## Q1- I) Convert to Postfix

Input	Postfix Stack	Operator Stack	Info
A+((B-C*D)/E)+F-G/H	Α		A is an operand
			append to output
A+((B-C*D)/E)+F-G/H	Α	+	+ is an operator
			append to operators
A+((B-C*D)/E)+F-G/H	Α	+(	( is an operator
			append to operators
A+((B-C*D)/E)+F-G/H	Α	+((	( is an operator
			append to operators
A+((B-C*D)/E)+F-G/H	AB	+((	B is an operand
			append to output
A+((B-C*D)/E)+F-G/H	AB	+((-	- is an operator
			append to operators
A+((B-C*D)/E)+F-G/H	ABC	+((-	C is an operand
			append to output
A+((B-C*D)/E)+F-G/H	ABC	+((-*	* is an operator
			append to operators
A+((B-C*D)/E)+F-G/H	ABCD	+((-*	D is an operand
			append to output
A+((B-C*D)/E)+F-G/H	ABCD*-	+(	Encountered ")", pop
			operators until next "("
A+((B-C*D)/E)+F-G/H	ABCD*-	+(/	/ is an operator
			append to operators
A+((B-C*D)/E)+F-G/H	ABCD*-E	+(/	E is an operand
			append to output
A+((B-C*D)/E)+F-G/H	ABCD*-E/	+	Encountered ")", pop
			operators until next "("
A+((B-C*D)/E)+F-G/H	ABCD*-E/+	+	+ is an operator with
			same precedence
			pop operators and
			push to output
A+((B-C*D)/E)+F-G/H	ABCD*-E/+F	+	F is an operand
. ((= 0.1=) (=)			append to output
A+((B-C*D)/E)+F-G/H	ABCD*-E/+F+	-	+ is an operator with
			same precedence
			pop operators and
			push to output
A+((B-C*D)/E)+F-G/H	ABCD*-E/+F+G	-	G is an operand
. (/2 2/2) /2 2 2 1		,	append to output
A+((B-C*D)/E)+F-G/H	ABCD*-E/+F+G	-/	/ is an operator
. (/= 0.1=) /=> = 0 // .		,	append to operators
A+((B-C*D)/E)+F-G/H	ABCD*-E/+F+GH	-/	H is an operand
			append to output
A+((B-C*D)/E)+F-G/H	ABCD*-E/+F+GH/-		No more items left
			Pop the operator
			stack until empty

## Q1- I) Evaluate the converted Postfix

Evaluating Postfix expression "ABCD\*-E/+F+GH/-" for: A=1, B=2, C=3, D=4, E=5, F=6, G=7, H=8

	<del></del>	, 0 7, 11 0	
Input	Operand Stack	Current Expression	Info
1234*-5/+6+7	3 1		1 is an operand
1 -			push to operand stack
1234*-5/+6+7	2 1		2 is an operand
1 -			push to operand stack
1234*-5/+6+7	3 2 1		3 is an operand
1 -			push to operand stack
1234*-5/+6+7	4321		4 is an operand
/ -			push to operand stack
1234*-5/+6+7	12 2 1	3 * 4 = 12	* is an operator
/ -			pop last 2 operands
			from stack and do
			expression
1234*-5/+6+7	-10 1	2 – 12 = -10	- is an operator
/ -			pop last 2 operands
			from stack and do
			expression
1234*-5/+6+7	5 -10 1		5 is an operand
/ -			push to operand stack
,			i i
1234*-5/+6+7	-2 1	-10 / 5 = -2	/ is an operator
<i> </i>			pop last 2 operands
			from stack and do
			expression
1234*-5/+6+7	-1	1 + -2 = -1	+ is an operator
/ -			pop last 2 operands
			from stack and do
			expression
1234*-5/+6+7	6 -1		6 is an operand
1 -			push to operand stack
·			,
1234*-5/+6+7	5	-1 + 6 = 5	+ is an operator
<i> </i>			pop last 2 operands
			from stack and do
			expression
1234*-5/+6+7	7 5		7 is an operand
/ -			push to operand stack
			,
1234*-5/+6+7	875		8 is an operand
/ -			push to operand stack
1234*-5/+6+7	0.875 5	7 / 8 = 0.875	/ is an operator
<i> </i>			pop last 2 operands
			from stack and do

			expression
1234*-5/+6+78	4.125	5 – 0.875 = 4.125	- is an operator
<i>/</i> -			pop last 2 operands
			from stack and do
			expression

### Q1- I) Convert to Prefix

To find the prefix of expression: A+((B-C\*D)/E)+F-G/H

1. Reverse the string: H/G-F+(E/(D\*C-B))+A

2. Find postfix of this reversed expression:

Input	Postfix Stack	Operator Stack
H/G-F+(E/(D*C-B))+A	Н	
H/G-F+(E/(D*C-B))+A	Н	1
H/G-F+(E/(D*C-B))+A	HG	1
H/G-F+(E/(D*C-B))+A	HG/	-
H/G-F+(E/(D*C-B))+A	HG/F	-
H/G-F+(E/(D*C-B))+A	HG/F-	+
H/G-F+ <mark>(</mark> E/(D*C-B))+A	HG/F-	+(
H/G-F+(E/(D*C-B))+A	HG/F-E	+(
H/G-F+(E <mark>/</mark> (D*C-B))+A	HG/F-E	+(/
H/G-F+(E/(D*C-B))+A	HG/F-E	+(/(
H/G-F+(E/(D*C-B))+A	HG/F-ED	+(/(
H/G-F+(E/(D*C-B))+A	HG/F-ED	+(/(*
H/G-F+(E/(D*C-B))+A	HG/F-EDC	+(/(*
H/G-F+(E/(D*C-B))+A	HG/F-EDC*	+(/(-
H/G-F+(E/(D*C-B))+A	HG/F-EDC*B	+(/(-
H/G-F+(E/(D*C-B))+A	HG/F-EDC*B-	+(/
H/G-F+(E/(D*C-B))+A	HG/F-EDC*B-/	+
H/G-F+(E/(D*C-B))+A	HG/F-EDC*B-/+	+
H/G-F+(E/(D*C-B))+A	HG/F-EDC*B-/+A	+
H/G-F+(E/(D*C-B))+A	HG/F-EDC*B-/+A+	

3. Now we reverse the postfix expression of previously reversed expression:

## +A+/-B\*CDE-F/GH

#### Q1- I) Evaluate the converted Prefix

Evaluating Prefix expression "+A+/-B\*CDE-F/GH" for:

A=1, B=2, C=3, D=4, E=5, F=6, G=7, H=8

Input	Operand Stack	Expression
+1+/-2*345-6/78	8	
+1+/-2*345-6/78	8 7	
+1+/-2*345-6/78	0,875	7 / 8 = 0,875
+1+/-2*345-6/78	0,875 6	
+1+/-2*345-6/78	5,125	6 - 0,875 = 5,125
+1+/-2*345-6/78	5,125 5	
+1+/-2*345-6/78	5,125 5 4	
+1+/-2*345-6/78	5,125 5 4 3	
+1+/-2*345-6/78	5,125 5 12	4 * 3 = 12
+1+/-2*345-6/78	5,125 5 12 2	
+1+/-2*345-6/78	5,125 5 -10	2 – 12 = -10
+1+/-2*345-6/78	5,125 -2	-10 / 5 = -2
+1+/-2*345-6/78	3,125	-2 + 5,125 = 3,125
+ 1 + / - 2 * 3 4 5 - 6 / 7 8	3,125 1	
+1+/-2*345-6/78	4,125	1 + 3,125 = 4,125

#### Q1- II) Convert to Postfix

Input	Postfix Stack	Operator Stack	Info
!(A&&!((B <c)  < td=""><td></td><td>!</td><td>! is an operator</td></c)  <>		!	! is an operator
(C>D)))  (C <e)< td=""><td></td><td></td><td>append to operators</td></e)<>			append to operators
! <mark>(</mark> A&&!((B <c)  < td=""><td></td><td>!(</td><td>( is an operator</td></c)  <>		!(	( is an operator
(C>D)))  (C <e)< td=""><td></td><td></td><td>append to operators</td></e)<>			append to operators
!(A&&!((B <c)  < td=""><td>A</td><td>!(</td><td>A is an operand</td></c)  <>	A	!(	A is an operand
(C>D)))  (C <e)< td=""><td></td><td></td><td>append to output</td></e)<>			append to output
!(A&&!((B <c)  < td=""><td>A</td><td>!(&amp;&amp;</td><td>&amp;&amp; is an operator</td></c)  <>	A	!(&&	&& is an operator
(C>D)))  (C <e)< td=""><td></td><td></td><td>append to operators</td></e)<>			append to operators
!(A&&!((B <c)  < td=""><td>A</td><td>!(&amp;&amp;!</td><td>! is an operator</td></c)  <>	A	!(&&!	! is an operator
(C>D)))  (C <e)< td=""><td></td><td></td><td>append to operators</td></e)<>			append to operators
!(A&&! <mark>(</mark> (B <c)  < td=""><td>A</td><td>!(&amp;&amp;!(</td><td>( is an operator</td></c)  <>	A	!(&&!(	( is an operator
(C>D)))  (C <e)< td=""><td></td><td></td><td>append to operators</td></e)<>			append to operators
!(A&&!( <mark>(</mark> B <c)  < td=""><td>A</td><td>!(&amp;&amp;!((</td><td>( is an operator</td></c)  <>	A	!(&&!((	( is an operator
(C>D)))  (C <e)< td=""><td></td><td></td><td>append to operators</td></e)<>			append to operators
!(A&&!((B <c)  < td=""><td>AB</td><td>!(&amp;&amp;!((</td><td>B is an operand</td></c)  <>	AB	!(&&!((	B is an operand
(C>D)))  (C <e)< td=""><td></td><td></td><td>append to output</td></e)<>			append to output

AB	!(&&!((<	< is an operators
		append to operators
ABC	!(&&!((<	C is an operand
		append to output
ABC<	!(&&!(	Encountered ")"pop
		the operator stack
		until next "(".
ABC<	!(&&!(	is an operator
		append to operators
ABC<	!(&&!(  (	( is an operator
		append to operators
ABC <c< td=""><td>!(&amp;&amp;!(  (</td><td>C is an operand</td></c<>	!(&&!(  (	C is an operand
		append to output
ABC <c< td=""><td>!(&amp;&amp;!(  (&gt;</td><td>&gt; is an operator</td></c<>	!(&&!(  (>	> is an operator
		append to operators
ABC <cd< td=""><td>!(&amp;&amp;!(  (&gt;</td><td>D is an operand</td></cd<>	!(&&!(  (>	D is an operand
		append to output
ABC <cd></cd>	!(&&!(	Encountered ")"pop
		the operator stack
		until next "(".
ABC <cd>  </cd>	!(&&!	Encountered ")"pop
		the operator stack
		until next "(".
ABC <cd>  !&amp;&amp;</cd>	!	Encountered ")"pop
		the operator stack
		until next "(".
ABC <cd>  !&amp;&amp;!</cd>		is an operator with
		less precedence pop
		operators and push to
		output
ABC <cd>  !&amp;&amp;!</cd>	(	( is an operator
		append to operators
ABC <cd>  !&amp;&amp;!C</cd>	(	C is an operand
		append to output
ABC <cd>  !&amp;&amp;!C</cd>	(<	< is an operator
		append to operators
ABC <cd>  !&amp;&amp;!CE</cd>	(<	E is an operand
		append to output
BC <cd>  !&amp;&amp;!CE&lt;  </cd>		Encountered ")"pop
		the operator stack
		until next "(".
3C <cd>  !&amp;&amp;!CE&lt;  </cd>		Empty the operator
		stack
	ABC   CD   ABC   A	ABC !(&&!((< ABC< !(&&!()   ABC< !(&&!()   ABC< !(&&!()   ABC <c !(&&!() ="" !(&&!() ()="" abc<c="" abc<cd="">   !(&amp;&amp;!()   ABC<cd>   !(&amp;&amp;!()   ABC<d  abc<d<="" abc<d ="" td=""></d ></cd></cd></cd></cd></cd></cd></cd></cd></cd></cd></cd></cd></cd></cd></cd></cd></cd></cd></cd></cd></cd></cd></cd></cd></cd></cd></cd></cd></cd></cd></cd></cd></cd></cd></cd></cd></cd></cd></cd></cd></cd></cd></cd></cd></cd></cd></cd></cd></cd></cd></cd></cd></c>

#### **Q1-II)** Evaluate the converted Postfix

Evaluating Postfix expression "ABC<CD>||!&&!CE<||" for:

Input	Operand Stack	Expression
101<10>  !&&!11<	1	
1 0 1 < 1 0 >    ! && ! 1 1 <	10	
101<10>  !&&!11<	101	
101<10>  ! &&!11<	11	0 < 1 = 1
101<10>  !&&!11<	111	
101<10>   ! && ! 11<	1110	
101<10>  !&&!11<	111	1 > 0 = 1
101<10>  !&&!11<	11	1    1 = 1
101<10>	10	!1 = 0
101<10>  ! &&!11<	0	1 && 0 = 0
101<10>  !&&!11<	1	!0
101<10>  !&&!11<	11	
101<10>  !&&!11<	111	
101<10>  !&&!11<	10	1 < 1 = 0
101<10>  !&&!11<	1	1    0 = 1

Infix version of expression was: !(A&&!((B<C)||(C>D)))||(C<E)

```
With values above: !(1&\&!((0<1)||(1>0)))||(1<1) !(1&\&!(1||(1)))||0 !(1&\&!1)||0 !(1&\&0)||0 !(0)||0 1||0 1
```

#### **Evaluations holds up.**

#### Q1- II) Convert to Prefix

To find the prefix of expression: !(A&&!((B<C)||(C>D)))||(C<E)

1. Reverse the string:  $(E < C) \| ((D > C) \| (C < B))! \& A)!$ 

2. Find postfix of this reversed expression:

Input	Postfix Stack	Operator Stack
(E <c)  (((d>C)   (C<b))!&&a)!< td=""><td></td><td>(</td></b))!&&a)!<></c)  (((d>		(
(E <c)  (((d>C)   (C<b))!&&a)!< td=""><td>Е</td><td>(</td></b))!&&a)!<></c)  (((d>	Е	(
(E <c)  (((d>C)   (C<b))!&&a)!< td=""><td>E</td><td>(&lt;</td></b))!&&a)!<></c)  (((d>	E	(<
(E <c)  (((d>C)   (C<b))!&&a)!< td=""><td>EC</td><td>(&lt;</td></b))!&&a)!<></c)  (((d>	EC	(<
(E <c)  (((d>C)  </c)  (((d>	EC<	

(C <b))!&&a)!< th=""><th></th><th></th></b))!&&a)!<>		
(E <c)  (((d>C)   (C<b))!&&a)!< td=""><td>EC&lt;</td><td>II</td></b))!&&a)!<></c)  (((d>	EC<	II
(E <c)  (((d>C)   (C<b))!&&a)!< td=""><td>EC&lt;</td><td>  (</td></b))!&&a)!<></c)  (((d>	EC<	(
(E <c)  (((d>C)   (C<b))!&&a)!< td=""><td>EC&lt;</td><td>  ((</td></b))!&&a)!<></c)  (((d>	EC<	((
(E <c)  (((d>C)   (C<b))!&&a)!< td=""><td>EC&lt;</td><td>11(((</td></b))!&&a)!<></c)  (((d>	EC<	11(((
(E <c)  (((<mark>D&gt;C)   (C<b))!&&a)!< td=""><td>EC<d< td=""><td>11(((</td></d<></td></b))!&&a)!<></c)  (((<mark>	EC <d< td=""><td>11(((</td></d<>	11(((
(E <c)  (((d>C)   (C<b))!&&a)!< td=""><td>EC<d< td=""><td>  (((&gt;</td></d<></td></b))!&&a)!<></c)  (((d>	EC <d< td=""><td>  (((&gt;</td></d<>	(((>
(E <c)  (((d><mark>C</mark>)   (C<b))!&&a)!< td=""><td>EC<dc< td=""><td>  (((&gt;</td></dc<></td></b))!&&a)!<></c)  (((d>	EC <dc< td=""><td>  (((&gt;</td></dc<>	(((>
(E <c)  (((d>C)   (C<b))!&&a)!< td=""><td>EC<dc></dc></td><td>  ((</td></b))!&&a)!<></c)  (((d>	EC <dc></dc>	((
(E <c)  (((d>C)   (C<b))!&&a)!< td=""><td>EC<dc></dc></td><td>II((II</td></b))!&&a)!<></c)  (((d>	EC <dc></dc>	II((II
(E <c)  (((d>C)   (C<b))!&&a)!< td=""><td>EC<dc></dc></td><td>  ((  (</td></b))!&&a)!<></c)  (((d>	EC <dc></dc>	((  (
(E <c)  (((d>C)   (C<b))!&&a)!< td=""><td>EC<dc>C</dc></td><td>  ((  (</td></b))!&&a)!<></c)  (((d>	EC <dc>C</dc>	((  (
(E <c)  (((d>C)   (C<b))!&&a)!< td=""><td>EC<dc>C</dc></td><td>  ((  (&lt;</td></b))!&&a)!<></c)  (((d>	EC <dc>C</dc>	((  (<
(E <c)  (((d>C)   (C&lt;<mark>B</mark>))!&amp;&amp;A)!</c)  (((d>	EC <dc>CB</dc>	((  (<
(E <c)  (((d>C)   (C<b))!&&a)!< td=""><td>EC<dc>CB&lt;</dc></td><td>II((II</td></b))!&&a)!<></c)  (((d>	EC <dc>CB&lt;</dc>	II((II
(E <c)  (((d>C)   (C<b))!&&a)!< td=""><td>EC<dc>CB&lt;  </dc></td><td>II(</td></b))!&&a)!<></c)  (((d>	EC <dc>CB&lt;  </dc>	II(
(E <c)  (((d>C)   (C<b))!&&a)!< td=""><td>EC<dc>CB&lt;  </dc></td><td>  (!</td></b))!&&a)!<></c)  (((d>	EC <dc>CB&lt;  </dc>	(!
(E <c)  (((d>C)   (C<b))!<mark>&amp;&amp;A)!</b))!<mark></c)  (((d>	EC <dc>CB&lt;  !</dc>	(&&
(E <c)  (((d>C)   (C<b))!&&a)!< td=""><td>EC<dc>CB&lt;  !A</dc></td><td>  (&amp;&amp;</td></b))!&&a)!<></c)  (((d>	EC <dc>CB&lt;  !A</dc>	(&&
(E <c)  (((d>C)   (C<b))!&&a)!< td=""><td>EC<dc>CB&lt;  !A&amp;&amp;</dc></td><td>II</td></b))!&&a)!<></c)  (((d>	EC <dc>CB&lt;  !A&amp;&amp;</dc>	II
(E <c)  (((d>C)   (C<b))!&&a)!< td=""><td>EC<dc>CB&lt;  !A&amp;&amp;</dc></td><td>  !</td></b))!&&a)!<></c)  (((d>	EC <dc>CB&lt;  !A&amp;&amp;</dc>	!
(E <c)  (((d>C)   (C<b))!&&a)!< td=""><td>EC<dc>CB&lt;  !A&amp;&amp;!  </dc></td><td></td></b))!&&a)!<></c)  (((d>	EC <dc>CB&lt;  !A&amp;&amp;!  </dc>	

3. Now we reverse the postfix expression of previously reversed expression:

# ||!&&A!||<BC>CD<CE

### Q1- II) Evaluate the converted Prefix

Evaluating Prefix expression "||!&&A!||<BC>CD<CE" for: A=1, B=0, C=1, D=0, E=1

Input	Operand Stack	Expression
! && 1!   < 0 1 > 1 0 < 1 1	1	
! && 1 !    < 0 1 > 1 0 < <mark>1</mark> 1	11	
! && 1!    < 0 1 > 1 0 < 1 1	0	1 < 1 = 0
! && 1 !    < 0 1 > 1 <mark>0</mark> < 1 1	0 0	
! && 1 !    < 0 1 > <mark>1</mark> 0 < 1 1	001	
! && 1!    < 0 1 > 1 0 < 1 1	0 1	1 > 0 = 1
! && 1 !    < 0 <mark>1</mark> > 1 0 < 1 1	011	
! && 1!   < 0 1 > 1 0 < 1 1	0110	
! && 1 !    < 0 1 > 1 0 < 1 1	011	0 < 1 = 1
! && 1!    < 0 1 > 1 0 < 1 1	0 1	1    1 = 1
! && 1!   < 0 1 > 1 0 < 1 1	0 0	!1 = 0
! && <mark>1</mark> !    < 0 1 > 1 0 < 1 1	001	
! && 1 !    < 0 1 > 1 0 < 1 1	0 0	0 && 1
! && 1 !    < 0 1 > 1 0 < 1 1	0 1	!0 = 1
! && 1 !    < 0 1 > 1 0 < 1 1	1	0    1

Which again, holds with the evaluation of infix version.