Postfix to Infix Conversion Algorithm

1. Initialize an empty stack to store operands (sub-expressions).
2. Scan the postfix expression from left to right.
3. For each character in the postfix expression:
    - **IF** the character is an operand (digit, variable, etc.), push it onto the stack.
    - **IF** the character is an operator:
        - a. Pop the top two operands from the stack.
        - b. Form the infix expression `(operand2 operator operand1)`.
        - c. Push the resulting infix sub-expression back onto the stack.
4. Repeat this process until the end of the postfix expression.
5. At the end, the stack will contain the final infix expression.

Prefix to Infix Conversion Algorithm

1. Initialize an empty stack to store operands (sub-expressions).
2. Scan the prefix expression from right to left.
3. For each character in the prefix expression:
    - **IF** the character is an operand (digit, variable, etc.), push it onto the stack.
    - **IF** the character is an operator:
        - a. Pop the top two operands from the stack.
        - b. Form the infix expression `(operand1 operator operand2)`.
        - c. Push the resulting infix sub-expression back onto the stack.
4. Repeat this process until the end of the prefix expression.
5. At the end, the stack will contain the final infix expression.