I) A + ((B – C * D) / E) + F – G / H

1- Infix to Postfix

1- Convert

Token	Operation	Stack	Postfix
A	Pass operand to out		A
+	Pass operator to stack if stack is empty	+	A
(	Pass left parenthesis into stack	+, (	A
(	Pass left parenthesis into stack	+, (, (	A
B	Pass operand to out	+, (, (	A B
-	Pass all operators down to first left parenthesis that have higher or equal precedence into stack and push token into stack	+, (, (, -	A B
C	Pass operand to out	+, (, (, -	A B C
*	Pass all operators down to first left parenthesis that have higher or equal precedence into stack and push token into stack	+, (, (, -, *	A B C
D	Pass operand to out	+, (, (, -, *	A B C D
)	Pass all operators down to first left parenthesis to out	+, (	A B C D * -
/	Pass all operators down to first left parenthesis that have higher or equal precedence into stack and push token into stack	+, (, /	A B C D * -
E	Pass operand to out	+, (, /	A B C D * - E
)	Pass all operators down to first left parenthesis to out	+	A B C D * - E /
+	Pass all operators down to first left parenthesis that have higher or equal precedence into stack and push token into stack	+	A B C D * - E / +
F	Pass operand to out	+	A B C D * - E / + F
-	Pass all operators down to first left parenthesis that have higher or equal precedence into stack and push token into stack	-	A B C D * - E / + F +
G	Pass operand to out	-	A B C D * - E / + F + G
/	Pass all operators down to first left parenthesis that have higher or equal precedence into stack and push token into stack	- /	A B C D * - E / + F + G
H	Pass operand to out	- /	A B C D * - E / + F + G H
			A B C D * - E / + F + G H / -

2- Evaluate

Infix notation: A + ((B – C * D) / E) + F – G / H

Postfix notation: A B C D * - E / + F + G H / -

Let A = 6, B = 12, C = 2, D = 1, E = 5, F = 8, G = 9, H = 3

Infix notation: 6 + ((12 – 2 * 1) / 5) + 8 – 9 / 3

Postfix notation: 6 12 2 1 * - 5 / + 8 + 9 3 / -

6 12 2 1 * - 5 / + 8 + 9 3 / -	2 1 *	6 12 2 - 5 / + 8 + 9 3 / -
6 12 2 - 5 / + 8 + 9 3 / -	12 2 -	6 10 5 / + 8 + 9 3 / -
6 10 5 / + 8 + 9 3 / -	10 5 /	6 2 + 8 + 9 3 / -
6 2 + 8 + 9 3 / -	6 2 +	8 8 + 9 3 / -
8 8 + 9 3 / -	8 8 +	16 9 3 / -
16 9 3 / -	9 3 /	16 3 -
16 3 -	16 3 -	13

2- Infix to Prefix

1- Convert

Reverse infix: H / G – F + (E / (D * C – B)) + A

Token	Operation	Stack	Postfix
H	Pass operand to out		H
/	Pass operator to stack if stack is empty	/	H
G	Pass operand to out	/	H G
-	Pass all operators down to first left parenthesis that have higher or equal precedence into stack and push token into stack	-	H G /
F	Pass operand to out	-	H G / F
+	Pass all operators down to first left parenthesis that have higher or equal precedence into stack and push token into stack	+	H G / F -
(	Pass left parenthesis into stack	+, (	H G / F -
E	Pass operand to out	+, (	H G / F - E
/	Pass all operators down to first left parenthesis that have higher or equal precedence into stack and push token into stack	+, (/	H G / F - E
(	Pass left parenthesis into stack	+, (/ (	H G / F - E
D	Pass operand to out	+, (/ (	H G / F – E D
*	Pass all operators down to first left parenthesis that have higher or equal precedence into stack and push token into stack	+, (/ (*	H G / F – E D
C	Pass operand to out	+, (/ (*	H G / F – E D C
-	Pass all operators down to first left parenthesis that have higher or equal precedence into stack and push token into stack	+, (/ (-	H G / F – E D C *
B	Pass operand to out	+, (/ (-	H G / F – E D C * B
)	Pass all operators down to first left parenthesis to out	+, (/	H G / F – E D C * B -
)	Pass all operators down to first left parenthesis to out	+	H G / F – E D C * B - /
+	Pass all operators down to first left parenthesis that have higher or equal precedence into stack and push token into stack	+	H G / F – E D C * B - / +
A	Pass operand to out	+	H G / F – E D C * B - / + A
			H G / F – E D C * B - / + A +

Reverse expression: + A + / - B * C D E – F / G H

2- Evaluate

Infix notation: A + ((B – C * D) / E) + F – G / H

Prefix notation: + A + / - B * C D E – F / G H

Let A = 6, B = 12, C = 2, D = 1, E = 5, F = 8, G = 9, H = 3

Infix notation: 6 + ((12 – 2 * 1) / 5) + 8 – 9 / 3

Prefix notation: + 6 + / - 12 * 2 1 5 – 8 / 9 3

+ 6 + / - 12 * 2 1 5 – 8 / 9 3	/ 9 3	+ 6 + / - 12 * 2 1 5 – 8 3
+ 6 + / - 12 * 2 1 5 – 8 3	- 8 3	+ 6 + / - 12 * 2 1 5
+ 6 + / - 12 * 2 1 5 5	* 2 1	+ 6 + / - 12 2 5 5
+ 6 + / - 12 2 5 5	- 12 2	+ 6 + / 10 5 5
+ 6 + / 10 5 5	/ 10 5	+ 6 + 2 5
+ 6 + 2 5	+ 2 5	+ 6 7
+ 6 7	+ 6 7	13

II) ! (A && ! ((B < C) | | (C > D))) | | (C < E)

1- Infix to Postfix

1- Convert

Token	Operation	Stack	Postfix
!	Pass operator to stack	!	
(	Pass left parenthesis into stack	!, (	
A	Pass operand to out	!, (	A
&&	Pass operator to stack	!, (, &&	A
!	Pass operator to stack	!, (, &&, !	A
(	Pass left parenthesis into stack	!, (, &&, !, (	A
(	Pass left parenthesis into stack	!, (, &&, !, (, (	A
B	Pass operand to out	!, (, &&, !, (, (	A B
<	Pass operator to stack	!, (, &&, !, (, (, <	A B
C	Pass operand to out	!, (, &&, !, (, (, <	A B C
)	Pass all operators down to first left parenthesis to out	!, (, &&, !, (	A B C <
| |	Pass operator to stack	!, (, &&, !, (, | |	A B C <
(	Pass left parenthesis into stack	!, (, &&, !, (, | |, (	A B C <
C	Pass operand to out	!, (, &&, !, (, | |, (	A B C < C
>	Pass operator to stack	!, (, &&, !, (, | |, (, >	A B C < C
D	Pass operand to out	!, (, &&, !, (, | |, (, >	A B C < C D
)	Pass all operators down to first left parenthesis to out	!, (, &&, !, (, | |	A B C < C D >
)	Pass all operators down to first left parenthesis to out	!, (, &&	A B C < C D > | | !
)	Pass all operators down to first left parenthesis to out		A B C < C D > | | ! && !
| |	Pass operator to stack	| |	A B C < C D > | | ! && !
(	Pass left parenthesis into stack	| |, (	A B C < C D > | | ! && !
C	Pass operand to out	| |, (	A B C < C D > | | ! && ! C
<	Pass operator to stack	| |, (, <	A B C < C D > | | ! && ! C
E	Pass operand to out	| |, (, <	A B C < C D > | | ! && ! C E
)	Pass all operators down to first left parenthesis to out	| |	A B C < C D > | | ! && ! C E <
			A B C < C D > | | ! && ! C E < | |

2- Evaluate

Infix notation: ! (A && ! ((B < C) | | (C > D))) | | (C < E)

Postfix notation: A B C < C D > | | ! && ! C E < | |

Let A = 1, B = 2, C = 3, D = 5, E = 3

Infix notation: ! (1 && ! ((2 < 3) | | (3 > 5))) | | (3 < 3)

Postfix notation: 1 2 3 < 3 5 > | | ! && ! 3 3 < | |

1 2 3 < 3 5 > | | ! && ! 3 3 < | |	2 3 <	1 1 3 5 > | | ! && ! 3 3 < | |
1 1 3 5 > | | ! && ! 3 3 < | |	3 5 >	1 1 0 | | ! && ! 3 3 < | |
1 1 0 | | ! && ! 3 3 < | |	1 0 | |	1 1 ! && ! 3 3 < | |
1 1 ! && ! 3 3 < | |	1 !	1 0 && ! 3 3 < | |
1 0 && ! 3 3 < | |	1 0 &&	0 ! 3 3 < | |
0 ! 3 3 < | |	0 !	1 3 3 < | |
1 3 3 < | |	3 3 <	1 0 | |
1 0 | |	1 0 | |	1

2- Infix to Prefix

1- Convert

Reverse infix: (E > C) || !(!((D < C) || (C > B)) && A)

Token	Operation	Stack	Postfix
(	Pass left parenthesis into stack	(	
E	Pass operand to out	(	E
>	Pass operator to stack	(, >	E
C	Pass operand to out	(, >	E C
)	Pass all operators down to first left parenthesis to out		E C >
||	Pass operator to stack	||	E C >
!	Pass operator to stack	||, !	E C >
(	Pass left parenthesis into stack	||, !, (	E C >
!	Pass operator to stack	||, !, (, !	E C >
(	Pass left parenthesis into stack	||, !, (, !, (	E C >
(	Pass left parenthesis into stack	||, !, (, !, (, (	E C >
D	Pass operand to out	||, !, (, !, (, (	E C > D
<	Pass operator to stack	||, !, (, !, (, (, <	E C > D
C	Pass operand to out	||, !, (, !, (, (, <	E C > D C
)	Pass all operators down to first left parenthesis to out	||, !, (, !, (	E C > D C <
||	Pass operator to stack	||, !, (, !, (, ||	E C > D C <
(	Pass left parenthesis into stack	||, !, (, !, (, ||, (	E C > D C <
C	Pass operand to out	||, !, (, !, (, ||, (	E C > D C < C
>	Pass operator to stack	||, !, (, !, (, ||, (, >	E C > D C < C
B	Pass operand to out	||, !, (, !, (, ||, (, >	E C > D C < C B
)	Pass all operators down to first left parenthesis to out	||, !, (, !, (, ||	E C > D C < C B >
)	Pass all operators down to first left parenthesis to out	||, !, (	E C > D C < C B > || !
&&	Pass operator to stack	||, !, (, &&	E C > D C < C B > || !
A	Pass operand to out	||, !, (, &&	E C > D C < C B > || ! A
)	Pass all operators down to first left parenthesis to out	||	E C > D C < C B > || ! A && !
			E C > D C < C B > || ! A && ! ||

Reverse expression: || ! && A ! || < B C > C D < C E

2- Evaluate

Infix notation: ! (A && ! ((B < C) | | (C > D))) | | (C < E)

Prefix notation: | | ! && A ! | | < B C > C D < C E

Let A = 1, B = 2, C = 3, D = 5, E = 3

Infix notation: ! (1 && ! ((2 < 3) | | (3 > 5))) | | (3 < 3)

Prefix notation: | | ! && 1 ! | | < 2 3 > 3 5 < 3 3

| | ! && 1 ! | | < 2 3 > 3 5 < 3 3	< 3 3	| | ! && 1 ! | | < 2 3 > 3 5 0
| | ! && 1 ! | | < 2 3 > 3 5 0	> 3 5	| | ! && 1 ! | | < 2 3 0 0
| | ! && 1 ! | | < 2 3 0 0	< 2 3	| | ! && 1 ! | | 1 0 0
| | ! && 1 ! | | 1 0 0	| | 1 0	| | ! && 1 ! 1 0
| | ! && 1 ! 1 0	! 1	| | ! && 1 0 0
| | ! && 1 0 0	&& 1 0	| | ! 0 0
| | ! 0 0	! 0	| | 1 0
| | 1 0	| | 1 0	1