

Infix to Postfix

- Q. Write a program to convert a given valid parenthesized infix expression to postfix expression.

```
#include <stdio.h>
```

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

```
#include <conio.h>
```

```
int i=0, pos=0, top=-1, length;  
char symbol, temp, infix[20], postfix[20], stack[20];
```

```
void infixtopostfix();
```

```
void push(char symbol);
```

```
char pop();
```

```
int precedence(char symbol);
```

```
int main()
```

```
{
```

```
printf("Enter infix expression: \n");
```

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

```
infixtopostfix();
```

```
printf("\n Infix Expression: \n %s", infix);
```

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

```
return 0;
```

```
}
```

```
void infixtopostfix() {
```

```
length = strlen(infix);
```

```
push('#');
```



```

while (i < length) {
    symbol = infix[i];
    switch (symbol) {
        case '(':
            push(symbol);
            break;

        case ')':
            temp = pop();
            while (temp != '(') {
                postfix[pos++] = temp;
                temp = pop();
            }
            break;

        case '+':
        case '-':
        case '*':
        case '/':
        case '^':
            while (preced(stack[top]) >= preced(symbol)) {
                temp = pop();
                postfix[pos++] = temp;
            }
            push(symbol);
            break;

        default:
            postfix[pos++] = symbol;
    }
    i++;
}

```



```
while (top > 0) {  
    temp = pop();  
    postfix[pos++] = temp;
```

```
}
```

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

```
}
```

```
void push(char symbol) {  
    top = top + 1;  
    stack[top] = symbol;
```

```
}
```

```
char pop() {  
    return stack[top--];
```

```
}
```

```
int precedence(char symbol) {
```

```
    int p;
```

```
    switch (symbol) {
```

```
        case '^': p = 3;
```

```
        break;
```

```
        case '*':
```

```
        case '/': p = 2;
```

```
        break;
```

```
        case '+':
```

```
        case '-': p = 1;
```

```
        break;
```

```
        case '(': p = 0;
```

```
        break;
```

```
        default: p = -1;
```

```
        break;
```

```
}
```


return p;

}

Output

Enter Infix Expression:

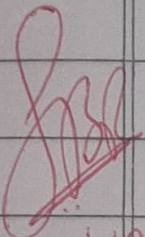
$$A^B * C - D + E / F / (G + H)$$

~~Infix Expression:~~

~~$$A^B * C - D + E / F / (G + H)$$~~

~~Postfix Expression:~~

~~$$AB^C * D - EF / GH + I +$$~~


8/10/24