

The screenshot shows a code editor interface with a dark theme. The menu bar includes 'File', 'View', 'Go', 'Run', 'Terminal', and 'Help'. The search bar at the top right contains the text '1BF24CS271'. The code editor displays a file named 'infix-to-postfix.c' with two tabs open. The left tab is labeled 'infix.c' and has a small number '2' next to it. The right tab is labeled 'infix-to-postfix.c' and has a small number '1' next to it. The code itself is as follows:

```
... C infix-to-postfix.c 2 X
C infix-to-postfix.c > main()
1 #include<stdio.h>
2 #include<ctype.h>
3 #include<string.h>
4 # define MAX 100
5 char stack[MAX];
6 int top=-1;
7 void push(char c){
8     if(top==MAX -1){
9         printf("Stack Overflow\n");
10        return ;
11    }
12    stack[++top]=c;
13 }
14 char pop(){
15     if(top== -1){
16         printf("stack Underflow\n");
17         return -1;
18     }
19     return stack[top--];
20 }
21 char peek(){
22     if(top== -1) return -1;
23     return stack[top];
24 }
25 int precedence(char op){
26     switch (op){
27         case '+':
28         case '-':
29             return 1;
30         case '*':
31         case '/':
32             return 2;
33         case '^':return 3;
34         case ')':
35             return 0;
36     }
37 }
38 return -1;
39 }
40 int associativity(char op){
41     if(op=='^')
42         return 1;
43     return 0;
44 }
45 void infixToPostfix(char infix[],char postfix[]){
46     int i,k=0;
47     char c;
48     for (i=0;infix[i]!='\0';i++){
49         if(isalnum(infix[i])){ // If operand
50             postfix[k]=infix[i];
51             k++;
52         }
53         else{ // If operator
54             if(precedence(c)<precedence(infix[i])){ // If current operator has higher precedence
55                 push(infix[i]);
56             }
57             else{ // If current operator has equal or lower precedence
58                 while(stack[0]>precedence(c)) // Pop operators from stack until current operator has higher precedence
59                     postfix[k]=pop();
60                 if(stack[0]==precedence(c)) // If current operator has equal precedence, pop it
61                     pop();
62                 push(infix[i]); // Push current operator onto stack
63             }
64         }
65     }
66     while(stack[0]) // Pop remaining operators from stack
67         postfix[k]=pop();
68     postfix[k]='\0';
69 }
```

```
PS C:\Users\student\Desktop\1BF24CS271\output> cd 'c:\Users\student\Desktop\1BF24CS271\output'
PS C:\Users\student\Desktop\1BF24CS271\output> & .\infix-to-postfix.exe
● enter a valid parantesized infix expression(A+(B*C-(D/E^F)*G)*H)
Postfix expr:ABC*DEF/G*-H*
● PS C:\Users\student\Desktop\1BF24CS271\output> cd 'c:\Users\student\Desktop\1BF24CS271\output'
● PS C:\Users\student\Desktop\1BF24CS271\output> & .\infix-to-postfix.exe
enter a valid parantesized infix expression(A+B)*C
Postfix expr:AB+C*
○ PS C:\Users\student\Desktop\1BF24CS271\output>
```

```
new Go Run Terminal Help
```

```
C infix-to-postfix 2 x
C infix-to-postfix.c > main()
40 int associativity(char op){
41     /* + - */
42     | return 1;
43     | return 0;
44 }
45 void infixToPostfix(char infix[],char postfix[]){
46     int i,k=0;
47     char c;
48     for (i=0;infix[i]!='\0';i++){
49         c=infix[i];
50         if (isalnum(c)){
51             | postfix[k++]=c;
52         }
53         else if(c=='('){
54             push(c);
55         }
56         else if(c==')'){
57             while (peek()!= '('){
58                 | postfix[k++]=pop();
59             }
60             pop();
61         }
62         else{
63             while ((top != -1&&(precedence(peek()) > precedence(c))||(precedence(peek()) == precedence(c)&& associativity(c)==0))){
64                 postfix[k++]=pop();
65             }
66             push(c);
67         }
68     }
69     while (top!=-1)
70     {postfix[k++]=pop();
71      /* code */
72     }
73     postfix[k]='\0';
74 }
75
76 int main() {
77     char infix[MAX],postfix[MAX];
78     printf("enter a valid parantesized infix expression");
79     scanf("%s",infix);
80     infixToPostfix(infix,postfix);
81     printf("Postfix expr:%s\n",postfix);
82     return 0;
83 }
```