Assignment-1

Q1) Show the different inputs & outputs of the various phases of a compiler for the assignment statement C = (F-32)* (5/9)

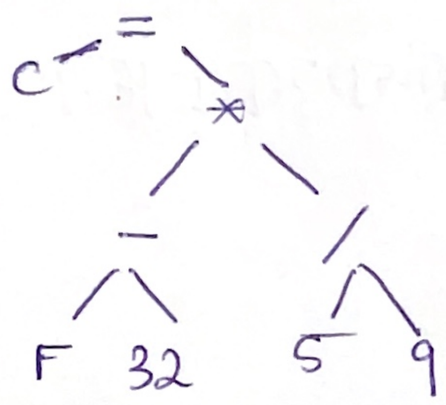
Ans → a) Lexical Analysis → Input → Source code as a sequence of characters.

Output → A stream of tokens:

Identifier (C), assign_op(=), left_paren, identifier(F), minus_op(-), number(32), right_paren, mult_op(*), left_paren, number(5), div_op(/), number(9), right_paren

b) Syntax Analysis → Input → Token stream from lexical analysis.

Output → Parse Tree / Abstract Syntax Tree (AST) representing the expression structure



c) Semantic Analysis → Input → AST from the syntax analysis phase.

Output → Annoted AST with type information and error checks:

i) Check type compatibility (e.g. numeric types for arithmetic)

ii) Validate variable declarations

d) Intermediate Code generation → Input → Annoted AST.

Output → Intermediate representation, often in three-address code (TAC) →

$t_1 = F - 32$

$t_2 = 5/9$

$t_3 = t_1 * t_2$

$C = t_3$

e) Code Optimisation → Input → Intermediate code

Output → Optimised intermediate code, removing redundancies:

$t1 = F - 32$

$C = t1 * 0.5556$

Name: _____ Regd No: _____

e) Code generation → Input → Optimised intermediate code

Output → Assembly or machine code:

```
Load F
Sub 32
Mul 0.5556
Store C
```

g) Linking and loading → Input → Object code.

Output → Executable machine code ready for execution.

2) Provide regular expressions to characterise the following the lexical items:

a) Identifiers → Regular expression → [a-zA-Z]([a-zA-Z0-9_]*[a-zA-Z0-9])?

b) Numbers → Regular expression → (0|[1-9][0-9]*)(\.[0-9]*[1-9])?([eE][+-]?[1-9][0-9]*)?

c) White space → [\t\n\r]+

d) Arithmetic Operators → (\+|\-|*|\/|div|mod)

e) Logical Constants → (true|false)

f) Logical Operators → ~~(<=|>=|<|>|=)~~ (not|and|or)

g) Comparison Operators → (<=|>=|<|>|=)

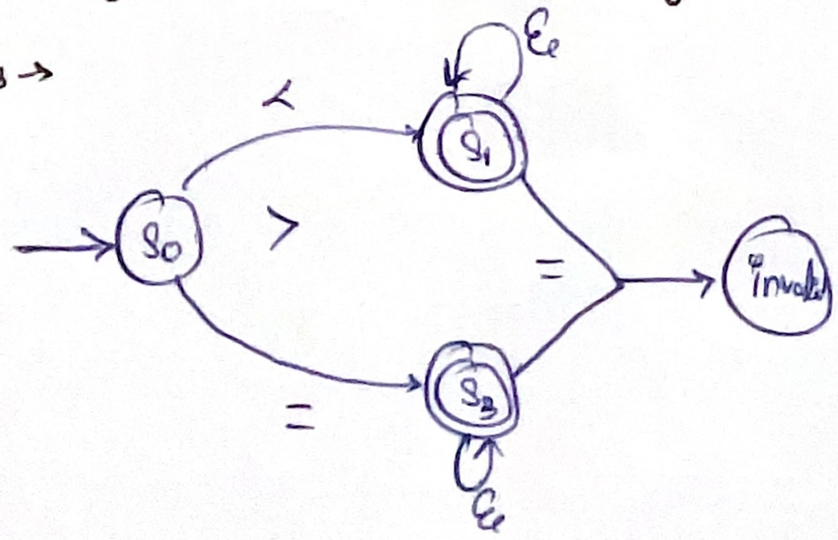
h) Parantheses → (\(|\))

i) Keywords → (if|then|else)

j) Literal Strings → "([^"\\]*(\\.[^"\\]*)*)"

(Q3) A grammar for branching statements is defined as

Ans →

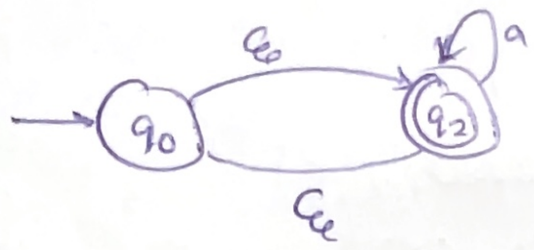


Name: _____ Regd No: _____

4) Design the DFA for the given Regular Expression $a(a|b)^*a$

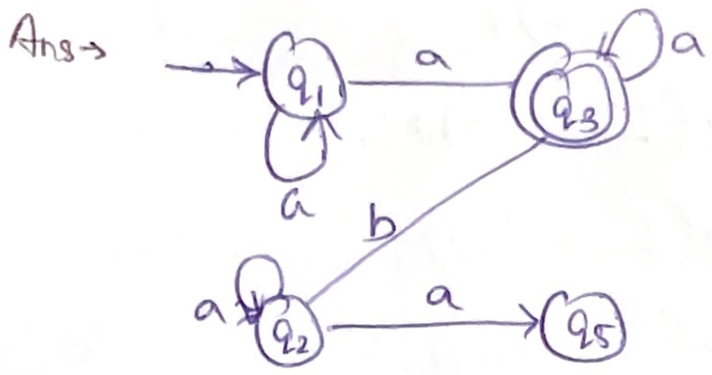
a) Convert the RE to ε-NFA.

Ans→

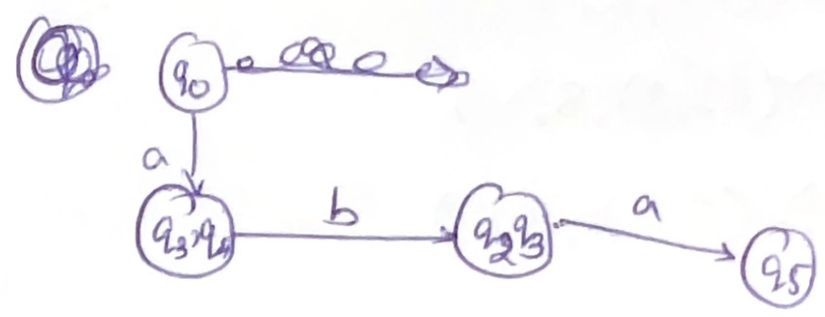


b) Convert the ε-NFA to NFA and NFA to Minimized DFA.

Ans→



Minimised DFA



5) Divide the following C program into appropriate lexemes and count the number of tokens.

```
float limitedSquare(x)
{
 float x;
 /* returns x-squared, but never more than 100 */
 return (x<=-10.0 || x>=10.00)? 100: x*x;
}
```

Ans→ Lexeme Analysis.

i) float → Keyword ii) limitedSquare → Identifier iii) (→ Left parenthis iv) x → Identifier v)) → Right parenthesis vi) { → Left brace vii) float → keyword viii) x → Identifier ix); → Punctuation (Semicolon) x) /* returns x-squared, but never more than 100 */ → Comment (Ignored by lexical analysis) xi) return → keyword xii) (→ Left parenthesis xiii) x → Identifier xiv) <= → Relational Operator xv) - → Arithmetic xvi) 10.0 → Number xvii) || → Logical xviii) x>= → ~~Relational~~ Identifier xix) 10.0 → ~~Number~~ Relational xx) 10.00 → Number xxi) ? → Conditional xxii) 100 → Number xxiii) : → Conditional xxiv) x → Identifier xxv) * → Arithmetic xxvi) x → Identifier xxvii) ; → Punctuation xxviii) } → Right Brace xxix)) → Right paren

~~) → Right Paren~~

Number of tokens → 29

Name: _____ Regd No: _____

6) Consider the context free grammar : $S \to (L) \mid a$ $L \to L,S \mid S$ and the string ((a,a),a,(a))

a) Give the leftmost derivation for the string.

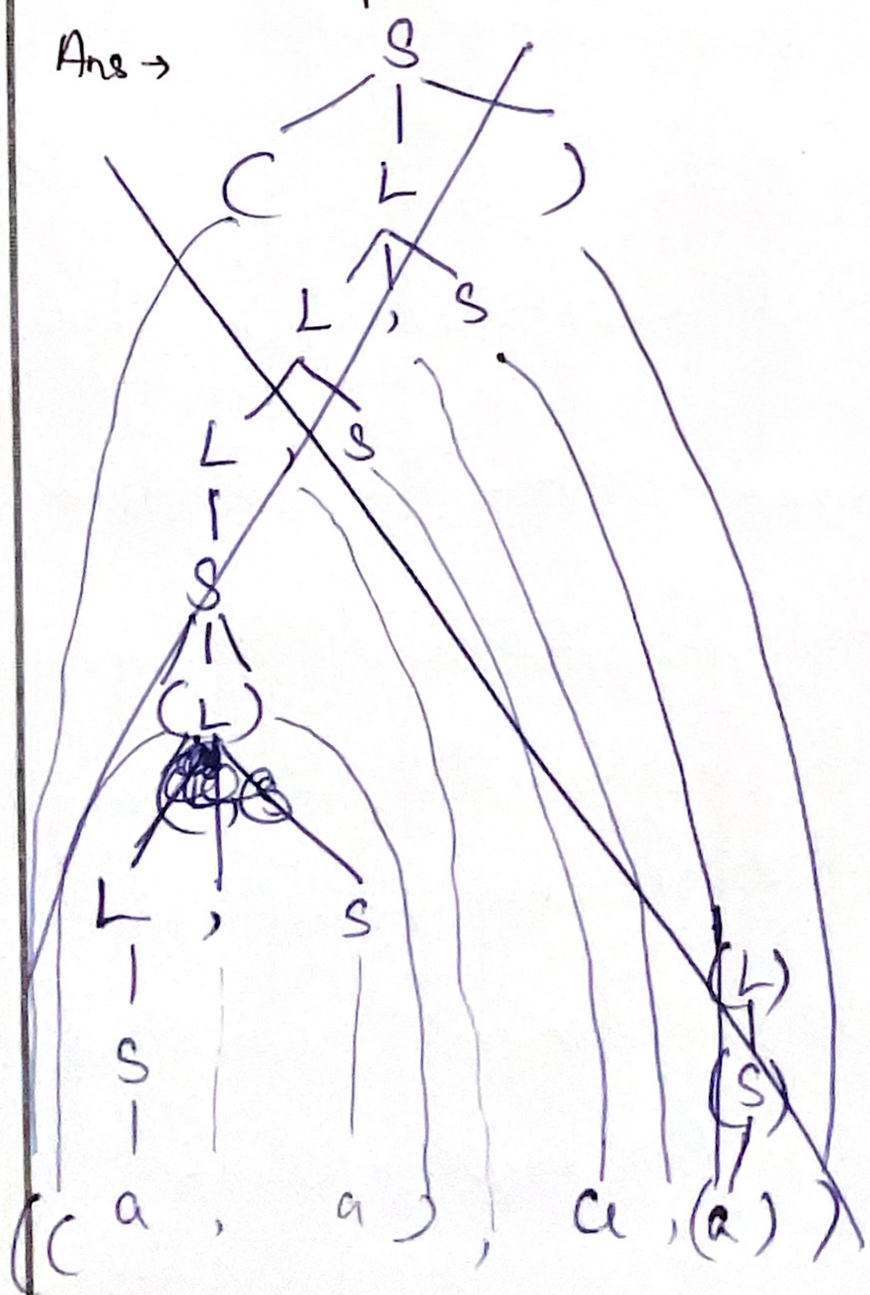
Ans →
$S \to (L)$
$\to (L,S)$
$\to (L,S,S)$
$\to (S,S,S)$
$\to ((L),S,S)$
$\to ((L,S),S,S)$
$\to ((S,S),S,S)$
$\to ((a,S),S,S)$
$\to ((a,a),S,S)$
$\to ((a,a),a,S)$
$\to ((a,a),a,(L))$
$\to ((a,a),a,(S))$
$\to ((a,a),a,(a))$

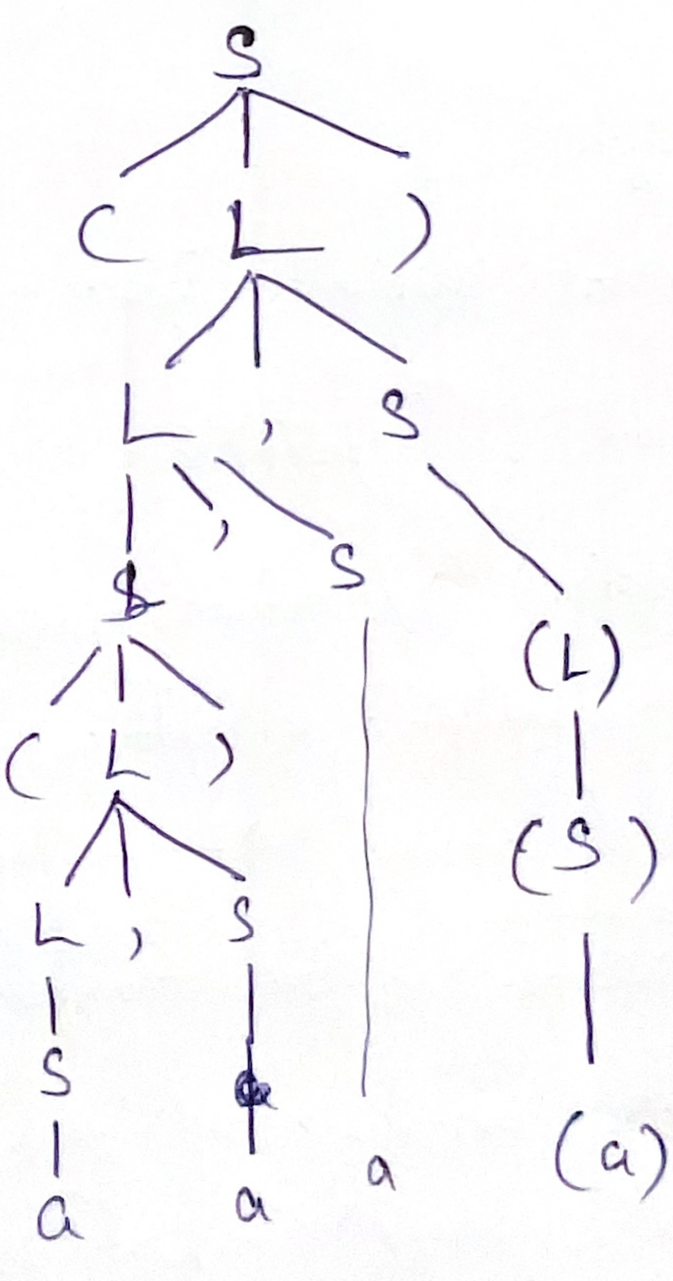
b) Give the rightmost derivation for the string.

Ans →
$S \to (L)$
$\to (L,S)$
$\to (L,(L))$
$\to (L,(a))$
$\to (L,S,(a))$
$\to (L,a,(a))$
$\to (S,a,(a))$
$\to ((L),a,(a))$
$\to ((L,S),a,(a))$
$\to ((L,a),a,(a))$
$\to ((S,a),a,(a))$
$\to ((a,a),a,(a))$

c) Give a parse tree for the string.

Ans →





Name: _____ Regd No: _____

d) Is the grammar ambiguous or unambiguous? Justify your answer.

Ans→ A grammar is ambiguous if a single string can have multiple distinct parse trees or different derivations. This grammar is ambiguous because the string can be derived in multiple ways, leading to different parse trees.

e) Eliminate the left recursion from the grammar (if any).

Ans→ $S \rightarrow (L) \mid a$

$L \rightarrow L, S \mid S$

Rule $L \rightarrow L, S$ is left-recursive because L appears on the left side immediately after L.

Rewrite rule for L

$L \rightarrow SL'$

$L' \rightarrow , SL' \mid \epsilon$

New Grammar

$S \rightarrow (L) \mid a$

$L \rightarrow SL'$

$L' \rightarrow , SL' \mid \epsilon$

7) Let G be the following context-free grammar, where E is the start symbol of G.

Ans→ $E \rightarrow E > E \mid E \geq E \mid E < E \mid E \leq E \mid E << E \mid E >> E \mid E != E \mid id$

Construction of unambiguous grammar.

$E \rightarrow E1$

$E1 \rightarrow E1 != E2 \mid E2$

$E2 \rightarrow E3 << E2 \mid E3 >> E2 \mid E3$

$E3 \rightarrow E3 > E4 \mid E3 \geq E4 \mid E3 < E4 \mid E3 \leq E4 \mid E4$

$E4 \rightarrow id$

Example derivation→ Lets derive: id << id > id != id

i) $E \rightarrow E1 \rightarrow E2 \rightarrow E3 << E2 \rightarrow E4 << E2 \rightarrow id << E2$

ii) $E2 \rightarrow E3 > E4 \rightarrow E4 > E4 \rightarrow id > id$

iii) id << id > id

iv) $E2 \rightarrow E1 != E2 \rightarrow E4 != E2 \rightarrow id != E2$

v) $E2 \rightarrow E3 \rightarrow E4 \rightarrow id$

vi) id << id > id != id

8) Let G be the following Context-free Grammar.

$P \to xQRS$
$Q \to yz / z$
$R \to w | \epsilon$
$S \to y$

Compute the FIRST and FOLLOW function for all the non-~~technicals~~ terminals present in the above grammar.

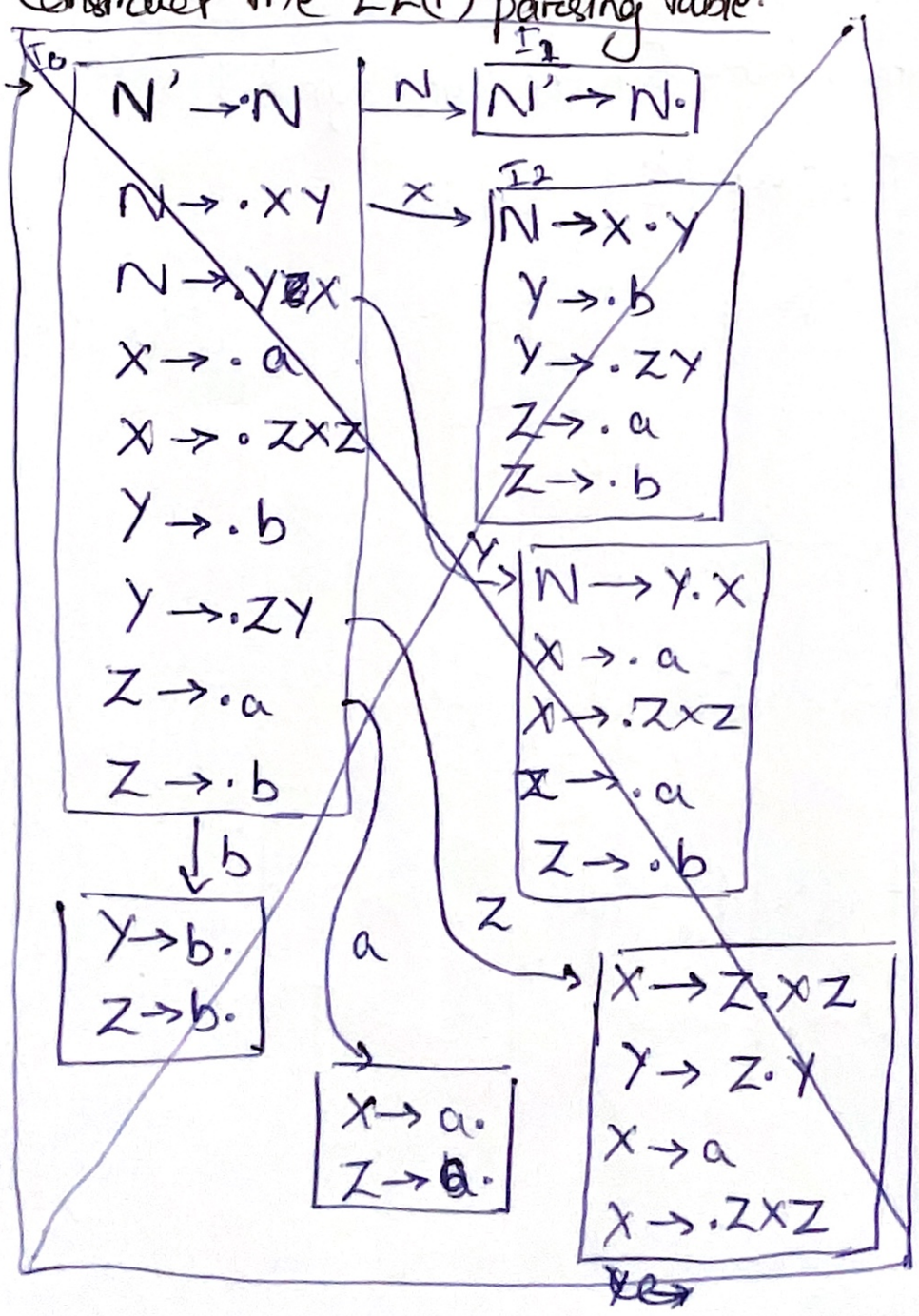
Ans →

	First	Follow
P	{x}	{$}
Q	{y, z}	{w, y}
R	{w, ε}	{y}
S	{y}	{$}

9) Consider the following grammar. $N \to XY | YX$
$X \to a | ZXZ$
$Y \to b | ZY$
$Z \to a | b$

a) Construct the LL(1) parsing table.

Ans →



	First	Follow
N	{a, b}	$
X	{a, b}	{b, a}
Y	{b, a}	{$, a, b}
Z	{a, b}	{a, b}

LL(1) Table →

	a	b
N	N → XY | YX	N → XY | YX
X	X → a | ZXZ	ZXZ
Y	ZY	b | ZY
Z	a	b

b) Is this grammar LL(1)?

Ans → No. Conflicts exists in the parsing table.
Multiple entries for same non-terminal on terminal pairs

Name: _____ Regd No: _____

c) Verify the given grammar is ambiguous or not.

Ans → Lets derive a string ababa.

N → XY
→ ZXZY
→ abZY

N → YX
→ ZYX
→ ZZYX
→ ZZZYX
→ ababa

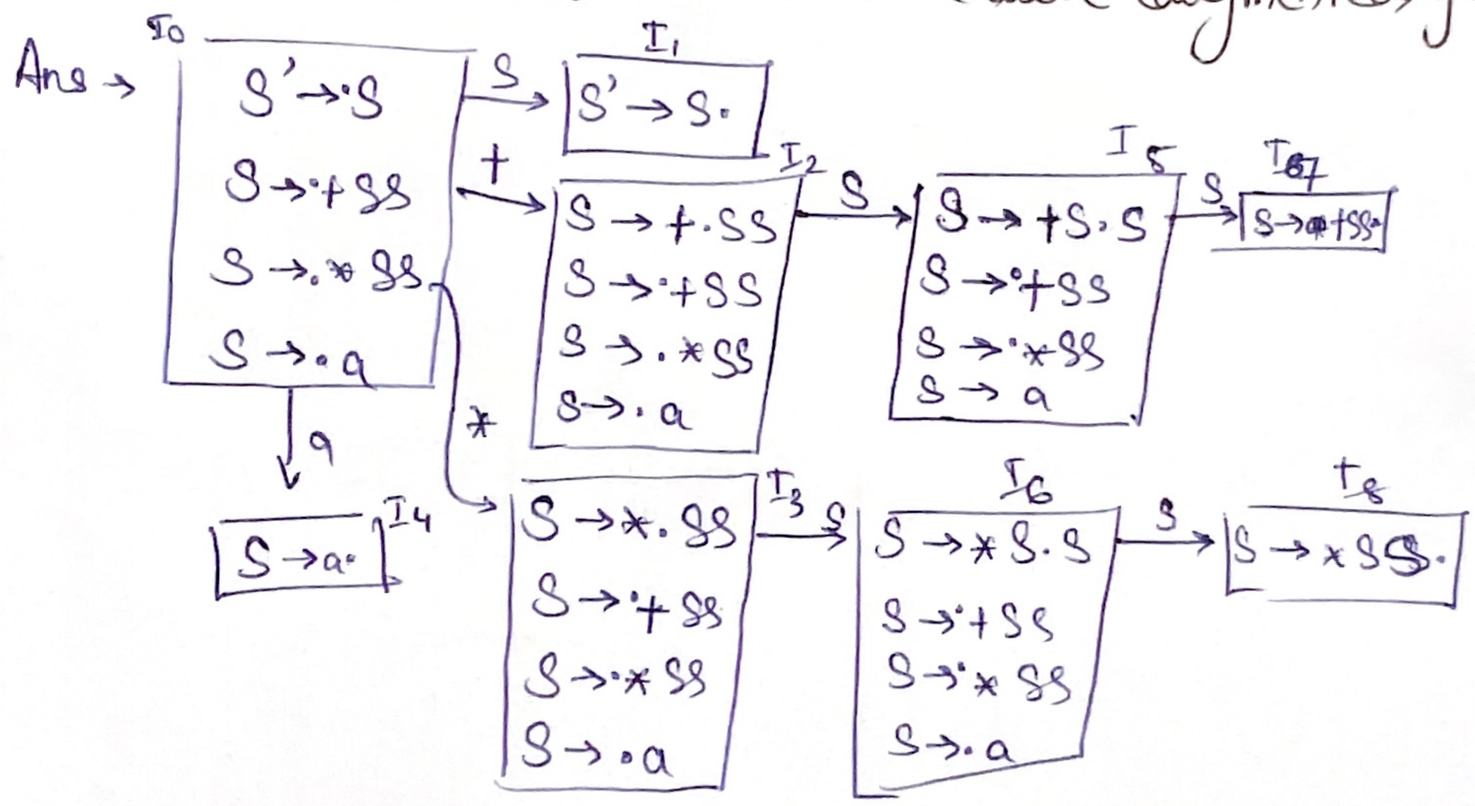
N → YZX
→ ZYX
→ ZYZXZ
→ abaZb

The grammar is not ambiguous as it didn't have same string generated from different parse trees.

10) Consider the following grammar: S → +SS | *SS | a and the string +a*aa.

a) Construct the SLR sets of items of the above (augmented) grammar.

Ans →



b) Compute the GOTO function for these ~~states~~ sets of items. Show the parsing table for this grammar.

Is this grammar ~~SLR~~ SLR?

Ans →

	GOTO				Action	
	+	*	*	a	$	S
I_0	I_2		I_3	I_4		
I_1					accept	
I_2	I_2		I_3	I_4		5
I_3	I_2		I_3	I_4		6
I_4	r_4		r_4	r_4	r_4	
I_5	I_2		I_3	I_4		7
I_6	I_2		I_3	I_4		8
I_7	r_7		r_7	r_7	r_7	
I_8	r_8		r_8	r_8	r_8	

Yes the grammar is ~~ambiguous~~ SLR(1) as it has no conflicts in parsing table.

Name: _____ Regd No: _____