



main.c

```
1 #include <stdio.h>
2
3 # define MAX 100
4 char stack[MAX];
5 int top=-1;
6
7 void push(char ch)
8 {
9     if (top==MAX-1)
10        printf("Stack is full\n");
11    else
12    {
13        top++;
14        stack[top]=ch;
15    }
16 }
17 char pop()
18 {
19     char item;
20     if (top==-1)
21         printf("\n stack is empty !");
22     else
23     {
24         item=stack[top];
25         top--;
26         return item;
27     }
28 }
```



main.c

```
24     item=stack[top];
25     top--;
26     return item;
27 }
28
29 }
30
31 int stackempty()
32 {
33     if(top== -1)
34         return 1;
35     else
36         return 0;
37 }
38
39 char stacktop()
40 {
41     if( top== -1)
42         printf("\n stack is empty!");
43     else
44         return stack[top];
45 }
46 int priority(char ch)
47 {
48     switch(ch)
49     {
50     case '+':
51     case '-':return (1);
```



```
1      case '+':  
2      case '-':return (1);  
3      case '*':  
4      case '/':return (2);  
5      case '^': return (3);  
6      default : return (0);  
7  }  
8  
9  
10  
11 int main(int argc, char **argv)  
12 {  
13     char infix[100];  
14     int i, item;  
15     printf("Enter the infix expression :");  
16     scanf("%s",infix);  
17     printf("Expression : %s",infix);  
18     printf("\n Postfix: ");  
19     i=0;  
20     while (infix[i]!='\0')  
21     {  
22         switch (infix[i])  
23         {  
24             case ' ': push(infix[i]);  
25                 break;  
26             case ')': pop();  
27             case ',': break;  
28             case '+': push(infix[i]);  
29                 break;  
30             case '-': push(infix[i]);  
31                 break;  
32             case '*': push(infix[i]);  
33                 break;  
34             case '/': push(infix[i]);  
35                 break;  
36             case '^': push(infix[i]);  
37                 break;  
38             case '.': push(infix[i]);  
39                 break;  
40             case '#': pop();  
41                 break;  
42             case '=': pop();  
43                 break;  
44             case '<': push(infix[i]);  
45                 break;  
46             case '>': push(infix[i]);  
47                 break;  
48             case '&': push(infix[i]);  
49                 break;  
50             case '|': push(infix[i]);  
51                 break;  
52             case '&&': push(infix[i]);  
53                 break;  
54             case '!': push(infix[i]);  
55                 break;  
56             case '<=': push(infix[i]);  
57                 break;  
58             case '>=': push(infix[i]);  
59                 break;  
60             case '<>': push(infix[i]);  
61                 break;  
62             case '<<': push(infix[i]);  
63                 break;  
64             case '>>': push(infix[i]);  
65                 break;  
66             case '<<=': push(infix[i]);  
67                 break;  
68             case '>>=': push(infix[i]);  
69                 break;  
70             case '<<>>': push(infix[i]);  
71                 break;  
72             case '<<>>=': push(infix[i]);  
73                 break;  
74             case '<<>>=': push(infix[i]);  
75                 break;  
76             case '<<>>=': push(infix[i]);  
77                 break;
```

```
switch (infix[i])
{
    case '(': push(infix[i]);
                break;
    case ')': while(( item=pop())!= '(')
                printf("%c",item);
                break;
    case '+':
    case '-':
    case '*':
    case '/':
    case '^':
                while(!stackempty() && priority(infix
[ini])<=priority(stacktop())))
                {
                    item=pop();
                    printf("%c", item);
                }
                push(infix[i]);
                break;
    default : printf("%c", infix[i]);
                break;
}
```



main.c

```
94         | | | | | break;
95         | | | | default : printf("%c", infix[i]);
96         | | | | break;
97
98     }
99     i++;
100 }
101
102
103 while(!stackempty())
104 {
105     char item;
106     item=pop();
107     printf("%c", item);
108
109 }
110 printf("\n");
111 return 0;
112 }
113 }
```

