Stack

Algorithm for Prefix to Infix:

- Read the Prefix expression in reverse order (from right to left)
- If the symbol is an operand, then push it onto the Stack
- If the symbol is an operator, then pop two operands from the Stack

  Create a string by concatenating the two operands and the operator between them.

  string = (operand1 + operator + operand2)

  And push the resultant string back to Stack
- Repeat the above steps until the end of Prefix expression.
- At the end stack will have only 1 string i.e resultant string

Expression: *+AB-CD
DC-BA+*

Expression	Stack	Result
D	D	
C	D,C	
-	Z1	Z1=C-D
B	Z1,B	
A	Z1,B,A	
+	Z1,Z2	Z2=A+B
*	Z	Z=Z2*Z1

So, the expression now becomes: Z= (A+B)*(C-D). Since Z1 and Z2 are seperate strings, we can enclose them in brackets

Algorithm for postfix to infix conversion

1. Read the symbol from the input .based on the input symbol go to step 2 or 3.
2. If symbol is operand then push it into stack.
3. If symbol is operator then pop top 2 values from the stack.
4. this 2 popped value is our operand .
5. create a new string and put the operator between this operand in string.
6. push this string into stack.
7. At the end only one value remain in stack which is our infix expression.

Expression: `AB*C+`

Expression	Stack	Result
A	A	
B	A,B	
*	Z1	Z1=A*B
C	Z1,C	
+	Z	Z=Z1+C

So, the expression now becomes: Z= (A*B)+C. Since Z1 is a seperate strings, we can enclose them in brackets