DeepSeek - Into the Unknown

chat.deepseek.com/a/chat/s/f4835de2-d967-4222-b90c-9c6d52cf35dd

Given Infix Expression:

Q:A+(B∗C−(D/E↑F)∗G)∗H

Steps to Convert Infix to Postfix:

1. Initialize an empty stack and an empty output list.
2. Scan the expression Q from left to right.
3. For each token in Q:
    - If the token is an **operand** (e.g., A,B,C), add it to the output list.
    - If the token is an **opening parenthesis** `(`, push it onto the stack.
    - If the token is a **closing parenthesis** `)`, pop from the stack and add to the output list until an opening parenthesis `(` is encountered. Pop and discard the `(`.
    - If the token is an **operator**:
        - Pop from the stack and add to the output list any operators with higher precedence (or equal precedence for left-associative operators) than the current token.
        - Push the current token onto the stack.
4. After scanning the entire expression, pop any remaining operators from the stack and add them to the output list.
5. The output list is the postfix expression P.

Operator Precedence and Associativity:

- Exponentiation (`↑`): Highest precedence, right-associative.
- Multiplication (`*`) and Division (`/`): Medium precedence, left-associative.
- Addition (`+`) and Subtraction (`-`): Lowest precedence, left-associative.

Step-by-Step Conversion:

Input Expression:

Q:A+(B∗C−(D/E↑F)∗G)∗H

Tokens:

A,+,(,B,∗,C,−,(,D,/,E,↑,F,),∗,G,),∗,H

Step-by-Step Process:

Token	Action	Stack	Output List
A	Operand: Add to output.	`[]`	`[A]`
+	Operator: Push onto stack.	`[+]`	`[A]`
(	Opening parenthesis: Push onto stack.	`[+, (]`	`[A]`
B	Operand: Add to output.	`[+, (]`	`[A, B]`
∗	Operator: Push onto stack.	`[+, (, *]`	`[A, B]`
C	Operand: Add to output.	`[+, (, *]`	`[A, B, C]`
−	Operator: Pop `*` (higher precedence) and add to output. Push `-`.	`[+, (, -]`	`[A, B, C, *]`
(	Opening parenthesis: Push onto stack.	`[+, (, -, (]`	`[A, B, C, *]`
D	Operand: Add to output.	`[+, (, -, (]`	`[A, B, C, *, D]`
/	Operator: Push onto stack.	`[+, (, -, (, /]`	`[A, B, C, *, D]`
E	Operand: Add to output.	`[+, (, -, (, /]`	`[A, B, C, *, D, E]`
↑	Operator: Push onto stack (higher precedence).	`[+, (, -, (, /, ↑]`	`[A, B, C, *, D, E]`
F	Operand: Add to output.	`[+, (, -, (, /, ↑]`	`[A, B, C, *, D, E, F]`
)	Closing parenthesis: Pop and add to output until `(` is encountered.	`[+, (, -]`	`[A, B, C, *, D, E, F, ↑, /]`
∗	Operator: Pop `-` (equal precedence) and add to output. Push `*`.	`[+, (, *]`	`[A, B, C, *, D, E, F, ↑, /, -]`
G	Operand: Add to output.	`[+, (, *]`	`[A, B, C, *, D, E, F, ↑, /, -, G]`
)	Closing parenthesis: Pop and add to output until `(` is encountered.	`[+]`	`[A, B, C, *, D, E, F, ↑, /, -, G, *]`

Token	Action	Stack	Output List
*	Operator: Pop + (equal precedence) and add to output. Push *.	[*]	[A, B, C, *, D, E, F, ↑, /, -, G, *, +]
H	Operand: Add to output.	[*]	[A, B, C, *, D, E, F, ↑, /, -, G, *, +, H]

Final Stack:

Pop the remaining * and add to output.

Output: [A, B, C, *, D, E, F, ↑, /, -, G, *, +, H, *]

Postfix Expression P:

P:ABC∗DEF↑/−G∗+H∗

Summary:

The infix expression Q:A+(B∗C−(D/E↑F)∗G)∗H is transformed into the postfix expression: P:ABC∗DEF↑/−G∗+H∗

This postfix expression can now be evaluated using a stack-based algorithm.