Infix to Prefix Conversion using Two Stacks

----

1. Code likhne ki soch (Thought Process)

Ye code ek infix arithmetic expression ko prefix me convert karta hai do stacks ka use karke:

- Value Stack (val) -> Ye operands (numbers) ko prefix format me store karta hai.
- Operator Stack (op) -> Ye operators (+, -, *, /, (,)) ko store karta hai aur precedence maintain karta hai.

Code ka flow:

1. Expression ko left se right read karo.
2. Agar koi number mile to use val stack me push karo.
3. Agar koi operator mile:
   - '(' aaya to use op stack me push karo.
   - ')' aaya to brackets ke andar ka sab evaluate karo.
   - Agar precedence zyada hai to push karo.
   - Agar kam hai to jab tak precedence rule satisfy na ho tab tak pop aur evaluate karo.
4. Loop khatam hone ke baad jo bache operators hai unko evaluate karo.
5. Final result val.top() me milega jo prefix expression hoga.

----

2. Dry Run & Step-by-Step Example

Test Case 1:
Expression: "1+(2+6)*4/8-3"
Expected Output (Prefix): "-+1/*+26483"

Step-by-Step Execution:

1. '1' -> Number -> val stack me push -> val = ["1"]
2. '+' -> Operator -> op stack me push -> op = ['+']
3. '(' -> Push to op -> op = ['+', '(']
4. '2' -> Number -> val stack me push -> val = ["1", "2"]
5. '+' -> Operator -> Push to op -> op = ['+', '(', '+']
6. '6' -> Number -> val stack me push -> val = ["1", "2", "6"]
7. ')' -> Evaluate (2+6) -> Pop '+' aur combine "2" aur "6" -> "+26" -> val = ["1", "+26"]
8. '*' -> Push to op -> op = ['+', '*']
9. '4' -> Number -> val stack me push -> val = ["1", "+26", "4"]
10. '/' -> Push to op -> op = ['+', '*', '/']
11. '8' -> Number -> val stack me push -> val = ["1", "+26", "4", "8"]
12. '-' -> Pop aur evaluate '/' -> "*+264" -> val = ["1", "/*+264", "8"]
13. Pop aur evaluate '*' -> "/*+2648" -> val = ["1", "/*+2648"]
14. Pop aur evaluate '+' -> "+1/*+2648" -> val = ["+1/*+2648"]
15. '3' -> Number -> val stack me push -> val = ["+1/*+2648", "3"]
16. Evaluate '-' -> Final result: "-+1/*+26483"

Summary & Short Notes:

- Two Stacks ka use kiya gaya hai (val & op).
- Operator precedence correctly handle hota hai.
- Brackets ko sahi tarike se process kiya gaya hai.
- Final prefix expression val.top() me store hota hai.

Important Points:
- Val stack numbers ko prefix format me store karta hai.
- Op stack operators ko precedence k hisaab se manage karta hai.
- '(' push hota hai, ')' aane par evaluate hota hai.
- Last me jo bache operators hote hain unko evaluate karke final result milta hai.

Common Mistakes & Fixes:

- priority() function me default return 0 karna chahiye.
- Brackets ka handling sahi se hona chahiye.
- Ye code sirf single-digit numbers ko support karta hai (multi-digit ke liye modify kar sakte ho).
