

LAB PROGRAM 2:-

```
#include<stdio.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(i=0;i<strlen(infix);i++)  
{  
symbol=infix[i];  
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';
```

```
}
```

```
void main()
```

```
{
```

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

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

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

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

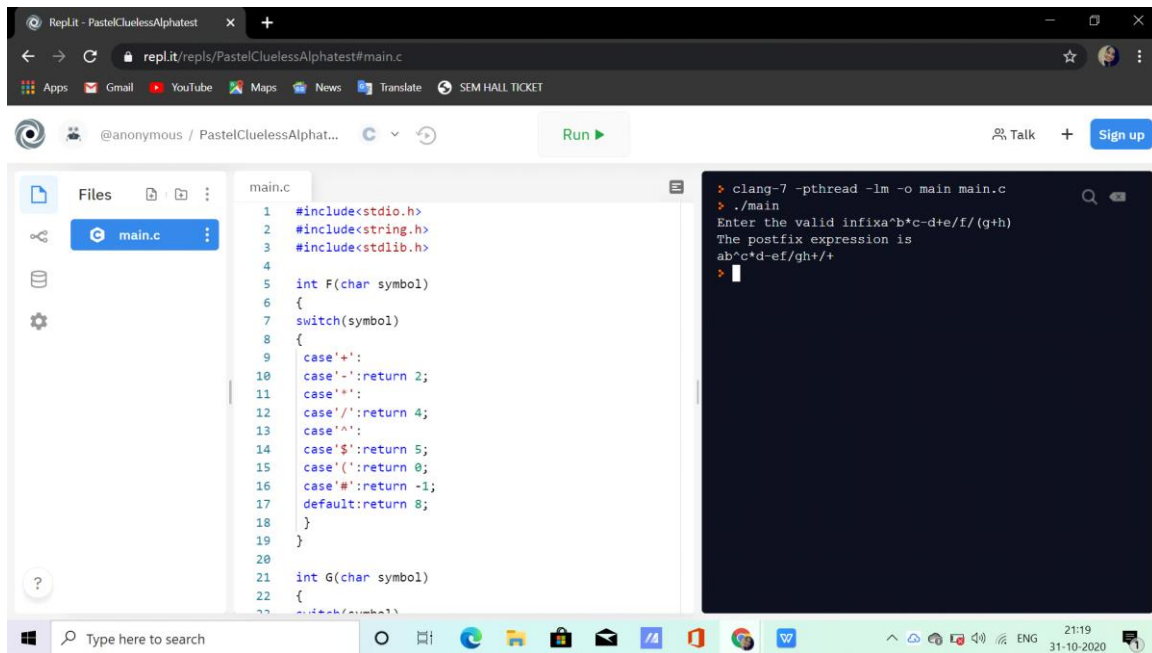
```
infix_postfix(infix , postfix );
```

```
printf("The postfix expression is \n");
```

```
printf("%s\n",postfix);
```

```
}
```

SCREENSHOT OF PROGRAM AND OUTPUT:-



Replit - PastelCluelessAlphatest

replit/repls/PastelCluelessAlphatest#main.c

@anonymous / PastelCluelessAlphat...

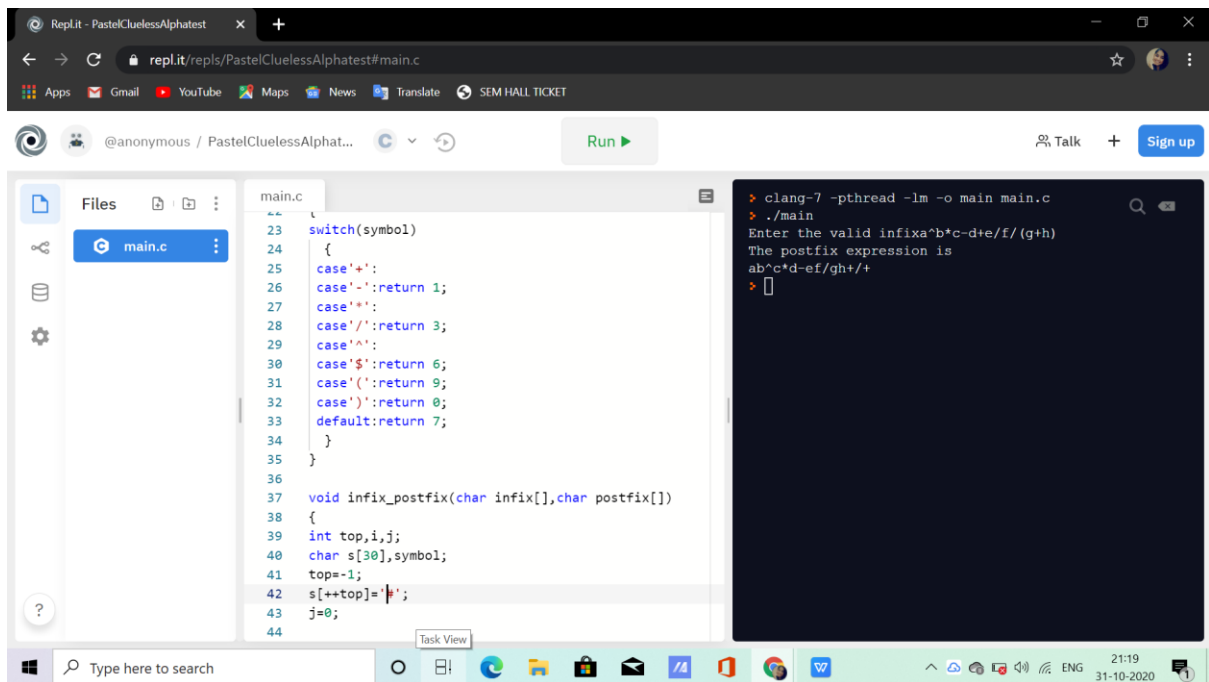
Run

Files

main.c

```
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)
```

```
> clang-7 -pthread -lm -o main main.c
> ./main
Enter the valid infixa^b*c-d+e/f/(g+h)
The postfix expression is
ab^c*d-ef/gh+/+
```



Replit - PastelCluelessAlphatest

replit/repls/PastelCluelessAlphatest#main.c

@anonymous / PastelCluelessAlphat...

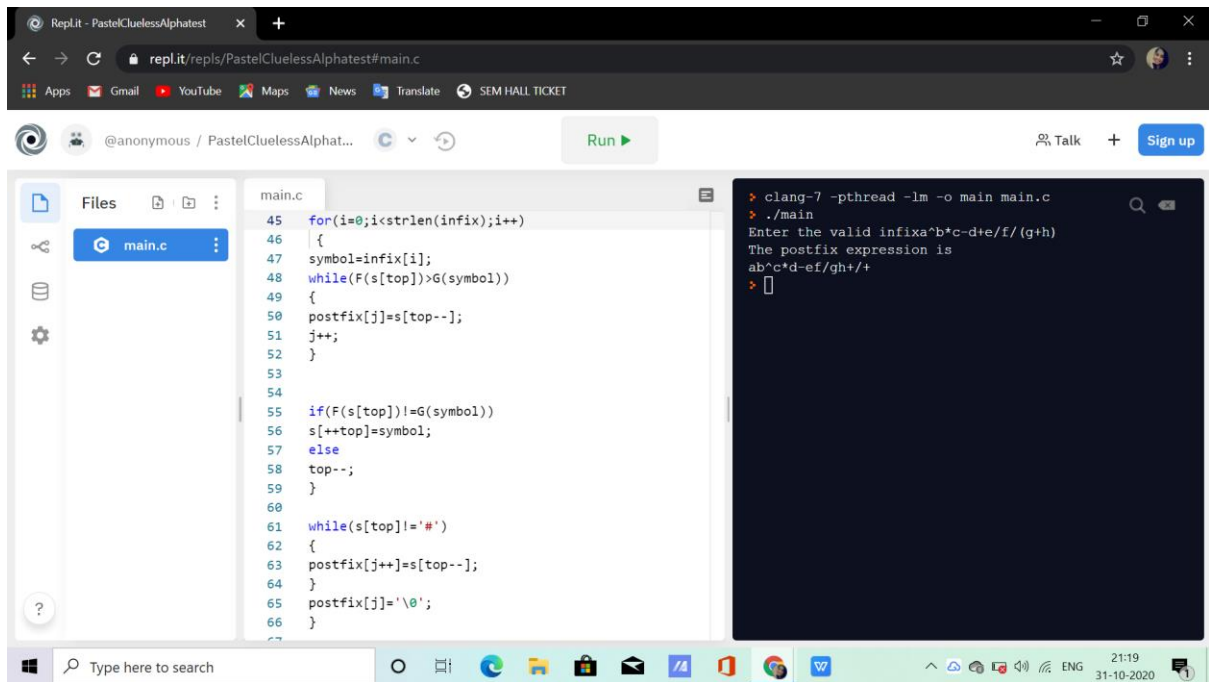
Run

Files

main.c

```
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[])
38 {
39     int top,i,j;
40     char s[30],symbol;
41     top=-1;
42     s[++top]=' ';
43     j=0;
44 }
```

```
> clang-7 -pthread -lm -o main main.c
> ./main
Enter the valid infixa^b*c-d+e/f/(g+h)
The postfix expression is
ab^c*d-ef/gh+/+
```



Replit - PastelCluelessAlphatest

replit/repls/PastelCluelessAlphatest#main.c

@anonymous / PastelCluelessAlphat...

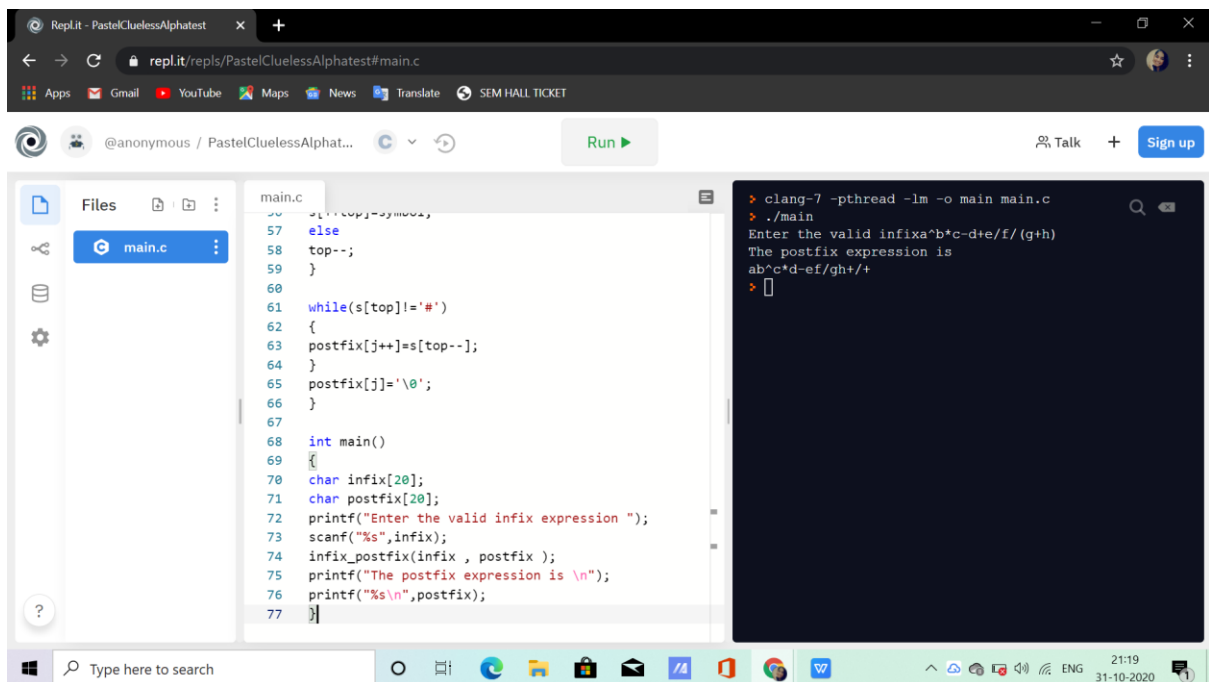
Run

Files

main.c

```
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 }
```

```
> clang-7 -pthread -lm -o main main.c
> ./main
Enter the valid infix a*b*c-d+e/f/(g+h)
The postfix expression is
ab^c*d-ef/gh+/+
```



Replit - PastelCluelessAlphatest

replit/repls/PastelCluelessAlphatest#main.c

@anonymous / PastelCluelessAlphat...

Run

Files

main.c

```
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 int main()
69 {
70     char infix[20];
71     char postfix[20];
72     printf("Enter the valid infix expression ");
73     scanf("%s",infix);
74     infix_postfix(infix , postfix );
75     printf("The postfix expression is \n");
76     printf("%s\n",postfix);
77 }
```

```
> clang-7 -pthread -lm -o main main.c
> ./main
Enter the valid infix a*b*c-d+e/f/(g+h)
The postfix expression is
ab^c*d-ef/gh+/+
```

WRITTEN:-

Qn:- WAP to convert a given valid parenthesized infix arithmetic expression to postfix expression. The expression consists of single character operands and the binary operators + (plus), - (minus), * (multiply) and / (divide).

- ① $(A + (B - C) * D)$
- ② $A + B$
- ③ $A \wedge b * c - d + e / f / (g + h)$
- ④ $x \wedge y \wedge z - m + n + p / q$
- ⑤ $a \wedge b * c - d + e / f / (g + h)$

Program:-

```
#include <stdio.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 (i = 0; i < strlen(infix); i++)
    {
        symbol = infix[i];
        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';  
}  
  
void main()  
{  
    char infix[20];  
    char postfix[20];  
    printf("Enter the valid infix expression");  
    scanf("%s", infix);  
    infix_postfix(infix, postfix);  
    printf("The postfix expression is \\n");  
    printf("%s \\n", postfix);  
}
```