practice\ds lab 2.c

C/C++

here X ds lab 2.c X

```
29          case '(':return 9;
30          case ')':return 0;
31          default :return 7 ;
32        }
33    }
34    void infix_postfix(char infix[],char postfix[])
35     {
36          int top,i,j;
37          char s[30],symbol;
38          top=-1;
39          s[++top]='#';
40          j=0;
41          for(i=0;i<strlen(infix);i++)
42          {
43              symbol=infix[i];
44              while(F(s[top])>G(symbol))
45              {
46                  postfix[j]=s[top--];
47                  j++;
48              }
49              if(F(s[top])!=G(symbol))
50                s[++top]=symbol;
51              else
52                top--;
53          }
54           while (s[top]!='#')
55            {
56                postfix [j++]=s[top--];
57            }
58         postfix[j]='\0';
59
60     }
61    void main()
62     {
63          char infix [20];
64          char postfix[20];
65          printf("Enter the valid infix expression:\n");
66          scanf("%s",infix);
67          infix_postfix(infix,postfix);
68          printf("postfix expression is :\n");
69          printf("%s",postfix);
70
71     }
72
```

thers

practice\ds lab 2.c C/C++