Infix to postfix Sums

1. A + B * C

Step	input	stack	output
1.	A+B*C	-	-
2.	+B*C	-	A
3.	B*C	+	A
4.	*C	+	AB
5.	C	+*	AB
6.		+*	ABC
7.		+	ABC *
8.			ABC * +

2. (A+B) * (C-D)

step	input	stack	output
1.	(A+B)*(C-D)	-	-
2.	A+B)*(C-D)	(	-
3.	+B)*(C-D)	(	A
4.	B)*(C-D)	C+	A
5.	)*(C-D)	(+	AB
6.	*(C-D)		AB+
7.	(C-D)	*	AB+
8.	C-D)	*(	AB+
9.	-D)	*C-	AB+C
10.	D)	*(-	AB+C
11.	)	*	AB+CD-

12. AB+CD*

Postfix to Infix

① AB+C*

step	postfix	stack
1.	AB+C*	[ ]
2.	B+C*	[A]
3.	+C*	[A, B]
4.	C*	[(A+B)]
5.	*	[(A+B), C]
6.		[((A+B)*C)]

② ABC*+D

step	postfix	stack
1.	ABC*+D	[ ]
2.	BC*+D-	A
3.	C*+D-	A, B
4.	*+D-	A, B, C
5.	+D-	A, (B*C)
6.	D-	[(A+B*C)]
7.	-	[(A+(B*C)), D]
8.		[(A+(B*C))-D]

Balancing parenthesis

(A+B) * (C-D)

Step	Read character	stack
1.		
2.	(	[(]
3.	A	[(]
4.	+	[(]
5.	B	[(]
6.	)	[ ]
7.	*	[ ]
8.	(	[(]
9.	(	[(]
10.	-	[(]
11.	D	[(]
	)	[ ]

Stack is empty.

{A+(B*C)-D})

Step	Read character	stack
1.		
2.	{	[{]
3.	A	[{]
4.	+	[{]
5.	(	[{,(]
6.	B	[{,(]
7.	)	[{]
8.	D	[{]
	}	[ ]
	)	[)]

Stack not empty.