Infix to Prefix Conversion using Two Stacks

---

1. Thought Process Behind the Code

This code converts an arithmetic expression written in infix notation into prefix notation using two stacks:

- Value Stack (val) -> Stores operands (numbers) in prefix format.
- Operator Stack (op) -> Stores operators (+, -, *, /, (,)) and ensures precedence is maintained.

Code Workflow:
1. Read the expression from left to right.
2. If a digit is found, push it into the val stack as a string.
3. If an operator is found:
   - If '(' is encountered, push it onto op stack.
   - If ')' is encountered, pop and evaluate everything inside the brackets.
   - If it has higher precedence than the top of op stack, push it.
   - If lower precedence, pop from op and evaluate until precedence rule is satisfied.
4. After loop ends, evaluate remaining operators.
5. The final result stored in val.top() contains the prefix expression.

---

2. Dry Run & Step-by-Step Test Case Explanations

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

Step-by-Step Execution:

1. Read '1' -> Operand -> Push to val stack -> val = ["1"]

2. Read '+' -> Operator -> Push to op stack -> op = ['+']

3. Read '(' -> Push to op stack -> op = ['+', '(']

4. Read '2' -> Operand -> Push to val -> val = ["1", "2"]

5. Read '+' -> Operator -> Push to op stack -> op = ['+', '(', '+']

6. Read '6' -> Operand -> Push to val -> val = ["1", "2", "6"]

7. Read ')' -> Evaluate (2+6) -> Pop '+' and combine "2" and "6" -> "+26" -> val = ["1", "+26"]

8. Read '*' -> Push to op -> op = ['+', '*']

9. Read '4' -> Operand -> Push to val -> val = ["1", "+26", "4"]

10. Read '/' -> Push to op -> op = ['+', '*', '/']

11. Read '8' -> Operand -> Push to val -> val = ["1", "+26", "4", "8"]

12. Read '-' -> Pop and evaluate '/' -> "*+264" -> val = ["1", "/*+264", "8"]

13. Pop and evaluate '*' -> "/*+2648" -> val = ["1", "/*+2648"]

14. Pop and evaluate '+' -> "+1/*+2648" -> val = ["+1/*+2648"]

15. Read '3' -> Operand -> Push to val -> val = ["+1/*+2648", "3"]

16. Evaluate '-' -> Final result: "-+1/*+26483"

Summary & Notes:

- Uses two stacks (val & op) for infix to prefix conversion.
- Handles operator precedence correctly.
- Brackets are processed in the right order.
- Final prefix expression stored in val.top().

Key Concepts:

- Two Stack Approach -> val (operands), op (operators).
- Operator Precedence -> * and / > + and -.
- Bracket Handling -> '(' push hota hai, ')' pe evaluation hoti hai.
- Final Evaluation -> Remaining operators are processed at the end.

Common Mistakes & Fixes:

- priority() should return 0 by default.
- Bracket handling must be implemented correctly.
- The code currently supports only single-digit numbers (multi-digit support can be added).
