*WAP to convert a given valid parenthesized infix arithmetic expression to postfix expression. The expression consists of single character operands & the binary operations +, -, * and /.

```
#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);
}
```

main.c

```c
#include<stdio.h>
#include<stdlib.h>
#include<string.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;
}
}
```

main.c

```
30  void infix_postfix(char infix[],char postfix[]){
31  int top,j,i;
32  char s[30];
33  char symbol;
34  top=-1;
35  s[++top]='#';
36  j=0;
37  for(i=0;i<strlen(infix);i++){
38  symbol=infix[i];
39
40  while(F(s[top])>G(symbol)){
41  postfix[j]=s[top--];
42  j++;
43  }
44  if(F(s[top])!=G(symbol)){
45  s[++top]=symbol;
46  }
47  else
48  top--;
49  }
50  while(s[top]!='#')
51  {
52  postfix[j++]=s[top--];
53  }
54  postfix[j]='\0';
55  }
56  void main()
57  {
58   char infix[20],postfix[20];
```

main.c

```
58      char infix[20],postfix[20];
59      int c1=0,c2=0;
60    printf("Enter the valid infix expression:\n");
61    scanf("%s",infix);
62    for(int k=0;k<strlen(infix);k++)
63    {
64        if(infix[k]=='(')
65        c1++;
66        else if(infix[k]==')')
67        c2++;
68        else
69        continue;
70    }
71    if(c1!=c2)
72    {
73        printf("Invalid infix expression!");
74        exit(0);
75    }
76    infix_postfix(infix,postfix);
77    printf("The postfix expression is:\n");
78    printf("%s\n",postfix);
79    }
```