Part 4
Problem: **Infix to Postfix Converter**

Use stack to handle the operator precedence

1. Get input: Infix expression.
2. **Convert Infix → Postfix**.
3. Calculate result: Postfix expression
4. Output result.

Example:

Input infix exp: 4 + 2 * 3

Convert infix-postfix: 4 + 2 * 3 $\rightarrow$ 4 2 3 * +

Calculate postfix exp: = 4 2 3 * +
= 4 6 +
= 10

Output: 10

Example

- 4 + 2 * 3 → 4 2 3 * +
- 4 * 2 + 3 → 4 2 * 3 +
- 2 + 3 – 45 * 1 / 3 → 2 3 + 45 1 * 3 / –

Simple Calculator : **Infix to Postfix Converter**

- The algorithm:
  a) **Operands:**
     - Immediately output.

  b) **Close parenthesis:**
     - Pop stack symbols until an open parenthesis appears.

  c) **Operator:**
     - Pop all stack symbols until a symbol of lower precedence or a right-associative symbol of equal precedence appears.
     - Then push the operator.

  d) **End of input:**
     - Pop all remaining stack symbols.

Demonstration

Stack Applications

Reverse order
- string
- palindrome

Simple calculator
1. Infix to Postfix Converter
   - 4 + 2 * 3 → 4 2 3 * +
2. Postfix Machine
   - 4 2 3 * +
   - = 4 6 +
   - = 10

Parsing in a compiler
- Balanced symbol checker. e.g. ( ) { } [ ]
- HTML tags