

```
1  #include<stdio.h>
2  #include<stdlib.h>
3  #include<ctype.h>
4  #include<string.h>
5
6  #define SIZE 25
7
8  char stack[SIZE];
9  int top = -1;
10
11 void push(char item)
12 {
13     if(top >= SIZE-1)
14     {
15         printf("\nStack Overflow.");
16     }
17     else
18     {
19         top = top+1;
20         stack[top] = item;
21     }
22 }
23
24 char pop()
25 {
26     char item ;
27
28     if(top == -1)
29         return -1;
30     else
31     {
32         item = stack[top];
33         top = top-1;
34         return(item);
35     }
```

```

36 }
37
38
39 int check_operator(char symbol)
40 {
41     if(symbol == '^' || symbol == '*' || symbol == '/' || symbol == '+' || symbol == '-')
42         return 1;
43     else
44         return 0;
45 }
46
47 int precedence(char symbol)
48 {
49     if(symbol == '^')
50         return(3);
51     else if(symbol == '*' || symbol == '/')
52         return(2);
53     else if(symbol == '+' || symbol == '-')
54         return(1);
55     else
56         return(0);
57 }
58
59 void InfixToPostfix(char infix_exp[], char postfix_exp[])
60 {
61     int i=0, j=0;
62     char item;
63     char x;
64
65     push('(');
66     strcat(infix_exp, ")");
67
68     item=infix_exp[i];
69
70     while(item != '\0')

```

```

71 {
72     if(item == '(')
73     {
74         push(item);
75     }
76     else if( isdigit(item) || isalpha(item))
77     {
78         postfix_exp[j] = item;
79         j++;
80     }
81     else if(check_operator(item) == 1)
82     {
83         x=pop();
84         while(check_operator(x) == 1 && precedence(x)>= precedence(item))
85         {
86             postfix_exp[j] = x;
87             j++;
88             x = pop();
89         }
90         push(x);
91         push(item);
92     }
93     else if(item == ')')
94     {
95         x = pop();
96         while(x != '(')
97         {
98             postfix_exp[j] = x;
99             j++;
100             x = pop();
101         }
102     }
103     else
104     {
105         printf("\nInvalid infix Expression.\n");

```

```

106         getchar();
107         exit(1);
108     }
109     i++;
110     item = infix_exp[i];
111 }
112 if(top>0)
113 {
114     printf("\nInvalid infix Expression.\n");
115     getchar();
116     exit(1);
117 }
118 if(top>0)
119 {
120     printf("\nInvalid infix Expression.\n");
121     getchar();
122     exit(1);
123 }
124
125 postfix_exp[j] = '\0';
126 }
127
128
129
130 int main()
131 {
132     char infix[SIZE], postfix[SIZE];
133     printf("\nEnter Infix expression : ");
134     gets(infix);
135
136     InfixToPostfix(infix,postfix);
137     printf("Postfix Expression: ");
138     puts(postfix);
139
140     return 0;

```