Ashritha.K
21B81A3307

ACD Assignment-1

1. Define a compiler. Briefly explain about the different phases of the compiler with example.

Compiler :- A compiler is a program that helps in translating the computer code from one programming language into another programming language.

Phases of a compiler

→ [compiler] → target program

Src prg

① Analysis phase (front end of compiler)

② Synthesis phase (back end of compiler)

Src prg
↓ character stream
[Lexical Analyzer] → reads chars from left to right one char at a time
↓ Tokens
[Syntax Analyzer]
↓ Parse tree
[Semantic Analyzer]
↓ Parse tree
[Intermediate Code Generator]
↓ Three address code
[Code optimization]
↓ Optimized 3 address code
[Code Generator]
↓ Assembly level language
Target prg

Ex: position = initial + rate * 60

id1 := id2 + id3 * 60

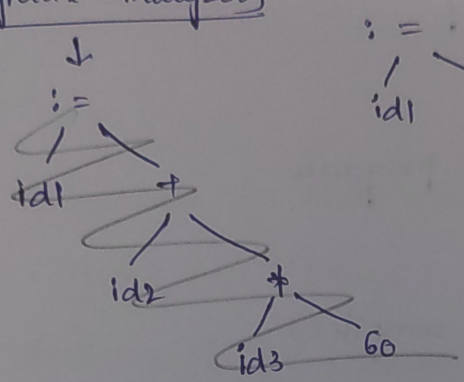
↓

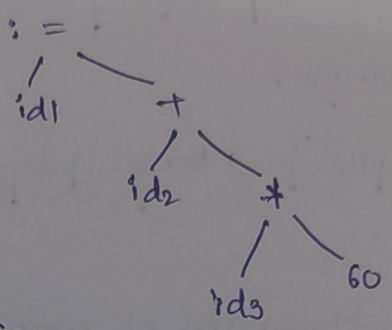
Lexical Analyzer

↓ id1, :=, id2, +, id3, *, 60

Syntax Analyzer

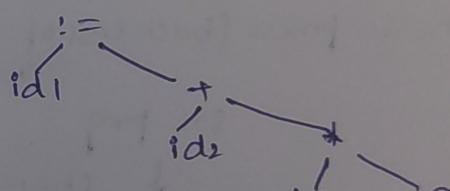
↓





Semantic Analyzer

↓



int to float (or) int to real

Intermediate Code Generator

↓

t1 = int to float(60)
t2 = id3 x t1
t3 = id2 + t2
id1 = t3

Code optimizer

↓

t1 = id3 * 60.0
id1 = id2 + t1

(or)

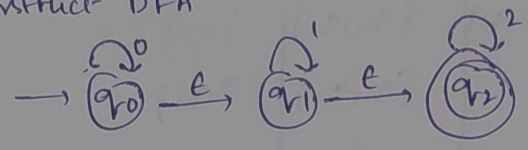
t1 = id3 x 60.0
t2 = id2 + t1
id1 = t2

Code Generator

↓

```
MOVF   id3 R2   // R2 = id3
MULF   # 60.0, R2   // R2 = R2 * 60.0
ADDF   id2 R1   // R1 = id2
ADDF   R2, R1   // R1 = R1 + R2
MOVF   R2, id1   // id1 = R1
```

2 Convert the given NFA with epsilon to NFA without epsilon and construct DFA



1) ϵ-closure $(q_0) = \{q_0, q_1, q_2\}$

ϵ-closure $(q_1) = \{q_1, q_2\}$

ϵ-closure $(q_2) = \{q_2\}$

2) Find δ' transitions for each state

$\delta'(q_0,0) = \epsilon$-closure $(\delta(\delta'(q_0,\epsilon),0))$

$= \epsilon$-closure $(\delta(\epsilon$-closure $(q_0),a))$

$= \epsilon$-closure $(\delta(q_0,q_1,q_2),0)$

$\delta'(q_0,0) = \epsilon$-closure $(q_0) = \{q_0,q_1,q_2\}$

$\delta'(q_0,1) = \epsilon$-closure $(\delta(\delta'(q_0,\epsilon),1))$

$= \epsilon$-closure $(\delta(q_0,q_1,q_2),1)$

$= \epsilon$-closure $(q_1) = \{q_1,q_2\}$

$\delta'(q_0,2) = \epsilon$-closure $(\delta(q_0,q_1,q_2),2)$

$= \epsilon$-closure $(q_2) = \{q_2\}$

$\delta'(q_1,0) = \epsilon$-closure $(\delta(\delta'(q_1,\epsilon),0))$

$= \epsilon$-closure $(\delta(q_1,q_2),0)) = \epsilon$-closure (ϕ)

$= \{\phi\}$

$\delta'(q_1,1) = \epsilon$-closure $(\delta(\delta'(q_1,\epsilon),1))$

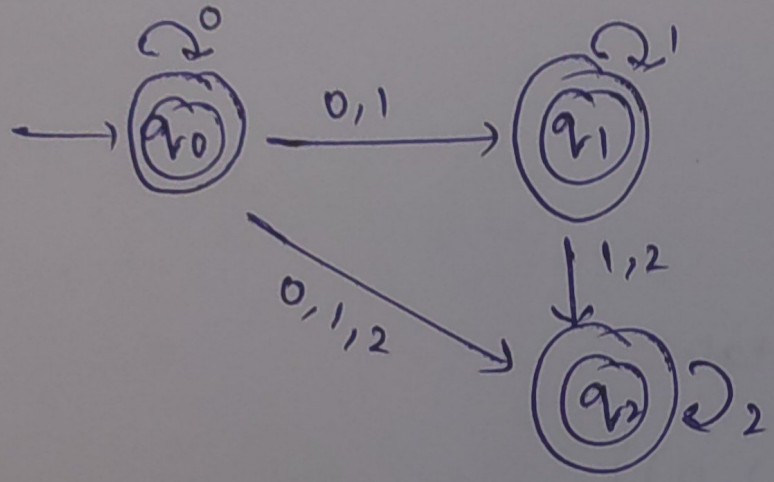
$= \{q_1,q_2\}$

$\delta'(q_1,2) = \epsilon$-closure $(\delta(q_1,q_2),2)$

$= \{q_2\}$

$\delta'(q_2,0) = \phi$ $\delta'(q_2,1) = \phi$ $\delta'(q_2,2) = \{q_2\}$

State \ I/P	0	1	2
q_0	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$
q_1	ϕ	$\{q_1, q_2\}$	$\{q_2\}$
q_2	ϕ	ϕ	$\{q_2\}$



$q_0, q_1 \& q_2$ are final states

2. conversion to DFA

→ $\delta'([q_0], 0) = \{q_0, q_1, q_2\}$

$\delta'([q_0], 1) = \{q_1, q_2\}$

$\delta'([q_0], 2) = \{q_2\}$

$\delta'([q_0, q_1, q_2], 0) = \{q_0, q_1, q_2\}$

$\delta'([q_0, q_1, q_2], 1) = \delta'([q_0], 1) \cup \delta'([q_1], 1) \cup \delta'([q_2], 1)$

$= \{q_1, q_2\}$

$\delta'([q_0, q_1, q_2], 2) = \delta'([q_0], 2) \cup \delta'([q_1], 2) \cup \delta'([q_2], 2)$

$= \{q_2\}$

$\delta'([q_1, q_2], 0) = \delta'([q_1], 0) \cup \delta'([q_2], 0)$

$= \phi$ -

$\delta'([q_1, q_2], 1) = \delta'([q_1], 1) \cup \delta'([q_2], 1)$

$= \{q_1, q_2\}$

$\delta'([q_1, q_2], 2) = \delta'([q_1], 2) \cup \delta'([q_2], 2)$

$= \{q_2\}$

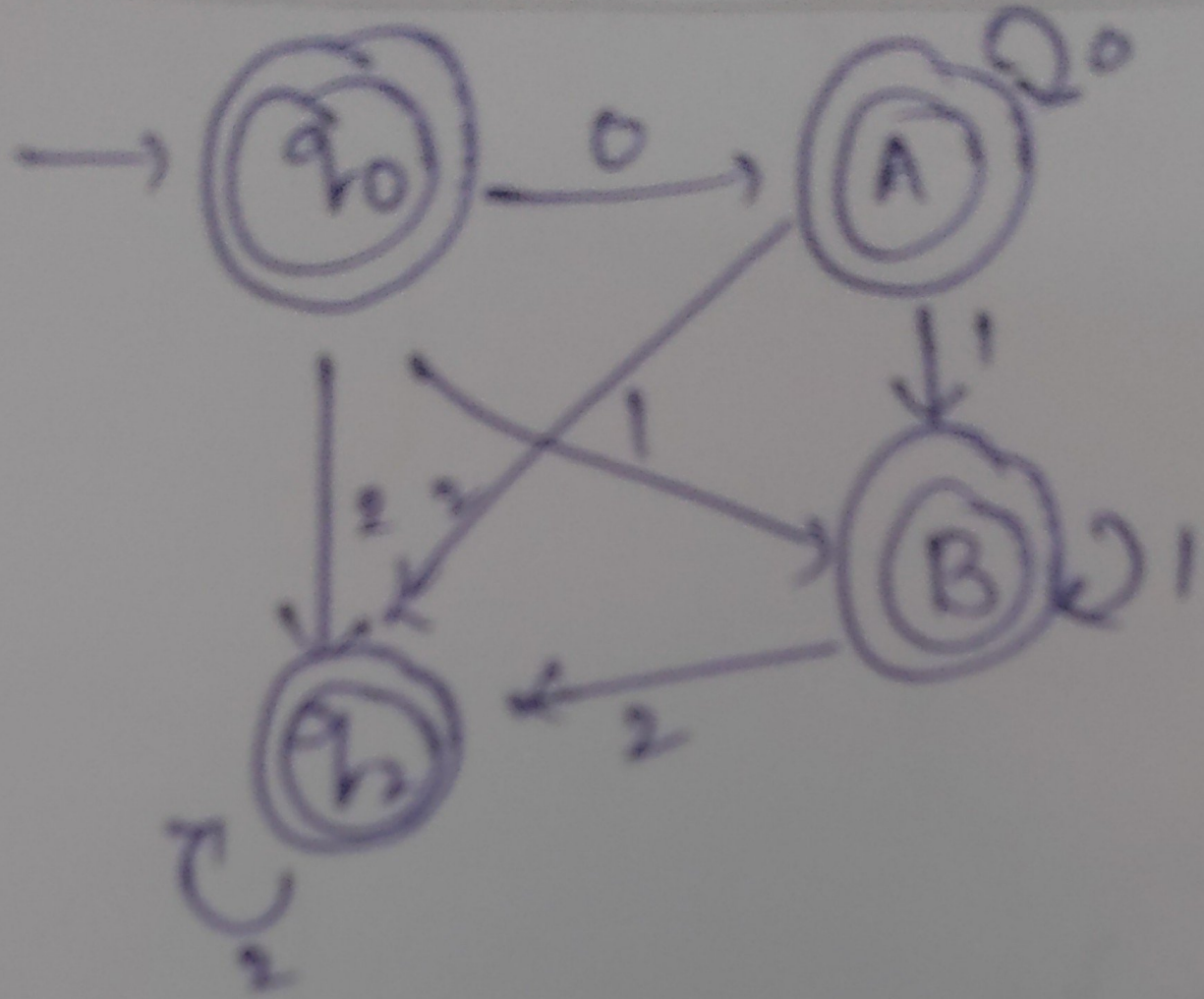
$\delta'([q_2], 0) =$ -

$\delta'([q_2], 1) =$ -

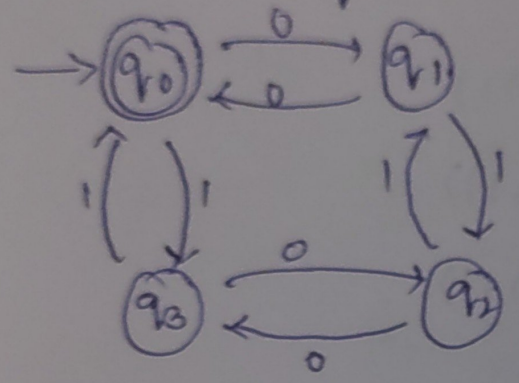
$\delta'([q_2], 2) = \{q_2\}$

DFA TT

State \ IP	0	1	2
q_0	$\{q_0, q_1, q_2\}^A$	$\{q_1, q_2\}^B$	q_2
A $\{q_0, q_1, q_2\}$	$\{q_0, q_1, q_2\}^A$	$\{q_1, q_2\}^B$	q_2
B $\{q_1, q_2\}$	-	$\{q_1, q_2\}^B$	q_2
q_2	-	-	q_2


q_0
0
A
0
1
1
2
2
B
1
q_2
2
2

3 Design FA with $\Sigma = \{0,1\}$ accepts even numbers of 0's and even number of 1's.



$\{\{q_0, q_1, q_2, q_3\}, \{0,1\}, \delta, q_0, \{q_0\}\}$

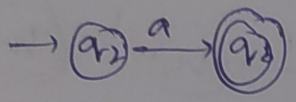
	0	1
q_0	q_1	q_3
q_1	q_0	q_2
q_2	q_3	q_1
q_3	q_2	q_0

4. Construct ϵ-NFA for the given regular expression $(a+b)^*abb$

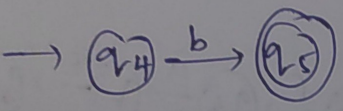
$r = (a+b)^* \; abb$
$\quad\; r_1 \quad\; r_2$

$r_1 = (a+b)^*$
$\quad\; r_3 \; r_4$

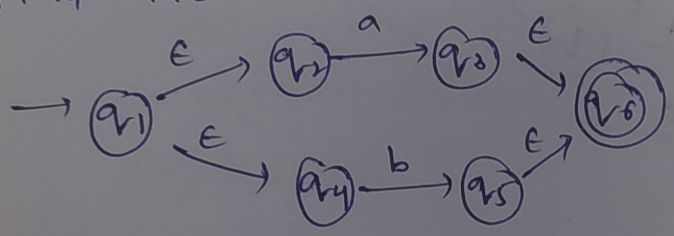
$r_3 = a$



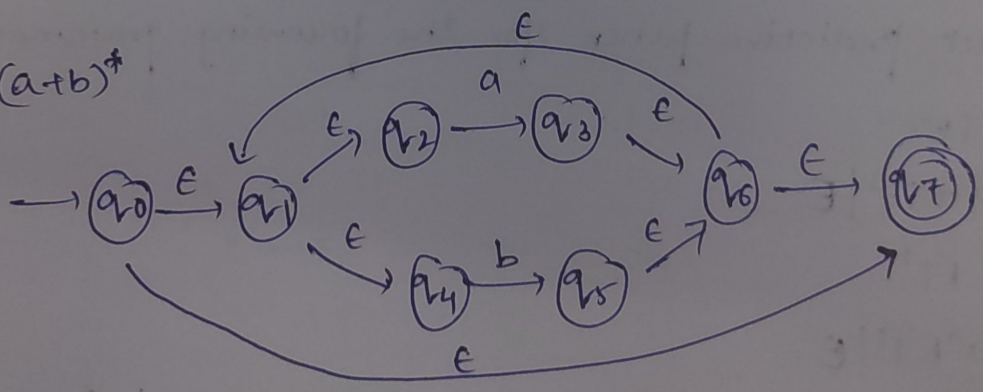
$r_4 = b$



$r_3 + r_4 = a+b$

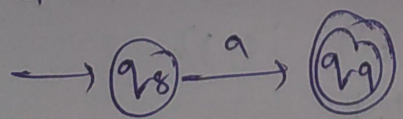


$r_1 = (a+b)^*$

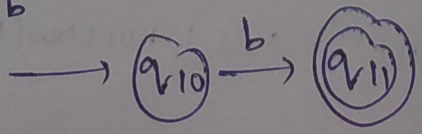


$r_2 = abb$
$\quad\; r_5 \; r_6 \; r_7$

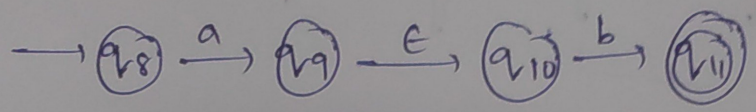
$r_5 = a$



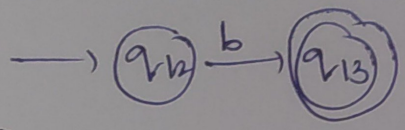
$r_6 = b$



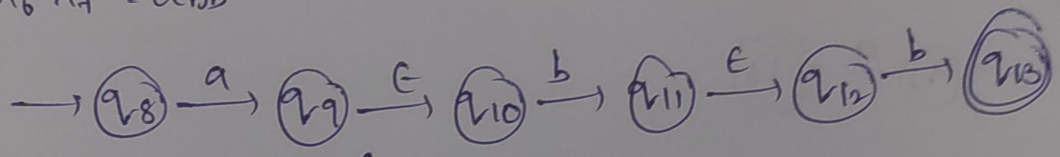
$r_5 \times r_6 = ab$



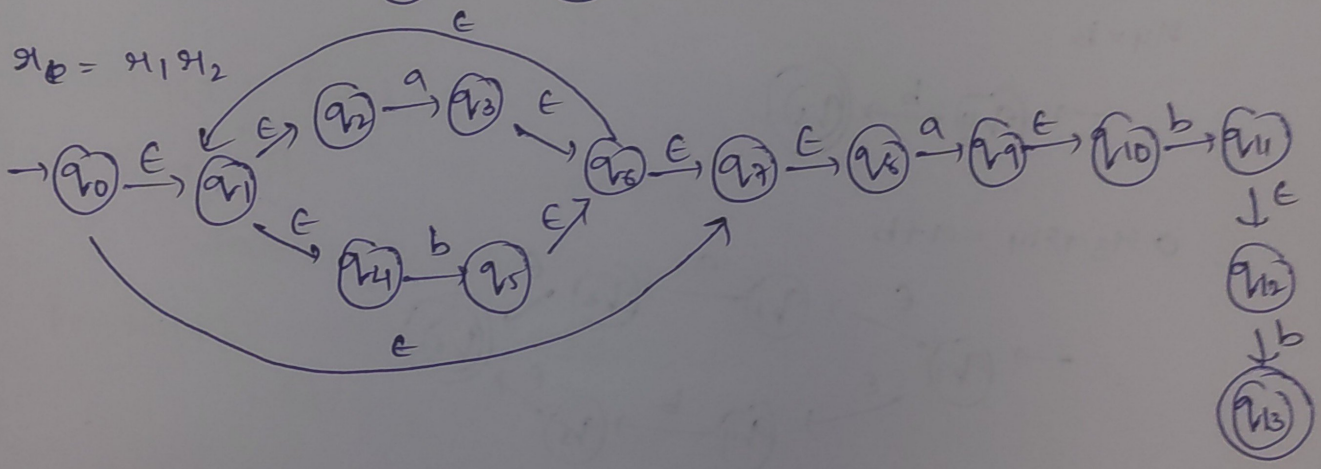
$r_7 = b$



$r_2 =$
$r_5 r_6 r_7 = abb$



$r_8 = r_1 r_2$



5. Construct predictive parser for the following grammar

$E \rightarrow TE'$

$E' \rightarrow +TE' | \epsilon$

$T \rightarrow FT'$

$T' \rightarrow *FT' | \epsilon$

$F \rightarrow id | (E)$ and parse the i/p string id+id*id

FIRST (E) = {(, id}

FIRST (E') = {+, ε}

FIRST (T) = {(, id}

FIRST (T') = {*, ε}

FIRST (F) = {(, id}

FOLLOW (E) = {$,)}

FOLLOW (E') = {$,)}

$E \rightarrow \overset{\alpha}{T}\overset{B}{E'}\overset{\beta}{}$

FOLLOW(E') = ε - ε ∪ FOLLOW(E) = {$,)}

$E' \longrightarrow +TE'$

FOLLOW(E') = {$,)}

FOLLOW(T) = {+, $,)}

$E \longrightarrow TE'$ (α B β) FOLLOW(T) = FIRST(E')

$= \{+, E\} - \epsilon \cup$ FOLLOW(E)

$= \{+, \$,)\}$

$E' \longrightarrow +TE'$ (α B β) FOLLOW(T) = {+, E} − ε ∪ FOLLOW(E')

$= \{+, \$,)\}$

FOLLOW(T') = {+, $,)}

$T \longrightarrow FT'$ (α B β) $T' \longrightarrow *FT' | \epsilon$

FOLLOW(T') = FOLLOW(T) = {+, $,)}

FOLLOW(F) = FIRST(T') − ε ∪ FOLLOW(T)

$= \{*, +, \$,)\}$

	First(α)	Pred Parsing Element
$E \to TE'$	(, id	M[E, (], M[E, id]
$E' \to +TE' \mid \epsilon$	+	M[E', +]
$E' \to \epsilon$	$,)	M[E', $], M[E',)]
$T \to FT'$	(, id	M[T, (], M[T, id]
$T' \to *FT'$	*	M[T', *]
$T' \to \epsilon$	+, $,)	M[T', +], M[T', $], M[T',)]
$F \to (E)$	(	M[F, (]
$F \to id$	id	M[F, id]

Terminals	+	*	(	)	$	id
E			$E \to TE'$			$E \to TE'$
E'	$E' \to +TE' / \epsilon$			$E' \to \epsilon$	$E' \to \epsilon$	
T			$T \to FT'$			$T \to FT'$
T'	$T' \to \epsilon$	$T' \to *FT'$		$T' \to \epsilon$	$T' \to \epsilon$	
F			$F \to (E)$			$F \to id$

Stack	I/P	O/P
$E	id+id*id$	$E \to TE'
$E'T	id+id*id$	$T \to FT'
$E'T'F	id+id*id$	$F \to id
$E'T'id	id+id*id$	$T' \to \epsilon
$E'	+id*id$	$E' \to +TE'
$E'T+	+id*id$	$T \to FT'
$E'T'	id*id$	$F \to id
$E'T'id	id*id$	$T' \to *FT'
$E'T'F*	*id$	$F \to id
$E'T'id	id$	$T' \to \epsilon
$E'	$	$E' \to \epsilon
$	$	Successful

6. Construct SLR parsing table for the following grammar and parse the I/P string aab.

1 $E \longrightarrow E+T$
2 $E \longrightarrow T$
3 $T \longrightarrow TF$
4 $T \longrightarrow F$
5 $F \longrightarrow F^*$
6 $F \longrightarrow a$
7 $F \longrightarrow b$.

1) $E' \longrightarrow E$

2) LR(0)

I_0:
$E' \longrightarrow .E$
$E \longrightarrow .E+T$
$E \longrightarrow .T$
$T \longrightarrow .Tf$
$T \longrightarrow .F$
$F \longrightarrow .F^*$
$F \longrightarrow .a$
$F \longrightarrow .b$

goto(I_0, E)

I_1:
$E' \longrightarrow E.$
$E \longrightarrow E.+T$

goto (I_0, T)

I_2:
$E \longrightarrow T.$
$T \longrightarrow T.F$
$F \longrightarrow .F^*$
$F \longrightarrow .a$
$F \longrightarrow .b$

3) goto(I_0, F)

I_3:
$T \longrightarrow F.$
$F \longrightarrow F.^*$

goto(I_0, a)

I_4:
$F \longrightarrow a.$

goto (I_0, b)

I_5:
$F \longrightarrow b.$

goto(I_1, +)

$E \to E+\cdot T$
$T \to \cdot TF$
$T \to \cdot F$
$F \to \cdot F*$
$F \to \cdot a$
$F \to \cdot b$

} I_6

goto(I_2, F)

$T \to TF\cdot$
$F \to F\cdot*$

} I_7

goto(I_2, a)

$F \to a\cdot$ } I_4

goto(I_2, b)

$F \to b\cdot$ } I_5

goto(I_3, *)

$F \to F*\cdot$ } I_8

goto(I_6, T)

$E \to E+T\cdot$
$T \to T\cdot F$
$F \to \cdot F*$
$F \to \cdot a$
$F \to \cdot b$

} I_9

goto(I_6, F)

$T \to F\cdot$
$F \to F\cdot*$

} I_3

goto(I_6, a)

$F \to a\cdot$ } I_4

goto(I_6, b)

$F \to b\cdot$ } I_5

goto(I_7, *)

$F \to F*\cdot$ } I_8

goto(I_9, F)

$T \to TF\cdot$
$F \to F\cdot*$

} I_7

goto(I_9, a)

$F \to a\cdot$ } I_4

goto(I_9, b)

$F \to b\cdot$ } I_5

4)

FOLLOW(E) = { $, + }

FOLLOW(T) = { $, +, a, b }

$E \to E \overset{\alpha}{+} T \overset{B}{} \overset{\beta}{}\cdot$

FOLLOW(E) = { $, + }

$T \to \overset{\alpha}{} \overset{B\cdot\beta}{TF}$

FOLLOW(F) = { $, +, *, a, b }

FIRST(E) = { a, b }

FIRST(T) = { a, b }

FIRST(F) = { a, b, * }

$E' \rightarrow E.$ I_1

② $E \rightarrow T.$ I_2

④ $T \rightarrow F.$ I_3

⑥ $F \rightarrow a.$ I_4

⑦ $F \rightarrow b.$ I_5

③ $T \rightarrow TF.$ I_7

⑤ $F \rightarrow F*.$ I_8

① $E \rightarrow E+T.$ I_9

SLR Parsing Table.

State	Action					goto		
	+	*	a	b	$	E	T	F
0			s4	s5		1	2	3
1	s6				Accept			
2	r2		s4	s5	r2			7
3	r4	s8	r4	r4	r4			
4	r6	r6	r6	r6	r6			
5	r7	r7	r7	r7	r7			
6			s4	s5			9	3
7	r3	s8	r3	r3	r3			
8	r5	r5	r5	r5	r5			
9	r1		s4	s5	r1			7

7. ~~Construct CLR and LALR parsing table for the +~~

Stack	I/P	Remarks
0	aab$	S4
0a4	ab$	r6 F→a pop 2
0F3	ab$	r4 T→F pop 2
0T2	ab$	S4
0T2a4	b$	r6 F→a pop 2
0T2F7	b$	r3 T→TF pop 4
0T2	b$	S5
0T2b5	$	r7 F→b pop 2
0T2F7	$	r3 T→TF pop 4
0T2	$	r2 E→T pop2
0E1	$	Accept

7 Construct a CLR & LALR parsing table for following grammar

S → AA

A → aA

A → b and parse the I/p string aab.

CLR

1) S' → .S, $

S → .AA, $

A → .aA, a|b

A → .b, a|b

} I_0

goto (I_0, S)

$S' \rightarrow S.$, \$ } I_1

goto (I_0, A)

$S \rightarrow A.A$, \$
$A \rightarrow .aA$, \$
$A \rightarrow .b$, \$
} I_2

first (E, \$) = \$

goto (I_0, a)

first (E, c/d) = c/d

$A \rightarrow a.A$, a/b
$A \rightarrow .aA$, a/b
$A \rightarrow .b$, a/b
} I_3

goto (I_0, b)

$A \rightarrow b.$, a/b } I_4

goto (I_3, b)

$A \rightarrow b.$, a/b } I_4

goto (I_6, A)

$A \rightarrow aA.$ } I_9

goto (I_9, b)

$A \rightarrow b.$, \$ } I_7

goto (I_2, A)

$S \rightarrow A.A.$, \$ } I_5

goto (I_2, a)

$A \rightarrow a.A$, \$
$A \rightarrow .aA$, \$
$A \rightarrow .b$, \$
} I_6

goto (I_2, b)

$A \rightarrow b.$, \$ } I_7

goto (I_3, A)

$A \rightarrow aA.$, a/b } I_8

goto (I_3, a)

$A \rightarrow a.A$, a/b
$A \rightarrow .aA$, a/b
$A \rightarrow .b$, a/b
} I_3

goto (I_6, a)

$A \rightarrow a.A$, \$
$A \rightarrow .aA$, \$
$A \rightarrow .b$, \$
} I_6

CLR parsing table.

State	Action			goto	
	a	b	$	S	A
0	S3	S4		1	2
1			Accept		
2	S6	S7			5
3	S3	S4			8
4	r3	r3			
5			r1		
6	S6	S7			9
7			r3		
8	r2	r2			
9			r2		

③ $A \rightarrow b\bullet, \$/a/b$ I_4

① $S \rightarrow AA\bullet, \$$ I_5

③ $A \rightarrow b\bullet, \$$ I_7

② $A \rightarrow aA\bullet, a/b$ I_8

② $A \rightarrow aA\bullet, \$$ I_9

State	I/p Buffer	Remarks
0	aab$	S3
0a3	ab$	S3
0a3a3	b$	S4
0a3a3b4	$	Error.

The given string is not accepted

LALR. Parsing table.

	a	b	$	S	A
0	s_{36}	s_{47}		1	2
1			Accept		
2	s_{36}	s_{47}			5
36	s_{36}	s_{47}			89
47	r_3	r_3	r_3		
5			r_1		
89	r_2	r_2	r_2		

State	I/p Buffer	Remarks
0	aab$	s_{36}
0a36	ab$	s_{36}
0a36a36	b$	s_{47}
0a36a36~~b47~~	$	r_3 A → b pop 2
0a36~~a36A89~~	$	r_2 A → aA pop 4
0~~a36A89~~	$	r_2 A → aA pop 4
0A2	$	Error.

The given string is not accepted.

8. Discuss about S-attributed and L-attributed SDTs in syntax directed translation

S-attributed

→ It uses only synthesized attribute
→ Semantic actions are placed at right end of a production
Ex: A → BC{}
→ Attributes are evaluated during bottom up parsing
→ It is based on LR grammar
→ These are implemented using LALR parser

L-Attributed

→ It uses both synthesized & inherited attribute
→ Semantic actions are placed anywhere
Ex: A → {}BC, A → BC{}, A → B{}C
→ Attributes are evaluated by translating parse tree to depth first
→ It is based on LL grammar
→ Implemented using predictive parser.