Assignment-2

1) Consider the grammar

S → aAd | bBd | aBe | bAe

A → c

B → c

i) Create the LR(1) canonical sets of items

ii) Construct the CLR(1) parsing table for the grammar.

iii) Construct the LALR (1) parsing table for the grammar. Is the grammar LALR(1)? Justify.

Ans → Augmented Grammar →

S' → S

S → aAd

S → bBd

S → aBe

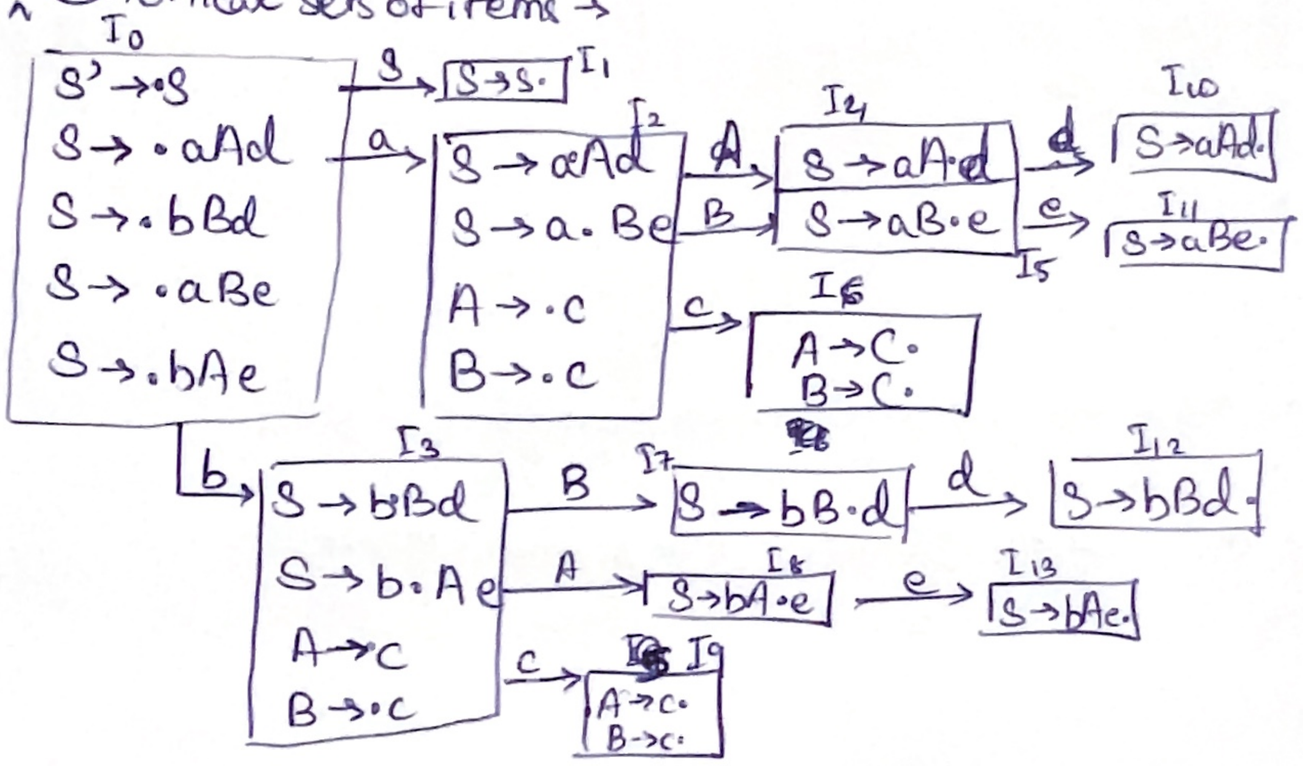
S → bAe

A → c

B → c

LR(1)

i) Canonical sets of items →



ii) CLR(1) Parsing Table →

	Action						GoTo		
	a	b	c	d	e	$	S	A	B
I_0	s2	s3					1		
I_1						accept			
I_2			s6					4	5
I_3			s9					8	7
I_4			s10						
I_5					s11				
I_6			r6	r7					
I_7			s12						
I_8					s13				
I_9			r7	r6					
I_{10}						r2			
I_{11}						r4			
I_{12}						r3			
I_{13}						r5			

Name: _____ Regd No: _____

iii) LALR(1) Parsing table →

	Action						GoTo		
	a	b	c	d	e	$	S	A	B
I_0	s2	s3					1		
I_1						accept			
I_2			s6					4	5
I_3			s6					8	7
I_4				s9					
I_5					s10				
I_6				r6/r7		r6/r7			
I_7				s11					
I_8					s12				
I_9						r2			
I_{10}						r4			
I_{11}						r3			
I_{12}						r5			

So, the grammar is not LALR(1) as there are reduce-reduce conflict in state 6.

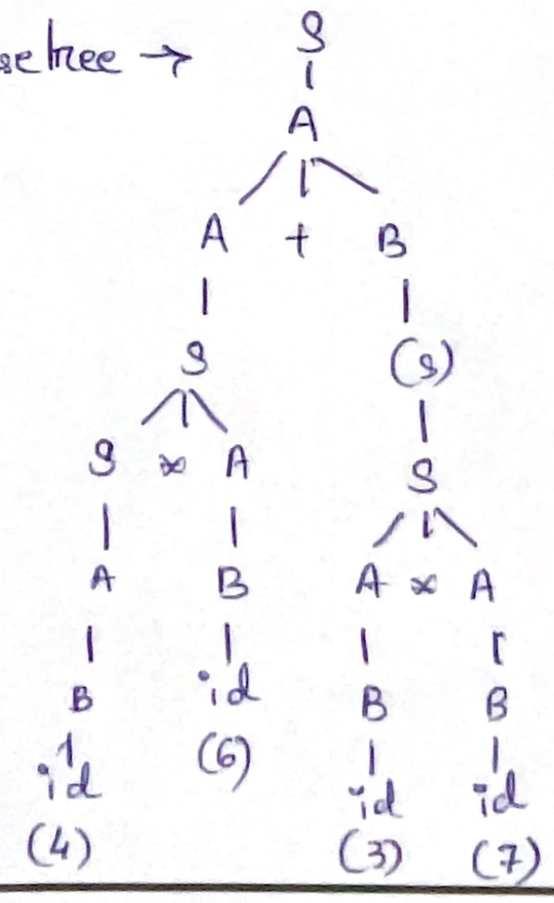
2) Consider the following grammar and their syntax Directed Translation (SDT) rules.

Productions	Semantic Rules
S → S*A	{S.val = S.val xA.val}
S → A	{S.val = A.val}
A → A+B	{A.val = A.val - B.val}
B → (S)	{B.val = 2}
A → B	{A.val = B.val}
B → id	{B.val = id.lexval}

Construct the annotated parse trees and evaluate the value of the expression 4*6+3*7.

Ans → S → A
→ A+B
→ (S)+B
→ (S*A)+B
→ (A*A)+B
→ (B*A)+B
→ (id*A)+B
→ (id*B)+B
→ (id*id)+B
→ (id*id)+(S)

Parse tree →



Name: _____ Regd No: _____

→ (id * id) + (S * A)

→ (id * id) + (A * A)

→ (id * id) + (B * A)

→ (id * id) + (id * A)

→ (id * id) + (id * B)

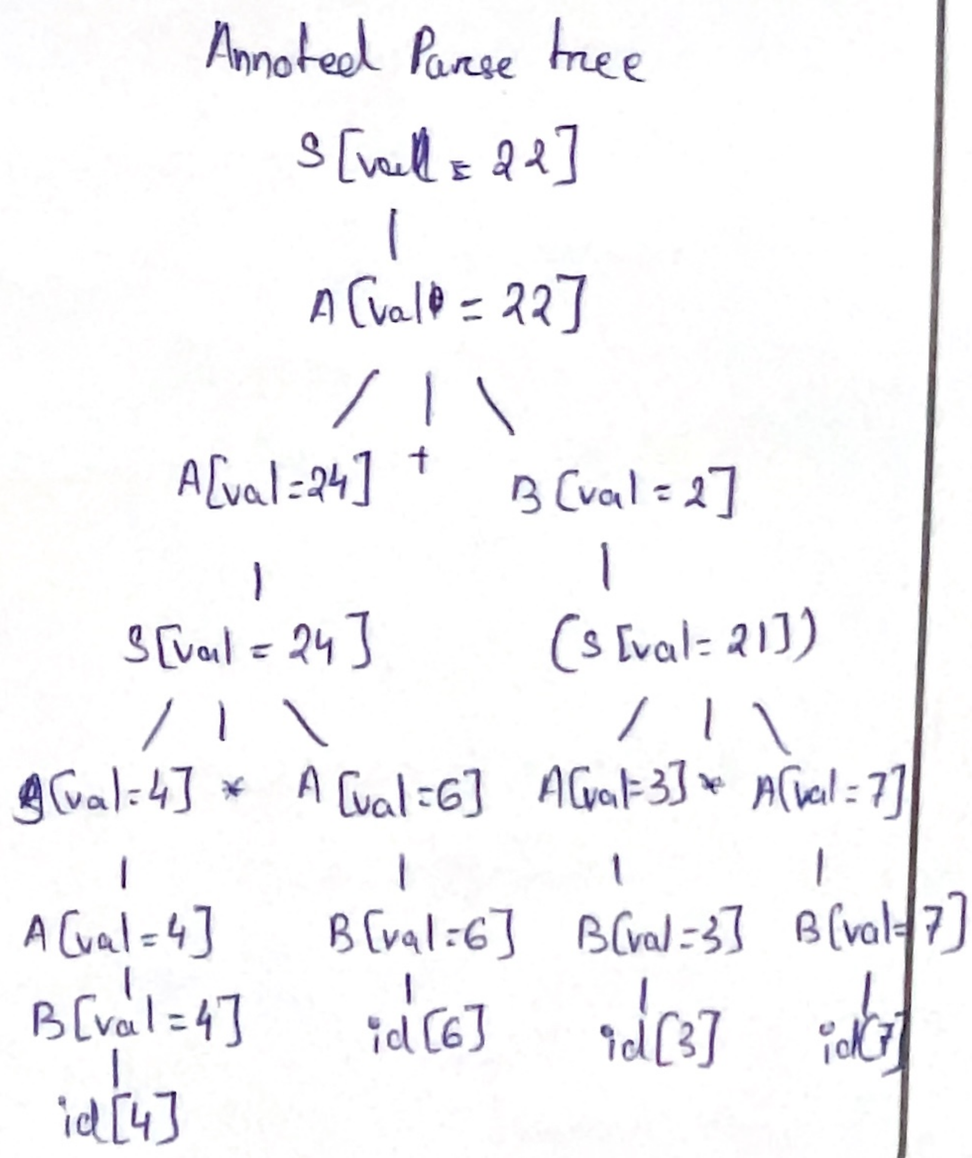
→ (id * id) + (id * id)

Parse tree with annotations →

- Leaves (terminals id) →
  - id(4): B → id, B.val = id.lexval = 4
  - id(6): B → id, B.val = id.lexval = 6
  - id(3): B → id, B.val = id.lexval = 3
  - id(7): B → id, B.val = id.lexval = 7
- Right subtree: B → (S) for 3*7:
  - Subtree for S (inside (S)):
    - Left S → A → B, B → id(3): B.val = 3
    - Right A → B → id(7): B.val = 7
    - S → S * A: S.val = S.val × A.val = 3 × 7 = 21
  - B → (S): B.val = 2 (fixed value for parenthesised expression)
- Left subtree: S → S * A for 4*6:
  - Left S → A → B → id(4): B.val = 4
  - Right A → B → id(6): B.val = 6
  - S.val = S.val × A.val = 4 × 6 = 24
- Top A → A + B:
  - Left A (from S of 4*6): A.val = 24
  - Right B (from (3*7)): B.val = 2.
  - A.val = A.val - B.val = 24 - 2 = 22
- Root S → A
  - S.val = A.val = 22.

∴ Value of expression as per SDT is 22.

Annotated Parse tree



Name: _____ Regd No: _____

3) A shift reduce parser carries out the actions specified within braces immediately after reducing with the corresponding rule of grammar.

S → aaW {print '1'}
S → b {print '2'}
W → Sc {print '3'}

Construct the annotated parse tree and find the translation of aaaabcc using the syntax directed translation scheme described by the above rules.

Ans → S → aaW
→ aaSc
→ aaaaWc
→ aaaaScc
→ aaaabcc

Annotated Parse tree →

Simulate shift reduce parsing →

- Input: aaaabcc
- Stack: Initially empty
- Actions:
  1. shift a: Stack = a
  2. shift a: Stack = aa
  3. shift a: Stack = aaa
  4. shift a: Stack = aaaa
  5. shift b: Stack = aaaab
  6. reduce S → b: Pop b, push S, print 2
     - Stack = aaaaS
  7. Shift c: Stack = a a a a Sc
  8. Reduce W → Sc: Pop Sc, push W, print 3
     - Stack = aaaaW
  9. Reduce S → aaW: Pop aaW, push S, print 1
     - Stack = aaW
  10. Shift c: Stack = aaSc
  11. Reduce W → Sc: Pop Sc, push W, print 3.
      - Stack → aaW
  12. Reduce S → aaW: Pop aaW, push S, print 1
      - Stack → S
  13. Accept.

∴ Output sequence → 2 3 1 3 1

Name: _____ Regd No: _____

4) Consider the expression $a+a*(b-c)+(b-c)*d$. Generate the three-address code TAC for the given expression.

Ans → Expression → $a+a*(b-c)+(b-c)*d$

Three-Address Code (TAC) → Compute Lets $t_1 = b-c$

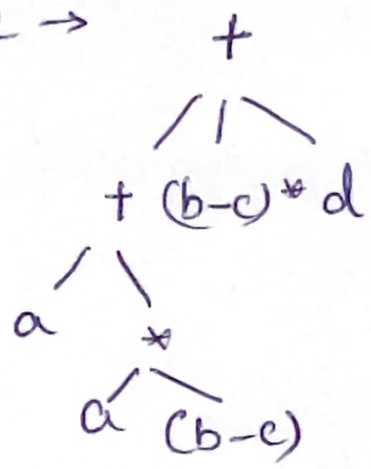
$t_2 = a*(b-c) = a*t1$

$t_3 = a+a*(b-c) = a+t_3$

$t_4 = \sout{a+a*(b-c)+}(b-c)*d = t1*d$

$t5 = t3 + t4$

Parse tree →



5) Represent the three-address code that is generated in Q4, using the following intermediate code formats:

i) Quadruples ii) Triples iii) Indirect Triples

Ans → i) Quadruples →

TAC to Quadruples →

- $t1 = b-c$ → (sub, b, c, t1)
- $t2 = a*t1$ → (mul, a, t1, t2)
- $t3 = a+t2$ → (add, a, t2, t3)
- $t4 = t1*d$ → (mul, t1, d, t4)
- $t5 = t3*t4$ → (add, t3, t4, t5)

Table →

Index	Op	Arg1	Arg 2	Result
0	sub	b	c	t1
1	mul	a	t1	t2
2	add	a	t2	t3
3	mul	t1	d	t4
4	add	t3	t4	t5

ii) Triples →

- $t1 = b-c$ → (sub, b, c) at index 0 (result is index 0)
- $t2 = a*t1$ → (mul, a, 0) at index 1 (uses t1 as index 0)
- $t3 = a+t2$ → (add, a, 1) at index 2 (uses t2 as index 1)
- $t4 = t1*d$ → (mul, 0, d) at index 3 (uses t1 as index 0)
- $t5 = t3+t4$ → (add, 2, 3) at index 4 (uses t3 as index 2, t4 as index 3)

Name: _____ Regd No: _____

Triples Table →

Index	Op	Arg 1	Arg 2
0	sub	b	c
1	mul	a	0
2	add	a	1
3	mul	0	d
4	add	2	3

iii) Indirect Triples → Indirect triples ~~these~~ store them same triples as above but use a separate instruction pointer list to reference them.

Instruction pointer list → Since no reordering is specified, the pointer list is sequential →

Index	Instruction
0	0
1	1
2	2
3	3
4	4

Name: _____ Regd No: _____