

```

1  #include<stdio.h>
2  #include<string.h>
3  int F(char symbol){
4      switch(symbol)
5      {
6          case '+':
7          case '-':return 2;
8          case '*':
9          case '/':return 4;
10         case '^':
11         case '$':return 5;
12         case '(':return 0;
13         case '#':return -1;
14         default:return 8;
15     }
16 }
17 int G(char symbol){
18     switch(symbol)
19     {
20         case '+':
21         case '-':return 1;
22         case '*':
23         case '/':return 3;
24         case '^':
25         case '$':return 6;
26         case '(':return 9;
27         case ')':return 0;
28         default:return 7;
29     }
30 }
31 void infix_postfix(char infix[],char
postfix[]){
32     int top,i,j;
33     char s[30],symbol;
34     top=-1;
35     s[++top]='#';

```

```

33     char s[30],symbol;
34     top=-1;
35     s[++top]='#';
36     j=0;
37     for(i=0;i<strlen(infix);i++){
38         symbol=infix[i];
39         while(F(s[top])>G(symbol)){
40             postfix[j]=s[top--];
41             j++;
42         }
43         if(F(s[top])!=G(symbol)){
44             s[++top]=symbol;
45         }
46         else
47             top--;
48     }
49     while(s[top]!='#'){
50         postfix[j++]=s[top--];
51     }postfix[j]='\0';
52 }
53 int main()
54 {
55     char infix[20],postfix[20];
56     printf("Enter the valid infix
expression\n");
57     scanf("%s",infix);
58     infix_postfix(infix,postfix);
59     printf("The postfix expression is \n");
60     printf("%s",postfix);
61 }

```

Enter the valid infix expression

$((a+b)*c-(d-e))^{(f+g)}$

The postfix expression is

$ab+c*de--fg+^$

[Program finished]

Enter the valid infix expression

$((a+(b-c)*d)^e+f)$

The postfix expression is

$abc-d*+e^f+$

[Program finished]

Enter the valid infix expression

$a^b * c - d + e / f / (g + h)$

[Program finished]The postfix expression is

$ab^c * d - ef / gh + / +$

LAB-2

WAP to convert infix expression to postfix expression.

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

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

```
#include <process.h>
```

```
int F(char Symbol)
```

```
{
```

```
    switch(symbol)
```

```
{
```

```
    case '+':
```

```
    case '-': return P;
```

```
    case '*':
```

```
    case '/': return H;
```

```
    case '^':
```

```
    case '$': return S;
```

```
    case '<': return O;
```

```
    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_to_postfix(char infix[3], char postfix[3])
{
    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]) > R(symbol))
        {
            postfix[j] = s[top--];
            j++;
        }
        if (F(s[top]) != R(symbol))
            s[++top] = symbol;
        else
            top--;
    }
    while (s[top] != '#')
    {
        postfix[j++] = s[top--];
    }
    postfix[j] = '\0';
}

void main ()
{
    char infix[20], postfix[20];
    printf("Enter valid infix expression\n");
    scanf("%s", infix);
}

```

```
cn_postfix (infix, postfix);  
printf ("The postfix expression is : \n");  
printf ("%s \n", @postfix);  
return 0;  
}
```