

```

1 #include<stdio.h>
2 #include<stdlib.h>
3 #include<ctype.h>
4 #include<string.h>
5
6 #define SIZE 100
7 char stack[SIZE];
8 int top = -1;
9
10 /* == push operation == */
11 void push(char item)
12 {
13     if(top >= SIZE-1)
14     {
15         printf("\n Stack Overflow.");
16     }
17     else
18     {
19         top = top+1;
20         stack[top] = item;
21     }
22 }
23
24 /* == pop operation == */
25 char pop()
26 {
27     char item ;
28
29     if(top <0)
30     {
31         printf("stack under flow: invalid infix expression");
32         getchar();
33         /* underflow may occur for invalid expression */
34         /* where ( and ) are not matched */
35         exit(1);
36     }
37     else
38     {
39         item = stack[top];
40         top = top-1;
41         return(item);
42     }
43 }
44
45 /* == define function that is used to determine whether any symbol is operator or not */
46
47 int is_operator(char symbol)
48 {
49     if(symbol == '^' || symbol == '-' || symbol == '/' || symbol == '*' || symbol == '%')
50     {
51         return 1;
52     }
53     else
54     {
55         return 0;
56     }
57 }
58
59 /* == define function that is used to assign precedence to operator. */
60
61 int precedence(char symbol)
62 {
63     if(symbol == '^')
64     {
65         return(3);
66     }
67     else if(symbol == '*' || symbol == '/')
68     {
69         return(2);
70     }
71     else if(symbol == '+' || symbol == '-')
72     {
73         return(1);
74     }
75     else
76     {
77         return(0);
78     }
79 }
80
81 void InfixToPostfix(char infix_exp[], char postfix_exp[])
82 {
83     int i, j;
84     char item;
85     char x;
86
87     push('('); /* push '(' onto stack */
88     strcat(infix_exp, " "); /* add ' ' to infix expression */
89
90     i=0;
91     j=0;
92     item=infix_exp[i];
93
94     while(item != '\0')
95     {
96         if(item == '(')
97         {
98             push(item);
99         }
100         else if( isdigit(item) || isalpha(item) )
101         {
102             postfix_exp[j] = item; /* add operand symbol to postfix expr */
103             j++;
104         }
105         else if(is_operator(item) == 1) /* means symbol is operator */
106         {
107             x=pop();
108             while(is_operator(x) == 1 && precedence(x) >= precedence(item))
109             {
110                 postfix_exp[j] = x; /* so pop all higher precedence operator and */
111                 j++;
112                 x = pop(); /* add them to postfix expression */
113             }
114             push(x);
115             push(item); /* push current operator symbol onto stack */
116         }
117         else if(item == ')') /* if current symbol is ')' then */
118         {
119             x = pop(); /* pop and keep popping until */
120             while(x != '(') /* '(' encountered */
121             {
122                 postfix_exp[j] = x;
123                 j++;
124                 x = pop();
125             }
126         }
127         else
128         {
129             /* if current symbol is neither operand nor '(' nor ')' and nor operator */
130             printf("\nInvalid infix Expression.\n");
131             getchar();
132             exit(1);
133         }
134         i++;
135     }
136
137     item = infix_exp[i];
138 }
139
140 if(top >0)
141 {
142     printf("\nInvalid infix Expression.\n");
143     getchar();
144     exit(1);
145 }
146
147 postfix_exp[j] = '\0'; /* add sentinel else puts() function */
148 /* will print entire postfix[] array upto SIZE */
149
150 }
151
152 /* == main function begins == */
153 int main()
154 {
155     char infix[SIZE], postfix[SIZE];
156
157     printf("\n Enter Infix expression : ");
158     gets (infix);
159
160     InfixToPostfix(infix,postfix);
161     printf(" Postfix Expression: ");
162     puts(postfix);
163
164     return 0;
165 }
166

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

● **student@dl405-HP-ProDesk-400-G7-Microtower-PC:~/Desktop/neel\$./a.out**

Enter Infix expression : a+b-c*d

Postfix Expression: ab+cd*-

● **student@dl405-HP-ProDesk-400-G7-Microtower-PC:~/Desktop/neel\$./a.out**

Enter Infix expression : 1+2-3

Postfix Expression: 12+3-

○ **student@dl405-HP-ProDesk-400-G7-Microtower-PC:~/Desktop/neel\$** █