(Prefix Infix Postfix Conversion)

1) Infix to Postfix :-

a + b * (c ^ d - e)

i=0	ut	ans
a		a
+	+	a
b	+	ab
*	+*	ab
(	+*(	ab
c	+*(	abc
^	+*(^	abc
d	+*(^	abcd
-	+*(^-	abcd^
e	+*(-	abcd^e
)	+*	abcd^e-
		abcd^e-*+

Pseudo code –

⇒

```
infixtopostfix (s)
{
    i=0, st, ans = " "
    While (i<n)
    {
        if (s[i] >= 'A' && s[i] <= 'Z') or
           (s[i] >= 'a' && s[i] <= 'z') or
           (s[i] >= '0' && s[i] <= '9')

            ans = ans + s[i];      —— O(1)
```

```
else if (s[i] == '(')
        st.push(s[i]);          ——— O(1)
else if (s[i] == ')')
{
    while (st.empty() && st.top() != '(')
    {
        ans += st.top()
        st.pop();               ——— O(N)
    }
    st.pop();
}
else {
    While(st.empty() && priority(s[i] <= priority(st
    {
        ans += st.top();
        st.pop();
    }                           ——— (N)
    st.push(s[i]);
}
i++;
}
```

TC :- O(N) + O(N)

SC :- O(N) + O(N)

```
While (! st.empty()){
        ans += st.top();
        st.pop();
}
return ans;
}
```

y) Infix to Prefix :-

- Revese the infix
- Infix 2 postfix
- Revese that answer

x ~~(F + D - C * (B~~

⇒ (A+B) * C - D + F

- Revese : F + D - C * (B + A)

symbol	stack	expression
F		F
+	+	F
D	+	FD
-	+ -	FD
C	+ -	FDC
*	+ - *	FDC
(	+ - * (	FDBC
+	+ - * (	FDBC
A	+ - * (+	FDBCA
)	+ - *	FDBCA+

FDBCA~~+~~* - +

• code :-

```
fn(A) {
    reverse(s) ——→ (swap ')' & '(' as well)
    i=0 , ans="" , st;

    while(i<N) {
        if (operand) ans+=s[i];
        else if (opening) st.push(s[i]);
        else if (closing) {
            while (!st.empty() && st.top()!='(')
            {
                ans+=st.top();
                st.pop();
            }
            st.pop();
        }
        else {
            if (s[i]=='^') {
                while (!st.empty() && priority(s[i])<
                                      priority(st.top()
                {
                    ans+=st.top();
                    st.pop();
                }
            }
            else {
                while (!st.empty() && priority(s[i]<pr
                                      (st.top())
                {
                    ans+=st.top();
                    st.pop();
                }
            }
        }
```

```
        st.push(s[i]);
    }
    i++;
}
while(!st.empty()){
    ans += st.top();
    st.pop();
}
ans = rev(ans);
return ans;
}
```

TC :- $O(N/2) + O(N/2) + O(2N) + O(3N)$

SC :- $O(N)$

) Postfix to Infix :-

$AB-DE+F*/$

1. start from left
2. push in order t2 & t1

i	st
A	A
B	AB
-	(A-B),
D	(A-B), D
E	(A-B), DE
+	(A-B), (D+E)
F	(A-B), (D+E), F
*	(A-B), ((D+E)*F)
/	(A-B)/((D+E)*F)

→ code :-

```
fcn(s) {
    i=0, st;
    while (i<N) {  ⟶ O(N)
        if(operand)  st.push(s[i]);
        else {
            t1 = st.top(); st.pop();
            t2 = st.top(); st.pop();  ⟶ O(N1+N2)
            s = "" + '(' + t2 + s[i] + t1 + ')';
            st.push(s);
        }
    i++;
}
return st.top();
}
```

TC :- O(N) + O(N)

SC :- O(N)

4) PrefixToInfix :-

* + P Q − M N

i
N
M
−
Q
P
+
*

st
N
N, M
(M−N)
(M−N), Q
(M−N), (Q P
(M−N), (P + Q)
(P+Q) * (N−N)

1. Start from r
2. Push in ord
   t1 & t2

Code :-

```
fc^n(s) {
    i = N-1
    while (i >= 0) { ⟶ O(N)

        if (operand)  st.push(s[i]);
        else {
            t1 = st.top()  st.pop()
            t2 = st.top()  st.pop()        ⟶ O(N1+N2)
            ts = '(' + t1 + s[i] + t2 + ')';
            st.push(ts)
        }
        i--;
    }
    return st.top()
}
```

TC :- O(N) + O(N)

SC :- O(N)

5] Postfix to prefix :-

AB - DE + F * /

i	st
A	A
B	A, B
-	-AB
D	-AB, D
E	-AB, D, E
+	-AB + DE
F	-AB, + DE, F
*	-AB, * + DEF
/	/-AB * + DEF

1. start from left
2. Insert operator at start

⇒ code :-

```
fcn(s) {
    i=0, st
    while (i < N) {          O(N)
        if (operand) st.push(s[i])
        else {
            t1 = st.top()  st.pop()
            t2 = st.top()  st.pop()
            st.push(s[i] + t2 + t1)      O(N)
        }
    i++;
}
return st.top()
}
```

TC :- O(N)
SC :- O(N)

6) Prefix to Postfix :-

/ - AB * + DE F

1. start from right
2. push operand at right

i	st
F	F
E	F, E
D	F, E, D
+	F, DE+
*	DE+F*
B	DE+F*, B
A	DE+F*, B, A
-	DE+F*, AB-
/	AB-DE+F*/

code :-

```
fcn(s) {
    i = N-1;
    while (i>=0) {
        if (operand)  st.push(s[i]);
        else {
            t1 = st.top()  st.pop()
            t2 = st.top()  st.pop()
            st.push(t1+t2+s[i])
        }
    i++
    }
return st.top()
}
```

TC :- O(2N)

SC :- O(N)