

DSA_College\infix_to_postfix.cpp

```
1  #include<iostream>
2  #include<stack>
3  #include<string>
4  using namespace std ;
5
6  // Function to check the precedence of operators
7  int precedence(char op)
8  {
9      if(op == '+' || op == '-')
10         return 1 ;
11
12         if(op == '*' || op == '/')
13             return 2 ;
14
15         if(op == '^')
16             return 3;
17
18         return 0; // for non-operators
19     }
20
21     bool isOperand(char x)
22     {
23         return ((x >= 'a' && x <= 'z') || (x >= 'A' && x <= 'Z')) ;
24     }
25
26     string infixToPostfix(const string &infix)
27     {
28         stack<char> st ;
29         string postfix = "" ;
30         for(int i=0; i<infix.length(); i++)
31         {
32             char c = infix[i] ;
33
34             if (isspace(c))
35                 continue;
36
37             // if the scanned character is an operand, add to o/p string
38             if(isOperand(c))
39             {
40                 postfix += c ;
41             }
42
43             // if the scanned character is '(' , add to stack
44             else if(c == '(')
45             {
46                 st.push('(') ;
47             }
48
49             // if the scanned character is ')', pop and add to o/p string
50             // from the stack until an '(' is found
51             else if(c == ')')
```

```
52     {
53         while(st.top() != '(')
54         {
55             postfix += st.top() ;
56             st.pop() ;
57         }
58         st.pop() ;
59     }
60
61     // if an operator is scanned, add to the stack
62     else{
63         while(st.empty() != 1 && precedence(c) <= precedence(st.top()))
64         {
65             postfix += st.top() ;
66             st.pop() ;
67         }
68         st.push(c) ;
69     }
70
71 }
72
73 // pop all the remaining elements from stack
74 while(!st.empty())
75 {
76     postfix += st.top() ;
77     st.pop() ;
78 }
79
80 return postfix ;
81 }
82
83 int main()
84 {
85     string infix = "((a+(b*c))-d)" ;
86     cout << "Infix Expression: " << infix << endl ;
87     cout << "Postfix Expression: " << infixToPostfix(infix)<< endl ;
88     return 0;
89 }
90
```