Simple LR(SLR) Parser:-

$S \rightarrow AA$

$A \rightarrow aA \mid b$

String : aabb

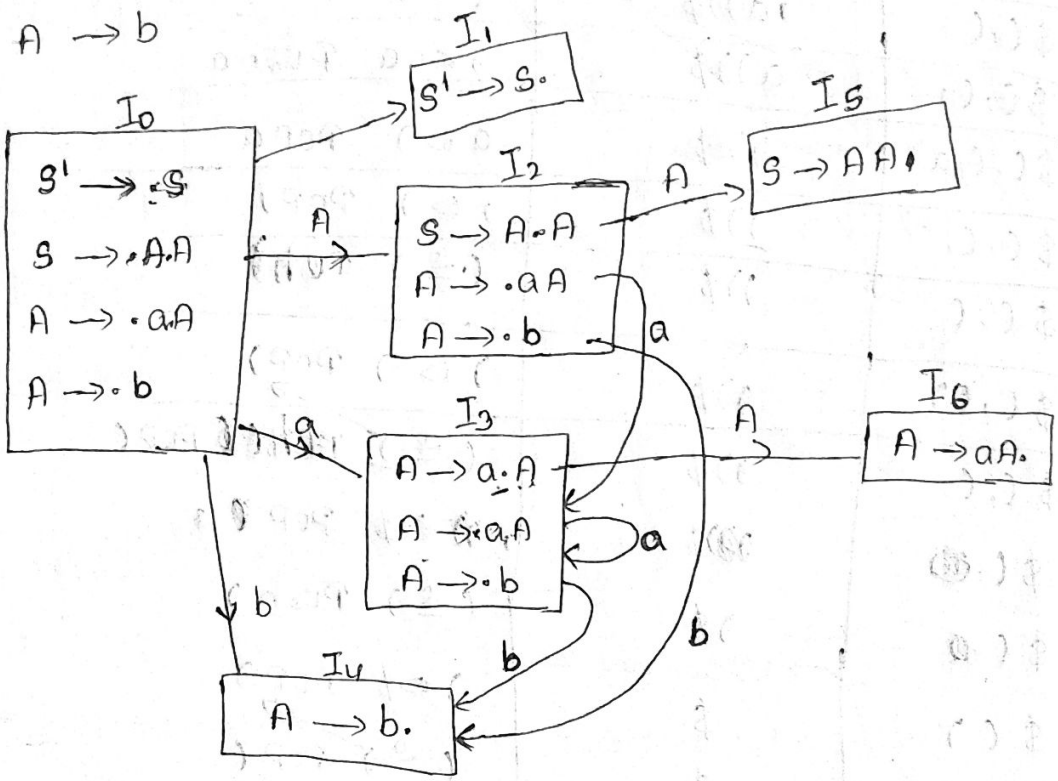
Augmented Grammar:-

$S' \rightarrow S$

$S \rightarrow AA$

$A \rightarrow aA$

$A \rightarrow b$



Parsing Table

	Action			Goto	
	a	b	$	A	S
0	S_3	S_4		2	1
1			Accept		
2	S_3	S_4		5	
3	S_3	S_4		6	
4	γ_3	γ_3	γ_3		

	Action			goto	
	a	b	$	A	S
5.	r_1	r_1	r_1		
6.	r_2	r_2	r_2		

Stack	Input	Action	
0	aabb$	Shift	
0a3	abb$	Shift	
0a3a3	bb$	Shift	
0a3a3b4	b$	Reduce A → b	[POP 1x2 elements]
0a3a3A6	b$	Reduce A → aA	[POP 2x2 elements]
0a3A6	b$	Reduce A → aA	[POP 2x2 elements]
0A2	b$	Shift	
0A2b4	$	Reduce A → b	[POP 1x2 elements]
0A2A5	$	Reduce S → AA	[POP 2x2 elements]
0S1	$	Accept	

② $E \rightarrow E+T \mid T$

$T \rightarrow T*F \mid F$

$F \rightarrow (E) \mid id$

Augmented Grammar:-

$E' \rightarrow E$

$E \rightarrow E+T$ ①

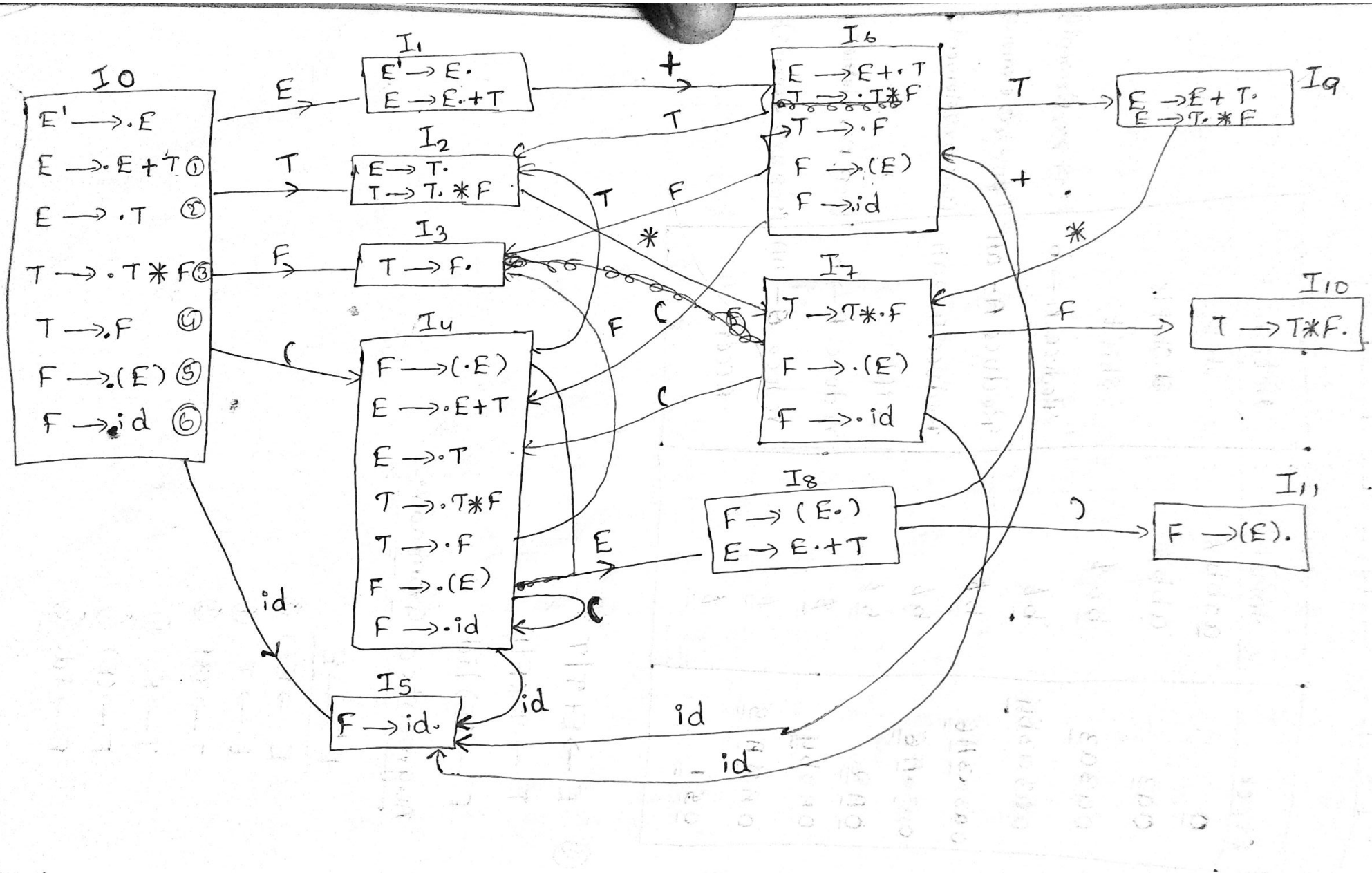
$E \rightarrow T$ ②

$T \rightarrow T*F$ ③

$T \rightarrow F$ ④

$F \rightarrow (E)$ ⑤

$F \rightarrow id.$ ⑥


I0
E' → .E
E → .E+T ①
E → .T ②
T → .T*F ③
T → .F ④
F → .(E) ⑤
F → .id ⑥
I1
E' → E.
E → E.+T
I2
E → T.
T → T.*F
I3
T → F.
I4
F → (.E)
E → .E+T
E → .T
T → .T*F
T → .F
F → .(E)
F → .id
I5
F → id.
I6
E → E+.T
T → .T*F
T → .F
F → .(E)
F → .id
I7
T → T*.F
F → .(E)
F → .id
I8
F → (E.)
E → E.+T
I9
E → E+T.
T → T.*F
I10
T → T*F.
I11
F → (E).
E
T
F
+
*
(
)
id

	Action						Goto		
	+	*	(	)	id	$	E	T	F
0			S_4		S_5		1	2	3
1	S_6					Accept			
2	γ_2	S_7		γ_2		γ_2			
3	γ_4	γ_4		γ_4		γ_4			
4			S_4		S_5		8	2	3
5	γ_6	γ_6		γ_6		γ_6			
6			S_4		S_5			9	3
7			S_4		S_5				10
8	S_6			S_{11}					
9	γ_1	S_7		γ_1		γ_1			
10	γ_3	γ_3		γ_3		γ_3			
11	γ_5	γ_5		γ_5		γ_5			

	First	Follow
E	{+, *, (, id}	{$, +,)}
T	{*, (, id}	{$, +, *,)}
F	{(, id}	{$, +,), *}

Stack	Input	Action
0	id+id$	shift
0id5	+id$	reduce F → id
0F3	+id$	reduce T → F
0T2	+id$	reduce E → T
0E1	+id$	shift
0E1+6	id$	shift
0E1+6id5	$	reduce F → (E)
0E1+6F3	$	reduce T → F

0 E 1 + 6 T 8	$	reduce E → E + T
0 E 1	$	Accept

Construct an SLR Parser for the given grammar

$S \rightarrow CC$

$C \rightarrow CC \mid d$

Parse the string : ccdd

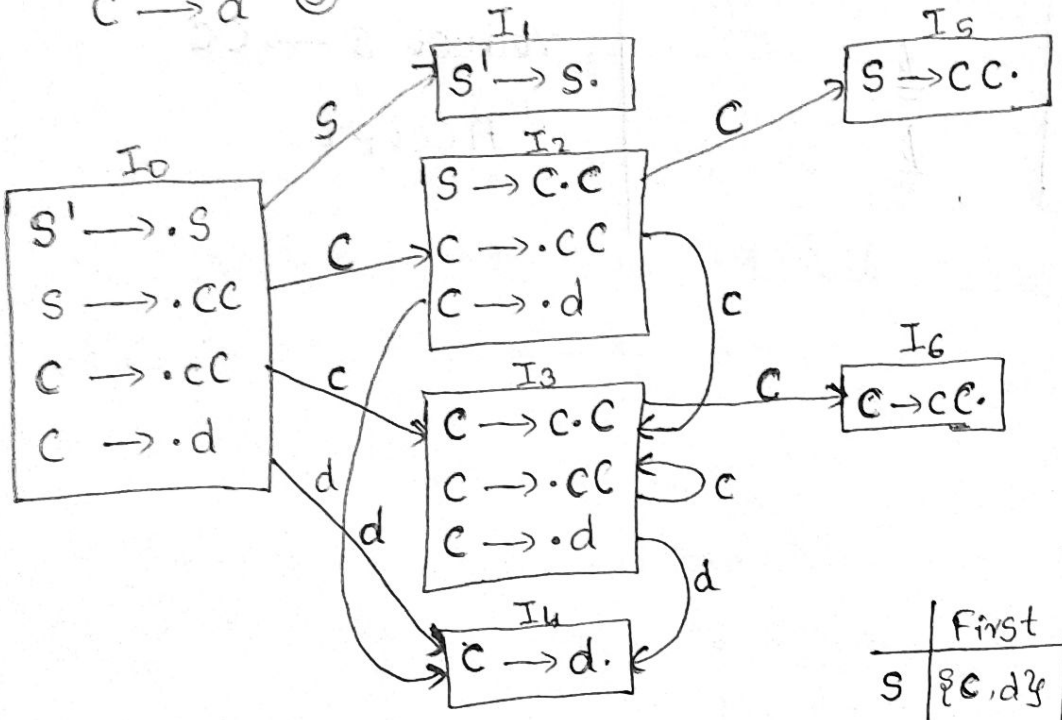
Augmented Grammar:-

$S' \rightarrow S$

$S \rightarrow CC$ ①

$C \rightarrow cC$ ②

$C \rightarrow d$ ③



	First	Follow
S	{c, d}	{$}
C	{c, d}	{c, d, $}

	c	d	$	S	C
0	S_3	S_4		1	2
1			Accept		
2	S_3	S_4			5
3	S_3	S_4			6
4	r_3	r_3	r_3		
5			r_1		
6	r_2	r_2	r_2		

Stack	Input	Action
0	ccdd$	Shift
0c3	cdd$	Shift
0c3c3	dd$	Shift
0c3c3d4	d$	reduce C → d
0c3c3C8	d$	reduce C → cC
0c2C6	d$	reduce C → cC
0C2	d$	Shift
0C2d4	$	reduce C → d
0c2C6	$	reduce S → CC
0S1	$	Accept

Compiler Design

① Operator grammer:-

$E \rightarrow E+T$

$E \rightarrow T$

$T \rightarrow T*F \mid F$

$F \rightarrow (E)$

$F \rightarrow id$

Ans:-

Step-1:-

Rule for Leading:-

① $A \rightarrow \alpha a \beta$ if $\boxed{\alpha \rightarrow \epsilon \mid \text{Single NT}}$

Leading(A) = a

② $A \rightarrow \alpha a \beta$ if $\boxed{\alpha \rightarrow NT}$

Leading(A) = Leading(α)

$E \rightarrow E+T$
$E \rightarrow T$
$T \rightarrow T*F$
$T \rightarrow F$
$F \rightarrow (E)$
$F \rightarrow id$

Leading(E) = {+, Leading(T)} = {+, *, (, id}

Leading(T) = {*, Leading(F)} = {*, (, id}

Leading(F) = {(, id}

Rules for Trailing:-

① $A \rightarrow \alpha a \beta$ if $\boxed{B \rightarrow \epsilon \mid \text{Sigle NT}}$

~~Leading(A) = Leading~~

Trailing(A) = a

② $A \rightarrow \alpha a \beta$ if $\boxed{\beta \rightarrow NT}$

Trailing(A) = Trailing(β)

Trailing(E) = {+, Trailing(T)} = {+, *,), id}

Trailing(T) = {*, Trailing(F)} = {*,), id}

Trailing(F) = {), id}

Step-2 :-

Operator grammer Precendence table :-

*it have two rules

Rule-1 :-

$\$ \lessdot \cdot$ Leading(S) & Trailing(S) $\gtrdot \$$ ∴ S → NT

Rule-2 :-

(a) if x_i & $x_{i+1} \in T$, $\longrightarrow$ $x_i \doteq x_{i+1}$

(b) if x_i & $x_{i+2} \in T$ $\longrightarrow$ $x_i \doteq x_{i+2}$

(c) if $x_i \in T$ & $x_{i+1} \in NT$, $x_i \lessdot \cdot$ Leading(x_{i+1})

(d) if $x_i \in NT$ & $x_{i+1} \in T$, Trailing(x_i) $\gtrdot x_{i+1}$

Now, x_1 x_{i+1} x_{i+2}.

$E \longrightarrow E + T$

Here c, & d Condition will Satisfied

① Trailing(E) $\gtrdot$ +

{+, *,), id} $\gtrdot$ +

② + $\lessdot \cdot$ Leading(T)

+ $\lessdot \cdot$ {*, (, id}

$T \longrightarrow T * F$

Here c & d condition will satisfied

③ Trailing(T) $\gtrdot$ *

{*,), id} $\gtrdot$ *

④ * $\lessdot \cdot$ Leading(F)

* $\lessdot \cdot$ {(, id}

$ $\gtrdot$ Trailing(S)

Leading(S) $\lessdot \cdot$ $

F → (E)

Here b, c, & d condition will statified

(5) c ≐)

(6) c <· Leading(E)

C < { +, *, (, id }

(7) Trailing(E) ·>)

{ +, *,), id } ·>)

id * id

	+	*	(	)	id	$
+	·>	<·	<·	·>	<·	·>
*	·>	·>	<·	·>	<·	·>
(	<·	<·	<·	≐	<·	e
)	·>	·>	e	·>	e	·>
id	·>	·>	e	·>	e	·>
$	<·	<·	<·	e	<·	accept

(	<·	<·	<·	=	<id	e
)	·>	·>	e	·>		·>
id	·>	·>	e	·>		
$	<·	<·	e			

Stack	Relation	Input	Action
$	$ <· id	id + id * id $	shift
$id	id ·> +	+ id * id $	Redu pop
$	$ <· +	+ id * id $	shift
$+	+ <· id	id * id $	shift
$+id	id ·> *	* id $	pop
$+	+ <· *	* id $	shift
$+*	* <· id	id $	shift
$+*id	id ·> $	$	pop
$+*	* ·> $	$	pop
$+	+ $	$	pop
$	$	$	accept

String = id + id * id

Stack	Relation	Input	Action
$	$ < id	id + id * id $	shift
$ id	id > +	+ id * id $	POP
$	$ < +	+ id * id $	shift
$ +	+ < id	id * id $	shift
$ + id	id > *	* id $	POP
$ +	+ < *	* id $	shift
$ + *	* < id	id $	shift
$ + * id	id > $	$	POP
$ + *	* > $	$	POP
$ +	+ > $	$	POP
$	$	$	accept