GIT Department of Computer Engineering
CSE 222/505- Spring 2020
Homework 4

Q1:

i) A + ((B – C * D) / E) + F – G / H

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

Solutions:

i) A + ((B – C * D) / E) + F – G / H

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

I reverse the string before the conversion of the infix expression to prefix expression.
Then I do conversion as conversion of the infix expression to postfix expression.
The new infix expression : H / G – F + (E / (D * C – B)) + A
The result for reversed string is: H G / F – E D C * B - / + A +

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

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

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

I reverse the string before the conversion of the infix expression to prefix expression.
Then I do conversion as conversion of the infix expression to postfix expression.
The new infix expression : (E > C) | | (((D < C) | | (C > B)) ! & & A) !
The result for reversed string is: E C > D C < C B > | | ! A & & ! | |

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

Next Token A+((B – C *D)/E)+F– G/H	Action	Effect on operatorStack	Effect on postfix
A	Append A to postfix.		A
+	The stack is empty Push + onto the stack		A
(	predence(( ) > predence(+), Push (onto the stack		A
(	Push (onto the stack		A
B	Append B to postfix		A B
-	Push – onto the stack		A B
C	Append C to postfix		A B C
*****	predence (*) > predence(-) Push * onto the stack		A B C
D	Append D to postfix		A B C D

)	Pop *, - , (off of stack and append * and - to postfix		A B C D * -
/	Push / onto the stack		A B C D * -
E	Append E to postfix		A B C D * - E
)	Pop / and (off of stack and append to / postfix		A B C D * - E /
+	predence (+) equals predence(+) Pop + off of stack and append to postfix		A B C D * - E / +
+	The stack is empty Push + onto the stack		A B C D * - E / +
F	Append F to postfix		A B C D * - E / + F
-	Predence (-) equals Predence(+) Pop + off of stack and append to postfix		A B C D * - E / + F +
-	The stack is empty Push - onto the stack		A B C D * - E / + F +
G	Append G to postfix		A B C D * - E / + F + G
/	predence(/) > predence(-), Push / onto the stack		A B C D * - E / + F + G
H	Append H to postfix		A B C D * - E / + F + G H
End of input	Stack is not empty, Pop / off of stack and append to postfix		A B C D * - E / + F + G H /
End of input	Stack is not empty, Pop - off of stack and append to postfix		A B C D * - E / + F + G H / -

Next Token H/G–F+(E/(D*C–B))+A	Action	Effect on operatorStack	Effect on Prefix (as postfix)
H	Append H to prefix.		H
/	The stack is empty Push / onto the stack	/	H
G	Append G to prefix.	/	H G
-	Predence(-) < Predence(/), Pop / off of stack and Append to prefix.		H G /
-	The stack is empty Push – onto the stack	-	H G /
F	Append F to prefix.	-	H G / F
+	Predence(+) equals Predence(-), Pop – off of stack and Append to prefix		H G / F -
+	The stack is empty Push + onto stack	+	H G / F -
(	Push (onto stack	(+	H G / F -
E	Append E to prefix	(+	H G / F - E
/	Push / onto stack	/ (+	H G / F - E
(	Push (onto stack	(/ (+	H G / F - E
D	Append D to prefix	(/ (+	H G / F – E D
*****	Push * onto stack	* (/ (+	H G / F – E D

C	Append C to prefix		H G / F – E D C
-	Predence(-) < predence(*), Pop * off the stack and append to prefix		H G / F – E D C *
-	Push – onto stack		H G / F – E D C *
B	Append B to stack		H G / F – E D C * B
)	Pop - and (off of stack and append – to postfix		H G / F – E D C * B -
)	Pop / and (off of stack and append / to postfix		H G / F – E D C * B - /
+	Predence(+) equals Predence(+), Pop – off of stack and Append to prefix		H G / F – E D C * B - / +
+	The stack is empty Push + onto the stack		H G / F – E D C * B - / +
A	Append A to prefix		H G / F – E D C * B - / + A
End of input	Stack is not empty, Pop + off the stack and Append to prefix		H G / F – E D C * B - / + A +

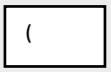
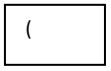
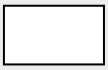
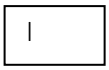
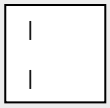
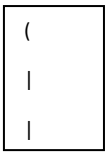
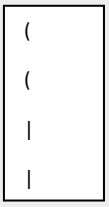
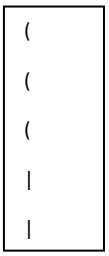
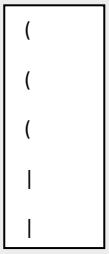
Next Token ***! (A && ! ((B < C) || (C > D))) || (C < E)***	Action	**Effect on** operatorStack	**Effect on** Postfix
!	The stack is empty Push ! onto stack	!	
(	Push (onto stack	(!	
A	Append A to postfix	(!	A
&	Push & onto stack	& (!	A
&	Push & onto stack	& & (!	A
!	Predence(!) > Predence(&) , Push ! onto stack	! & & (!	A
(	Push (onto stack	(! & & (!	A
(	Push (onto stack	((! & & (!	A

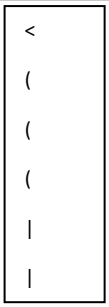
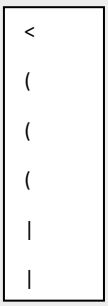
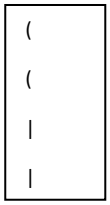
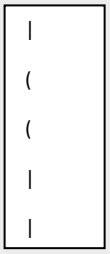
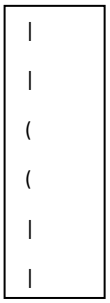
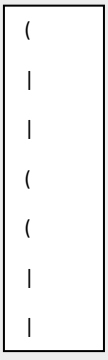
B	Append B to postfix		A B
<	Push < onto stack		A B
C	Append C to postfix		A B C
)	Pop < and) off the stack And append < to postfix		A B C <

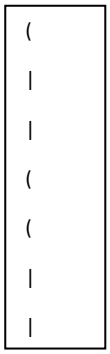
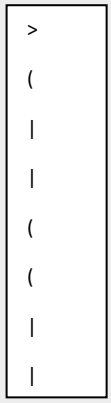
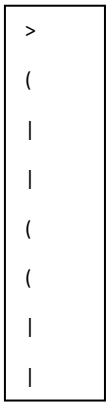
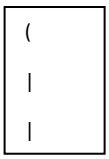
|	Push | onto stack		A B C <
|	Push | onto stack		A B C <
(	Push (onto stack		A B C <
C	Append C to postfix		A B C < C

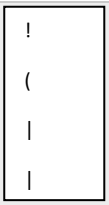
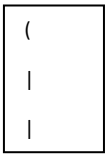
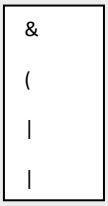
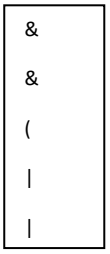
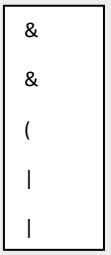
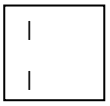
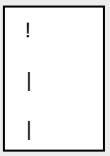
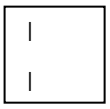
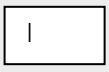
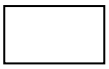
>	Push > onto stack		A B C < C
D	Append D to postfix		A B C < C D
)	Pop > and (off the stack And append > to postfix		A B C < C D >
)	Pop | , | and) off the stack And append | and | to postfix		A B C < C D > | |

)	Pop ! , & , & and (off the stack and append !, & and & to postfix		A B C < C D > | | ! & &
|	The stack is empty Push | onto stack		A B C < C D > | | ! & & !
|	Push _ onto stack		A B C < C D > | | ! & & !
(	Push (onto stack		A B C < C D > | | ! & & !
C	Append C to postfix		A B C < C D > | | ! & & ! C
<	Push < onto stack		A B C < C D > | | ! & & ! C
E	Append E to postfix		A B C < C D > | | ! & & ! C E
)	Pop < and (off the stack And append < to postfix		A B C < C D > | | ! & & ! C E <
End of input	Stack is not empty, Pop | off the stack and Append to postfix		A B C < C D > | | ! & & ! C E < |
End of input	Stack is not empty, Pop | off the stack and Append to postfix		A B C < C D > | | ! & & ! C E < | |

Next Token *(E > C) | | (((D < C) | | (C > B)) ! & & A) !*	Action	**Effect on** operatorStack	**Effect on** Prefix(as postfix)
(	Push (onto stack		
E	Append E to prefix		E
>	Push > onto stack		E
C	Append C to prefix		E C
)	Pop > and (off the stack And append > to prefix		E C >
|	Stack is empty, Push | onto stack		E C >
|	Push | onto stack		E C >
(	Push (onto stack		E C >
(	Push (onto stack		E C >
(	Push (onto stack		E C >
D	Append D to prefix		E C > D

<	Push < onto stack		E C > D
C	Append C to prefix		E C > D C
)	Pop < and (off the stack And append < to prefix		E C > D C <
|	Push | onto stack		E C > D C <
|	Push | onto stack		E C > D C <
(	Push (onto stack		E C > D C <

C	Append C to prefix		E C > D C < C
>	Push > onto stack		E C > D C < C
B	Append B to prefix		E C > D C < C B
)	Pop > and (off the stack and append > to prefix		E C > D C < C B >
)	Pop | , | and (off the stack and append | and | to prefix		E C > D C < C B > | |

!	Push ! onto stack		E C > D C < C B > | |
&	Predence(&) < predence(&), Pop ! off the stack and append to prefix		E C > D C < C B > | | !
&	Push & onto stack		E C > D C < C B > | | !
&	Push & onto stack		E C > D C < C B > | | !
A	Append A to prefix		E C > D C < C B > | | ! A
)	Pop & and & off the stack and append & and & to prefix		E C > D C < C B > | | ! A & &
!	Predence(!) > predence(|), Push ! onto stack		E C > D C < C B > | | ! A & &
End of input	Stack is not empty, Pop ! off the stack and append to prefix		E C > D C < C B > | | ! A & & !
End of input	Stack is not empty, Pop | off the stack and append to prefix		E C > D C < C B > | | ! A & & ! |
End of input	Stack is not empty, Pop | off the stack and append to prefix		E C > D C < C B > | | ! A & & ! | |

EVALUATION:

i) A + ((B – C * D) / E) + F – G / H
A = 12, B = 38, C = 4, D = 7, E = 5, F = 9, G = 26, H = 13.

Infix expression : 12 + ((38 – 4*7) / 5) + 9 – 26 / 13 = 21

Postfix expression : 12 38 4 7 * - 5 / + 9 + 26 13 / -

Expression	Action	Stack
12 38 4 7 * - 5 / + 9 + 26 13 / -	Push 12	12
12 38 4 7 * - 5 / + 9 + 26 13 / -	Push 38	38 12
12 38 4 7 * - 5 / + 9 + 26 13 / -	Push 4	4 38 12
12 38 4 7 * - 5 / + 9 + 26 13 / -	Push 7	7 4 38 12
12 38 4 7 * - 5 / + 9 + 26 13 / -	Pop 7 and 4 Evaluate 4 * 7 Push 28	28 38 12
12 38 4 7 * - 5 / + 9 + 26 13 /	Pop 28 and 38 Evaluate 38 – 28 Push 10	10 12
12 38 4 7 * - 5 / + 9 + 26 13 / -	Push 5	5 10 12
12 38 4 7 * - 5 / + 9 + 26 13 / -	Pop 5 and 10 Evaluate 10 / 5 Push 2	2 12
12 38 4 7 * - 5 / + 9 + 26 13 / -	Pop 2 and 12 Evaluate 12 + 2 Push 14	14
12 38 4 7 * - 5 / + 9 + 26 13 / -	Push 9	9 14
12 38 4 7 * - 5 / + 9 + 26 13 / -	Pop 9 and 14 Evaluate 14 + 9 Push 23	23
12 38 4 7 * - 5 / + 9 + 26 13 / -	Push 26	26 23

12 38 4 7 * - 5 / + 9 + 26 13 / -	Push 13	13 26 23
12 38 4 7 * - 5 / + 9 + 26 13 / -	Pop 13 and 26 Evaluate 26 / 13 Push 2	2 23
12 38 4 7 * - 5 / + 9 + 26 13 / -	Pop 2 and 23 Evaluate 23 – 2 Push 21	21
12 38 4 7 * - 5 / + 9 + 26 13 / -	Pop 21 Stack is empty Result is 21	

Prefix expression : + 12 + / - 38 * 4 7 5 – 9 / 26 13 (It starts from the end of the expression.)

Expression	Action	Stack
+ 12 + / - 38 * 4 7 5 – 9 / 26 13	Push 13	13
+ 12 + / - 38 * 4 7 5 – 9 / 26 13	Push 26	26 13
+ 12 + / - 38 * 4 7 5 – 9 / 26 13	Pop 26 and 13 Evaluate 26 / 13 Push 2	2
+ 12 + / - 38 * 4 7 5 – 9 / 26 13	Push 9	9 2
+ 12 + / - 38 * 4 7 5 – 9 / 26 13	Pop 9 and 2 Evaluate 9 – 2 Push 7	7
+ 12 + / - 38 * 4 7 5 – 9 / 26 13	Push 5	5 7
+ 12 + / - 38 * 4 7 5 – 9 / 26 13	Push 7	7 5 7
+ 12 + / - 38 * 4 7 5 – 9 / 26 13	Push 4	4 7 5 7
+ 12 + / - 38 * 4 7 5 – 9 / 26 13	Pop 4 and 7 Evaluate 4 * 7 Push 28	28 5 7
+ 12 + / - 38 * 4 7 5 – 9 / 26 13	Push 38	38 28 5 7

+ 12 + / - 38 * 4 7 5 – 9 / 26 13	Pop 38 and 28 Evaluate 38 – 28 Push 10	10 5 7
+ 12 + / - 38 * 4 7 5 – 9 / 26 13	Pop 10 and 5 Evaluate 10 / 5 Push 2	2 7
+ 12 + / - 38 * 4 7 5 – 9 / 26 13	Pop 2 and 7 Evaluate 2 + 7 Push 9	9
+ 12 + / - 38 * 4 7 5 – 9 / 26 13	Push 12	12 9
+ 12 + / - 38 * 4 7 5 – 9 / 26 13	Pop 12 and 9 Evaluate 12 + 9 Push 21	21
	Peek 21 Result is 21	

ii) ! (A && ! ((B < C) || (C > D))) || (C < E)
A = 67, B = 12, C = 17, D = 8, E = 25

Infix expression : ! (67 && !((12 < 17) || (17 > 8))) || (17 < 25) = 1

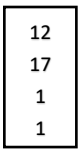
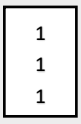
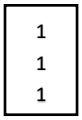
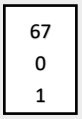
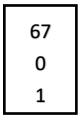
Postfix : 67 12 17 < 17 8 > | | ! & & ! 17 25 < | |

Expression	Action	Stack
67 12 17 < 17 8 > | | ! & & ! 17 25 < | |	Push 67	67
67 12 17 < 17 8 > | | ! & & ! 17 25 < | |	Push 12	12 67
67 12 17 < 17 8 > | | ! & & ! 17 25 < | |	Push 17	17 12 67
67 12 17 < 17 8 > | | ! & & ! 17 25 < | |	Pop 17 and 12 Evaluate 12 < 17 Push 1	1 67
67 12 17 < 17 8 > | | ! & & ! 17 25 < | |	Push 17	17 1 67
67 12 17 < 17 8 > | | ! & & ! 17 25 < | |	Push 8	8 17 1 67
67 12 17 < 17 8 > | | ! & & ! 17 25 < | |	Pop 8 and 17 Evaluate 17 > 8 Push 1	1 1 67

67 12 17 < 17 8 > | | ! & & ! 17 25 < | | 	(Go to next character)	1 1 67
67 12 17 < 17 8 > | | ! & & ! 17 25 < | | 	(if it is |) Pop 1 and 1 Evaluate 1 || 1 Push 1	1 67
67 12 17 < 17 8 > | | ! & & ! 17 25 < | | 	Pop 1 Evaluate ! 1 Push 0	0 67
67 12 17 < 17 8 > | | ! & & ! 17 25 < | | 	(Go to next character)	0 67
67 12 17 < 17 8 > | | ! & & ! 17 25 < | | 	(if it is &) Pop 0 and 67 Evaluate 67 && 0 Push 0	0
67 12 17 < 17 8 > | | ! & & ! 17 25 < | | 	Pop 0 Evaluate ! 0 Push 1	1
67 12 17 < 17 8 > | | ! & & ! 17 25 < | | 	Push 17	17 1
67 12 17 < 17 8 > | | ! & & ! 17 25 < | | 	Push 25	25 17 1
67 12 17 < 17 8 > | | ! & & ! 17 25 < | | 	Pop 25 and 17 Evaluate 17 < 25 Push 1	1 1
67 12 17 < 17 8 > | | ! & & ! 17 25 < | | 	(Go to next character)	1 1
67 12 17 < 17 8 > | | ! & & ! 17 25 < | | 	(if it is |) Pop 1 and 1 Evaluate 1 || 1 Push 1	1
67 12 17 < 17 8 > | | ! & & ! 17 25 < | | 	Pop 1 Stack is empty Result is 1	

Prefix : | | ! & & 67 ! | | < 12 17 > 17 8 < 17 25 (It starts from the end of the expression.)

Expression	Action	Stack
| | ! & & 67 ! | | < 12 17 > 17 8 < 17 25 	Push 25	25
| | ! & & 67 ! | | < 12 17 > 17 8 < 17 25 	Push 17	17 25
| | ! & & 67 ! | | < 12 17 > 17 8 < 17 25 	Pop 17 and 25 Evaluate 17 < 25 Push 1	1
| | ! & & 67 ! | | < 12 17 > 17 8 < 17 25 	Push 8	8 1

	Push 17	
	Pop 17 and 8 Evaluate 17 > 8 Push 1	
	Push 17	
	Push 12	
	Pop 12 and 17 Evaluate 12 < 17 Push 1	
	(Go to next character)	
	(if it is |) Pop 1 and 1 Evaluate 1 || 1 Push 1	
	Pop 1 Evaluate ! 1 Push 0	
	Push 67	
	(Go to next character)	
	(if it is &) Pop 67 and 0 Evaluate 67 && 0 Push 0	
	Pop 0 Evaluate ! 0 Push 1	
	(Go to next character)	
	(if it is |) Pop 1 and 1 Evaluate 1 || 1 Push 1	
	Peek 1 Result is 1	