

② WAP to convert a given valid parenthesized infix arithmetic expression to postfix expression.
The expression consists of single character
operands & the binary operators.

→

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

```
int f(char symbol)
```

```
{
```

```
switch (symbol)
```

```
{
```

```
case '+':
```

```
case '-': return 2;
```

```
case '*':
```

```
case '/': return 4;
```

```
case '^':
```

```
case '&': return 5;
```

```
case '(': return 0;
```

```
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 - postfix (char infix[], char postfix[])

{

int top, i, j;

char S[30], symbol;

top = -1;

S[++top] = '#';

j = 0;

```
for (q=0; q < strlen(infix); q++)
```

```
{
```

```
symbol = infix[q];
```

```
while (F(s[top]) > G(symbol))
```

```
{
```

```
postfix[j] = s[top--];
```

```
j++;
```

```
}
```

```
if (F(s[top]) != G(symbol))
```

```
s[++top] = symbol;
```

```
else
```

```
top--;
```

```
}
```

```
while (s[top] != '#')
```

```
{
```

```
postfix[j++] = s[top--];
```

```
}
```

```
postfix[j] = '\0';
```

```
}
```

```
main()
```

```
{
```

```
char infix[20];
```

```
char postfix[20];
```

```
printf ("Enter the valid infix expression ");
```

```
scanf ("%s", infix);
```

enfix - postfix (infix, postfix);
printf ("The postfix expression is \n");
printf ("%s \n", postfix);
}

Output

Enter the valid infix expression ~~the postfix expression~~

~~A B C D E F - E R P~~

$(A + B)^* ((-D) / E) / F$

The postfix expression is

abc - d * +

Code, Compile & Run | CodeChef

codechef.com/ide

Apps Gmail YouTube Maps

C++14 (GCC 6.3)

```
1 #include<stdio.h>
2 #include<string.h>
3 #include<stdlib.h>
4
5 int F(char symbol)
6 {
7     switch(symbol)
8     {
9         case '+':
10        case '-':return 2;
11        case '^':
12        case '/':return 4;
13        case '*':
14        case '$':return 5;
15        case '(':return 0;
16        case ')':return -1;
17        default:return 8;
18    }
19 }
20
21 int G(char symbol)
22 {
23     switch(symbol)
24     {
25         case '+':
26        case '-':return 1;
27        case '^':
28        case '/':return 3;
29        case '*':
30        case '$':return 6;
31        case '(':return 9;
32        case ')':return 0;
33        default:return 7;
34    }
35 }
36
37 void infix_postfix(char infix[],char postfix[])
```

We use cookies to improve your experience and for analytical purposes.
Read our [Privacy Policy](#) and [Terms](#) to know more. You consent to our cookies if you continue to use our website.

Okay



Code, Compile & Run | CodeChef

codechef.com/ide

Apps Gmail YouTube Maps

C++14 (GCC 6.3)

```
42 s[++top]='#';
43 j=0;
44
45 for(i=0;i<strlen(infix);i++)
46 {
47     symbol=infix[i];
48     while(F(s[top])>G(symbol))
49     {
50         postfix[j]=s[top--];
51         j++;
52     }
53
54     if(F(s[top])!=G(symbol))
55     s[++top]=symbol;
56     else
57     top--;
58 }
59
60 while(s[top]!='#')
61 {
62     postfix[j++]=s[top--];
63 }
64 postfix[j]='\0';
65
66
67
68
69 main()
70 {
71     char infix[20];
72     char postfix[20];
73     printf("Enter the valid infix expression \n");
74     scanf("%s",infix);
75     infix_postfix(infix , postfix );
76     printf("The postfix expression is \n");
77     printf("%s\n",postfix);
78 }
```

We use cookies to improve your experience and for analytical purposes.
Read our [Privacy Policy](#) and [Terms](#) to know more. You consent to our cookies if you continue to use our website.

Okay



03:12 PM
05-10-2020

Code, Compile & Run | CodeChef

codechef.com/ide

Apps Gmail YouTube Maps

C++14 (GCC 6.3)

```
33     default:return '/';
34 }
35 }
36
37 void infix_postfix(char infix[],char postfix[])
38 {
39     int top,i,j;
40     char s[30],symbol;
41     top=-1;
42     s[++top]='#';
43     j=0;
44
45     for(i=0;i<strlen(infix);i++)
46     {
47         symbol=infix[i];
48         while(F(s[top])>G(symbol))
49     {
50             postfix[j]=s[top--];
51             j++;
52         }
53
54
55         if(F(s[top])!=G(symbol))
56             s[++top]=symbol;
57         else
58             top--;
59     }
60
61     while(s[top]!='#')
62     {
63         postfix[j++]=s[top--];
64     }
65     postfix[j]='\0';
66 }
67
68
69 main()
```

We use cookies to improve your experience and for analytical purposes.
Read our [Privacy Policy](#) and [Terms](#) to know more. You consent to our cookies if you continue to use our website.

Okay



Code, Compile & Run | CodeChef

codechef.com/ide

Status Successfully executed Date 2020-10-05 09:35:04 Time 0 sec Mem 15.232 kB

Input
(a+(b-c)*d)

Output
Enter the valid infix expression
The postfix expression is
abc-d*+

We use cookies to improve your experience and for analytical purposes.
Read our [Privacy Policy](#) and [Terms](#) to know more. You consent to our cookies if you continue to use our website.

Okay

03:12 PM
05-10-2020