2.b) Find the FIRST and FOLLOW sets for each non-terminal of the grammar given below:

$S \rightarrow ABa / bCA$

$A \rightarrow cBCD / \epsilon$

$B \rightarrow CdA / ad$

$C \rightarrow eC / \epsilon$

$D \rightarrow bSF / a$

Sol

FIRST(D) $\rightarrow \{b, a\}$

FIRST(C) $\rightarrow \{e, \epsilon\}$

FIRST(B) $\rightarrow \{e, d\} \cup \{a, d\}$

FIRST(B) $\rightarrow \{e, d, a\}$

FIRST(A) $\rightarrow \{c, \epsilon\}$

FIRST(S) → FIRST(ABa) ∪ FIRST(BCa)
∪ {b}

FIRST(ABa) → FIRST(A) - {ϵ} ∪ FIRST(Ba)
→ {c, ϵ} - {ϵ} ∪ {e, d, a}
→ {c, e, d, a}

FIRST(BCa) → FIRST

FIRST(S) → {c, e, d, a, b}

FOLLOW(S) → {$, F}

FOLLOW(A) → {e, d, a} ∪ {$, F} ∪ {a, e, b}
→ {a, b, d, e, $, F}

FOLLOW(B) → {a, e, b}

FOLLOW(C) → {a, b, c, d}

FOLLOW(D) → {a, b, d, e, F, $}

3. Construct an LALR (1) parsing table for the following grammar.

$S \to Aa / aAC / Bc / bBa$

$A \to d$

$B \to d$

Sol Step 1: Argumented Grammar

I_0:

$S1 \to \cdot S$, \$

$S \to \cdot Aa$, \$

$S \to \cdot aAC$, \$

$S \to \cdot Bc$, \$

$S \to \cdot bBa$, \$

$A \to \cdot d$, a/c

$B \to \cdot d$, c/a

goto $(I_0, S) \to \{S1 \to S\cdot, \$\}$ I_1

goto $(I_0, A) \to \{S \to A\cdot a, \$\}$ I_2

goto $(I_0, B) \to \{S \to B\cdot c, \$\}$ I_3

goto $(I_0, a) \to \{S \to a\cdot AC, A \to \cdot d, a/c\}$ I_4

goto $(I_0, b) \to \{S \to b\cdot Ba, \$, B \to \cdot d, c/a\}$ I_5

goto $(I_0, d) \rightarrow \begin{Bmatrix} A \rightarrow d\cdot, a/c \\ B \rightarrow d\cdot, c/a \end{Bmatrix} I_6$

goto $(I_2, a) \rightarrow \{S \rightarrow Aa\cdot, \$\} - I_7$

goto $(I_3, c) \rightarrow \{S \rightarrow Bc\cdot, \$\} I_8$

goto $(I_4, A) \rightarrow \{S \rightarrow aA\cdot C, \$\} I_9$

goto $(I_4, d) \rightarrow \{A \rightarrow d\cdot, a/c\} I_{10}$

goto $(I_5, B) \rightarrow \{S \rightarrow bB\cdot a, \$\} I_{11}$

goto $(I_5, d) \rightarrow \{B \rightarrow d\cdot, c/a\} I_{12}$

goto $(I_9, c) \rightarrow \{S \rightarrow aAc\cdot, \$\} I_{13}$

goto $(I_{11}, a) \rightarrow \{S \rightarrow bBa\cdot, \$\} I_{14}$

	a	b	c	d	$	S	A
I0	I_4	I_5	~~I_6~~	I_6		1	2
I1					Acc (accept)		
I2	~~S_7~~						
I3			S_8				
I4				S_{10}			
I5				S_{12}			11
I6	R_5		R_6				
I7				~~R_9~~	R_1		
I8				~~R_3~~	R_3		
I9	S_{13}						
I10	R_6						
I11			S_{14}				
I12			R_5				
I13					R_4		
I14					R_2		

Find whether the following grammar is LL(1) or not

$S \rightarrow AB / eDa$

$A \rightarrow ab / c$

$B \rightarrow dC$

$C \rightarrow eC / \epsilon$

$D \rightarrow FD / \epsilon$

FIRST(A) $\rightarrow \{a, c\}$

FIRST(B) $\rightarrow \{d\}$

FIRST(C) $\rightarrow \{e, \epsilon\}$

FIRST(D) $\rightarrow \{f, \epsilon\}$

FIRST(S) $\rightarrow$ FIRST(AB) $\cup$ FIRST(eDa)

$\rightarrow \{a, c\} \cup \{e\}$

$\rightarrow \{a, c, e\}$

$\because$ for LL(1)

$\because$ FIRST(α) $\cap$ FIRST(β) $\rightarrow \{\phi\}$

$\alpha = AB$, $\beta = eDa$

FIRST(AB) $\cap$ FIRST(eDa)

$\rightarrow \{a, c\} \cap \{e\}$

$\rightarrow \{\phi\}$

$\therefore$ FIRST(AB) $\cap$ FIRST(eDa) $\rightarrow \{\phi\}$

∴ It is LL(1) grammer

FOLLOW(S) → {$}

FOLLOW(A) → FIRST(B)
→ {d}

FOLLOW(B) → FOLLOW(S)
→ {$}

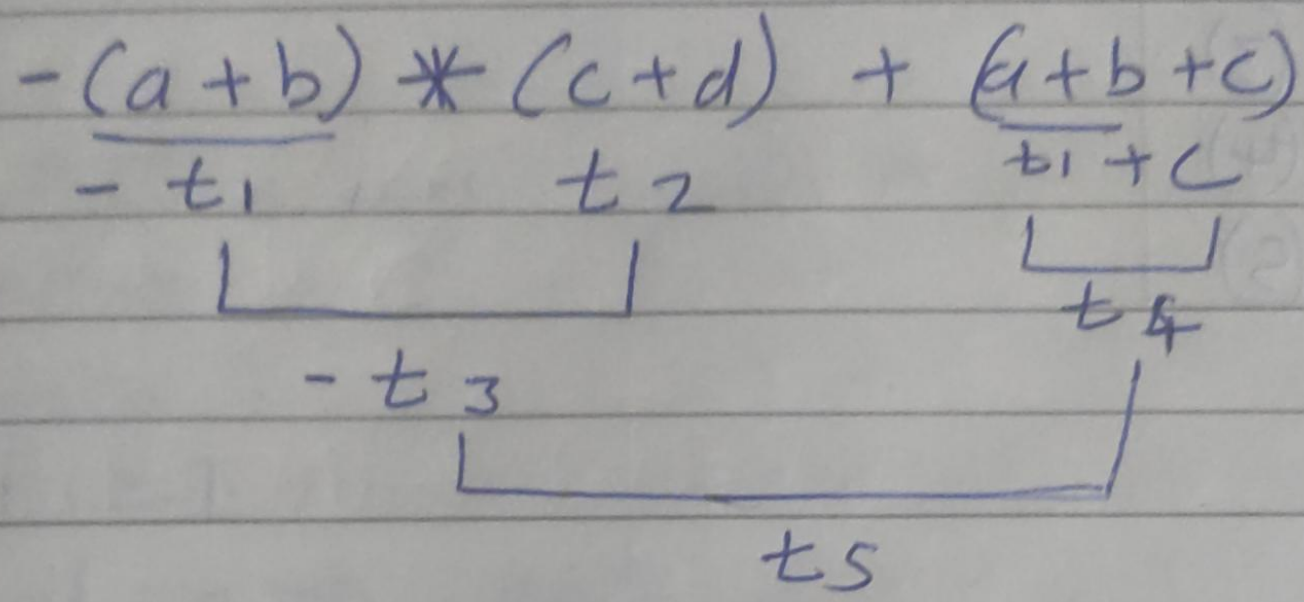
FOLLOW(C) → FOLLOW(B)
→ {$}

FOLLOW(D) → FIRST(a)
→ {a}

	a	b	c	d	e	f	$
S	S→AB		S→AB		S→eDa		
A	A→ab		A→C				
B				B→dC			
C					C→eC		C→ϵ
D	D→ϵ					D→fD	

Show Quadruple, Triple and Indirect triples for the following expression.

$-(a+b) * (c+d) + (a+b+c)$



Quadruple

	Operator	oprand 1	oprand 2	result
(1)	+	a	b	t_1
(2)	+	c	d	t_2
(3)	*	t_1	t_2	t_3
(4)	+	t_1	c	t_4
(5)	-	t_4	t_3	t_5

iii) Indirect Triple

1	(1)
2	(2)
3	(3)
4	(4)
5	(5)

Assignment 2

Translate following statement into Three Address Code

If $x < y$ and $a > b$

If $x < y$ goto 4

$t_1 = 0$

goto 5

$t_1 = 1$

if $a > b$ goto 8

$t_2 = 0$

goto 9

$t_2 = 1$

$t_3 = t_1$ and t_2

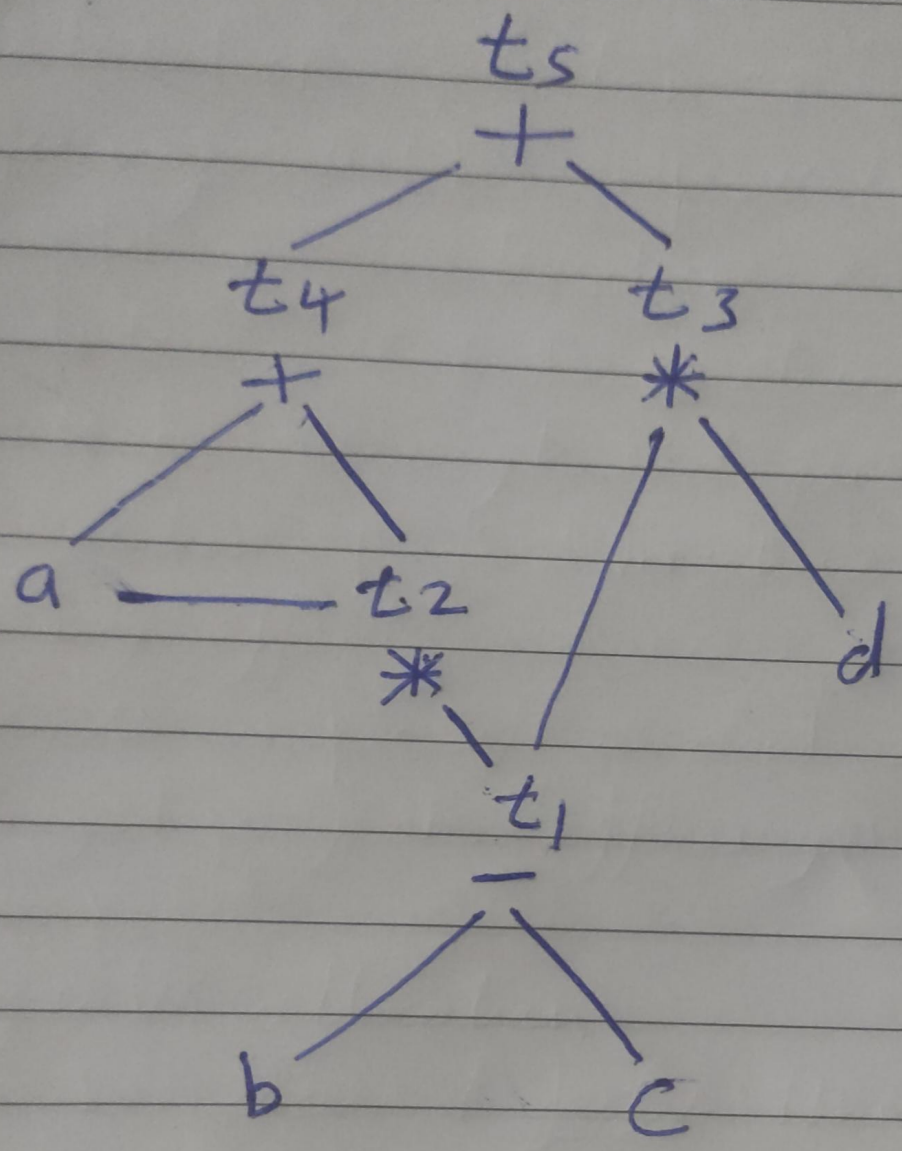
Q.5 Construct the given expression into DAG
a+a*(b-c)+(b-c)*d

Ans Steps for constructing a DAG →

① d_1 = leaf (id, entry - a)
② d_2 = leaf (id, entry - a) = d_1
③ d_3 = leaf (id, entry - b)
④ d_4 = leaf (id, entry, - c)
⑤ d_5 = node ('-', d_3, d_4)
⑥ d_6 = node ('*', d_1, d_5)
⑦ d_7 = node ('+', d_1, d_6)
⑧ d_8 = leaf (id, entry - b) = d_3
⑨ d_9 = leaf (id, entry - c) d_4
⑩ d_{10} = node ('-', d_3, d_4) = d_5
⑪ d_{11} = leaf (id, entry - d)
⑫ d_{12} = node ('*', d_5, d_{11})
⑬ d_{13} = node ('+', d_7, d_{12})

Now,

~~$t_1 = b =$~~
$t_1 = b - c$
$t_2 = a * t_1$
$t_3 = t_1 * d$
$t_4 = a + t_2$
$t_5 = t_4 + t_3$



DAG for $a + a * (b - c) + (b - c) * d$