

```
1
2#include<stdio.h>
3#include<stdlib.h>
4#include<ctype.h>
5#include<string.h>
6
7#define SIZE 100
8char stack[SIZE];
9int top = -1;
10
11void push(char item){
12    if(top >= SIZE-1)
13    {
14        printf("\n Stack Overflow.");
15    }
16    else
17    {
18        top = top+1;
19        stack[top] = item;
20    }
21}
22
23
24char pop(){
25    char item ;
26    if(top <0)
27    {
28        printf("stack under flow: invalid infix expression");
29        getchar();
30        exit(1);
31    }
32    else
33    {
34        item = stack[top];
35        top = top-1;
36        return(item);
37    }
38}
39
40
41int is_operator(char symbol){
42    if(symbol == '^' || symbol == '*' || symbol == '/' || symbol == '+' || symbol == '-')
43    {
44        return 1;
45    }
46}
```

C Tab Width: 8 Ln 2, Col 15 INS

```
39
40
41int is_operator(char symbol){
42    if(symbol == '^' || symbol == '*' || symbol == '/' || symbol == '+' || symbol == '-')
43    {
44        return 1;
45    }
46    else
47    {
48        return 0;
49    }
50}
51
52
53int precedence(char symbol){
54    if(symbol == '^')
55    {
56        return(3);
57    }
58    else if(symbol == '*' || symbol == '/')
59    {
60        return(2);
61    }
62    else if(symbol == '+' || symbol == '-')
63    {
64        return(1);
65    }
66    else
67    {
68        return(0);
69    }
70}
71
72void InfixToPostfix(char infix_exp[], char postfix_exp[]){
73    int i, j;
74    char item;
75    char x;
76    push('(');
77    strcat(infix_exp, "(");
78    i=0;
79    j=0;
80    item=infix_exp[i];
81
82    while(item != '\0'){
83        if(item == '(')
```

C Tab Width: 8 Ln 2, Col 4 INS

```

68     return(0);
69 }
70 }
71
72 void InfixToPostfix(char infix_exp[], char postfix_exp[]){
73     int i, j;
74     char item;
75     char x;
76     push('(');
77     strcat(infix_exp, "(");
78     i=0;
79     j=0;
80     item=infix_exp[i];
81
82     while(item != '\0'){
83         if(item == '(')
84         {
85             push(item);
86         }
87         else if( isdigit(item) || isalpha(item))
88         {
89             postfix_exp[j] = item;
90             j++;
91         }
92         else if(is_operator(item) == 1)
93         {
94             x=pop();
95             while(is_operator(x) == 1 && precedence(x) >= precedence(item))
96             {
97                 postfix_exp[j] = x;
98                 j++;
99                 x = pop();
100             }
101             push(x);
102             push(item);
103         }
104         else if(item == ')')
105         {
106             x = pop();
107             while(x != '(')
108             {
109                 postfix_exp[j] = x;
110                 j++;
111                 x = pop();
112             }

```

C Tab Width: 8 Ln 2, Col 4 INS

```

102         push(item);
103     }
104     else if(item == ')')
105     {
106         x = pop();
107         while(x != '(')
108         {
109             postfix_exp[j] = x;
110             j++;
111             x = pop();
112         }
113     }
114     else
115     {
116         printf("\nInvalid Infix Expression.\n");
117         getchar();
118         exit(1);
119     }
120     i++;
121     item = infix_exp[i];
122 }
123 if(top>0)
124 {
125     printf("\nInvalid Infix Expression.\n");
126     getchar();
127     exit(1);
128 }
129 postfix_exp[j] = '\0';
130 }
131
132 }
133
134 int main(){
135     char infix[SIZE], postfix[SIZE];
136
137     printf("\n Enter Infix expression : ");
138     scanf("%s",infix);
139
140     InfixToPostfix(infix,postfix);
141     printf(" Postfix Expression: ");
142     printf("%s",postfix);
143
144     return 0;
145 }
146

```

C Tab Width: 8 Ln 2, Col 4 INS

```
dash: ./a.out: No such file or directory
dl0409@noname:~$ gcc meet3.c
dl0409@noname:~$ ./a.out

Enter Infix expression : a+b
Postfix Expression: ab+dl0409@noname:~$ ./a.out

Enter Infix expression : (a+b(c*d))
Postfix Expression: abcd*+dl0409@noname:~$ █
```