SRM Institute of Science and Technology
College of Engineering and Technology
SCHOOL OF COMPUTING
SRM Nagar, Kattankulathur – 603203, Chengalpattu District, Tamilnadu

SET-B

Academic Year: 2022-23 (EVEN)

Test: CLAT-2 **Date : 04.04.2023**
Course Code & Title: 18CSC304J -COMPILER DESIGN **Duration : 2 Periods**
Year & Sem: III & VI **Max. Marks: 50**

Course Articulation Matrix:

S.No.	Course Outcome	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
1	CO3	3	3	3									

Part – A (10 X 1 = 10 Marks)
Answer ALL Questions

Q. No	Question	Marks	BL	CO	PO	PI Code
1	The grammar **A → Ax | (A) | ε** is not suitable for predictive-parsing because the grammar is? a) Left factoring **b) Left recursive** c) Right recursive d) An operator grammar Answer: B	**1**	**1**	**3**	**1**	**1.7.1**
2	For the grammar, **E → EE | (E) | ε**, number of parse trees to produce empty string is? a) One b) Two c) Three **d) Infinite** Answer: D	**1**	**1**	**3**	**1**	**1.7.1**
3	Which grammar rules violate the requirements of an operator grammar? 1. E → FG 2. F → E s F 3. G → F t H p 4. H → ε a) 1 only b) 1 and 3 only **c) 1 and 4 only** d) 1, 3 and 4 only Answer: C	**1**	**2**	**3**	**2**	**2.8.2**
4	A form of recursive descent parsing that does not require any back-tracking is known as? a) recursive parsing b) non-recursive parsing **c) predictive parsing** d) non-predictive parsing Answer: C	**1**	**1**	**3**	**1**	**1.7.1**

5	For the grammar given below, find FIRST(X) X → Ya | bZ Y → c | ε a) {a, b} b) {c, ε} c) {a, b, c} **d) {a, b, c, ε}** Answer: D	**1**	**1**	**3**	**1**	**1.7.1**
6	In shift reduce parsing handle is at _____ **a) Top of the stack** b) Bottom of the stack c) Anywhere in the stack d) Nowhere in the stack Answer: A	**1**	**2**	**3**	**2**	**2.8.2**
7	Choose the correct precedence relations in operator precedence parsing: if operator O_1 has higher precedence than operator O_2 a) $O_1 \cdot> O_2$ b) $O_1 =\cdot O_2$ c) $O_2 <\cdot O_1$ **d) $O_1 \cdot> O_2$ and $O_2 <\cdot O_1$** Answer: D	**1**	**2**	**3**	**4**	**4.6.2**
8	In SLR, CLR and LALR parser, which have same number of states? a) SLR and CLR **b) SLR and LALR** c) CLR and LALR d) SLR, CLR and LALR Answer: B	**1**	**2**	**3**	**1**	**1.7.1**
9	What is the LEADING(X) for the following grammar? X → X – B | B B →B*A | A A → (X) | id a) LEADING(X)={-,*,(,)} **b) LEADING(X)={-,*,),id}** c) LEADING(X)={-,*,(,id} d) LEADING(X)={-,*,(} Answer: B	**1**	**2**	**3**	**4**	**4.6.2**
10	Construction of parsing table in which strategies do not need the FOLLOW set? a) SLR and CLR b) SLR and LALR **c) CLR and LALR** d) SLR, CLR and LALR Answer: C	**1**	**1**	**3**	**3**	**3.8.2**

SRM Institute of Science and Technology
College of Engineering and Technology
SCHOOL OF COMPUTING
SRM Nagar, Kattankulathur – 603203, Chengalpattu District, Tamilnadu

SET-B

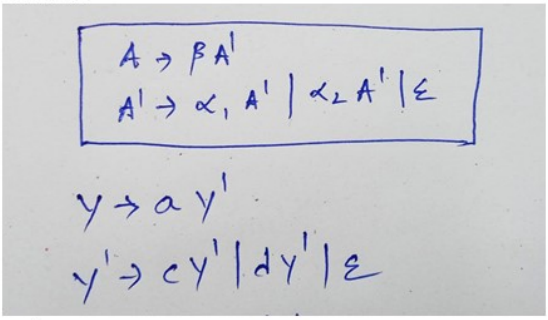
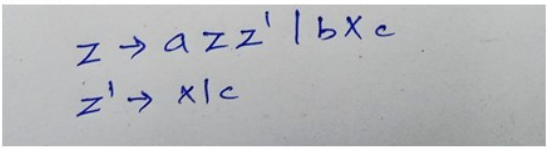
Academic Year: 2022-23 (EVEN)

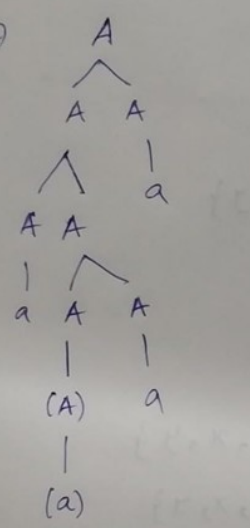
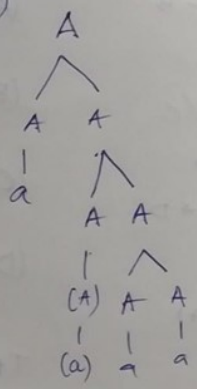
Test: CLAT-2 **Date : 04.04.2023**
Course Code & Title: 18CSC304J -COMPILER DESIGN **Duration : 2 Periods**
Year & Sem: III & VI **Max. Marks: 50**

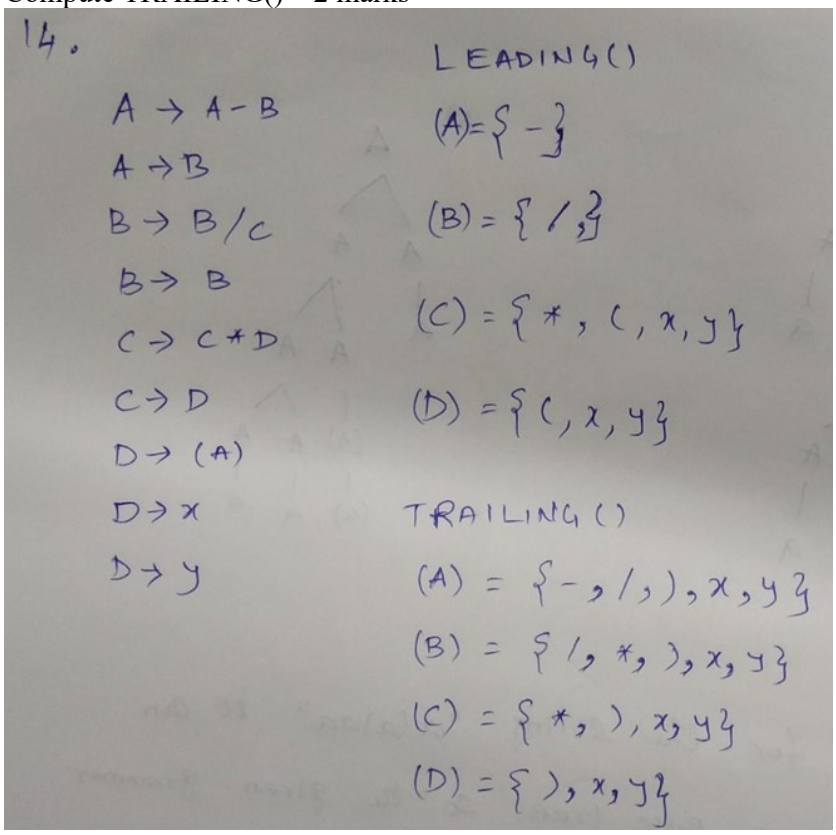
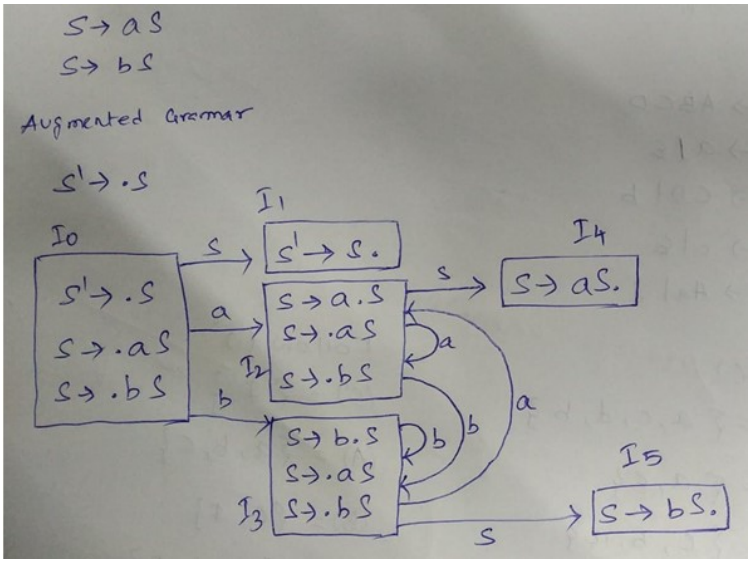
Course Articulation Matrix:

S.No.	Course Outcome	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
1	CO3	3	3	3									

PART B (4 x 4 = 16) ANSWER ANY FOUR

11	Eliminate left recursion and left factoring in the following grammar: X → Ya | b | c Y → Yc | Yd | a Z → aZX | bXc | aZc **Left recursion – 2 marks**  **Left factoring – 2 marks**  	**4**	**2**	**3**	**3**	**3.8.2**
12	Check the following grammar is ambiguous or not by parsing the input string "**a(a)aa**": A→ AA A → (A) A → a	**4**	**2**	**3**	**4**	**4.6.2**

	12. a)  b)  for the string "a(a)aa" it can generate 2 Parse tree, so the given grammar is Ambiguous.					
13	Compute FIRST() and FOLLOW() for the grammar: S → ABCD A → a | ε B → CD | b C → c | ε D → Aa | d Compute FIRST() – 2 marks Compute FOLLOW() – 2 marks 13. S → ABCD A → a | ε B → CD | b C → c | ε D → Aa | d FIRST() (S) = { a, c, d, b } (A) = { a, ε } (B) = { c, b, a } (C) = { c, ε } (D) = { a, d, ε } Follow () (S) = { $ } (A) = { a, b, c } (B) = { c, $ } (C) = { a, d, $ } (D) = { c, $ }	**4**	**3**	**3**	**3**	**3.8.2**
14	Find LEADING() and TRAILING() for all the non-terminals in the following grammar: A → A – B | B	**4**	**3**	**3**	**3**	**3.8.2**

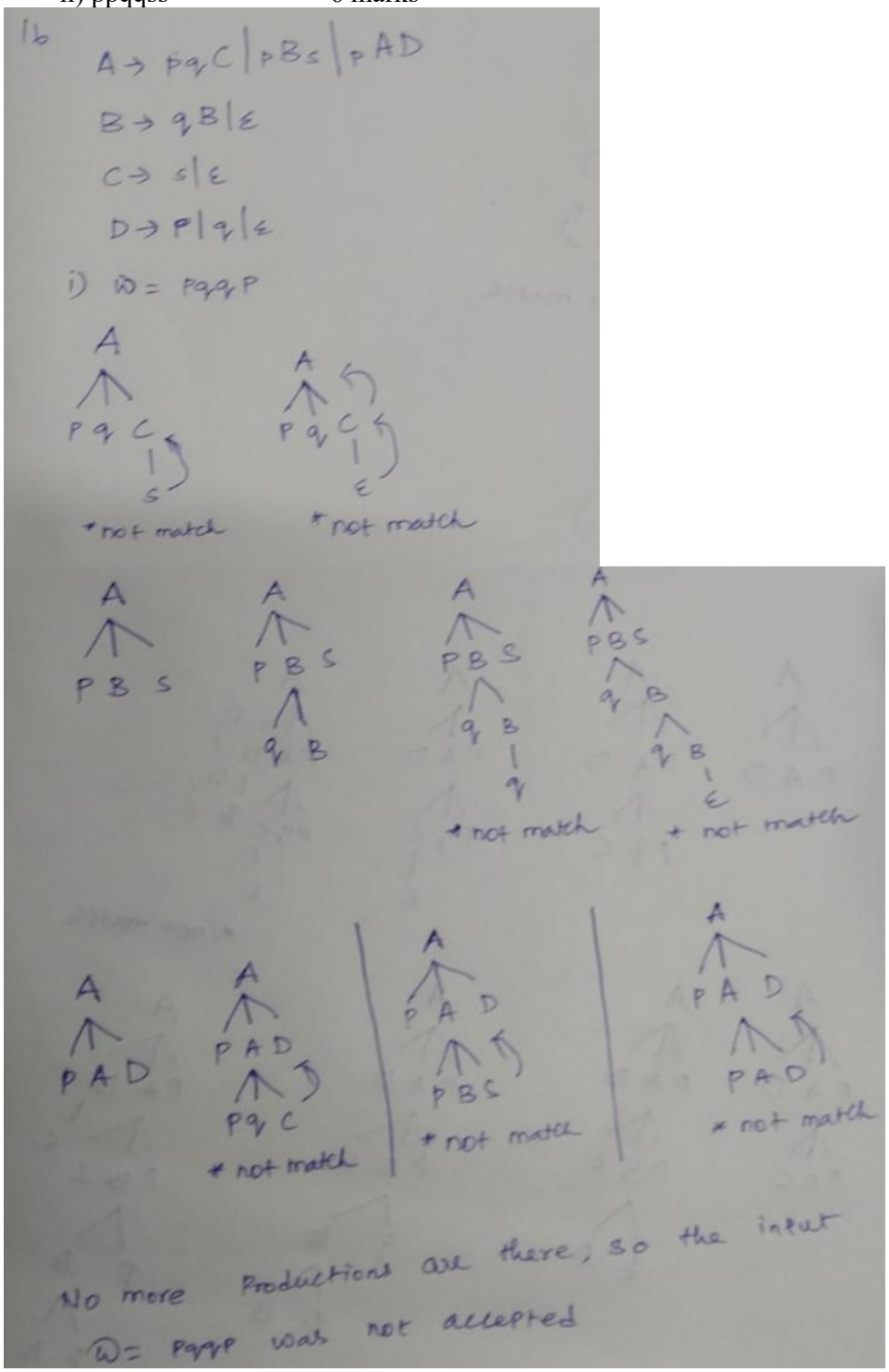
	B → B / C | B C → C * D | D D → (A) | x | y Compute LEADING() – 2 marks Compute TRAILING() – 2 marks  					
15	Find the canonical collection of LR(0) items for the following grammar: S → aS | bS  	**4**	**3**	**3**	**2**	**2.6.4**
	Part – C (2 x 12 = 24 Marks) Answer ALL Questions					
16.	Consider the grammar: A → pqC | pBs | pAD B → qB | ε C → s | ε	**12**	**2**	**3**	**3**	**3.8.2**

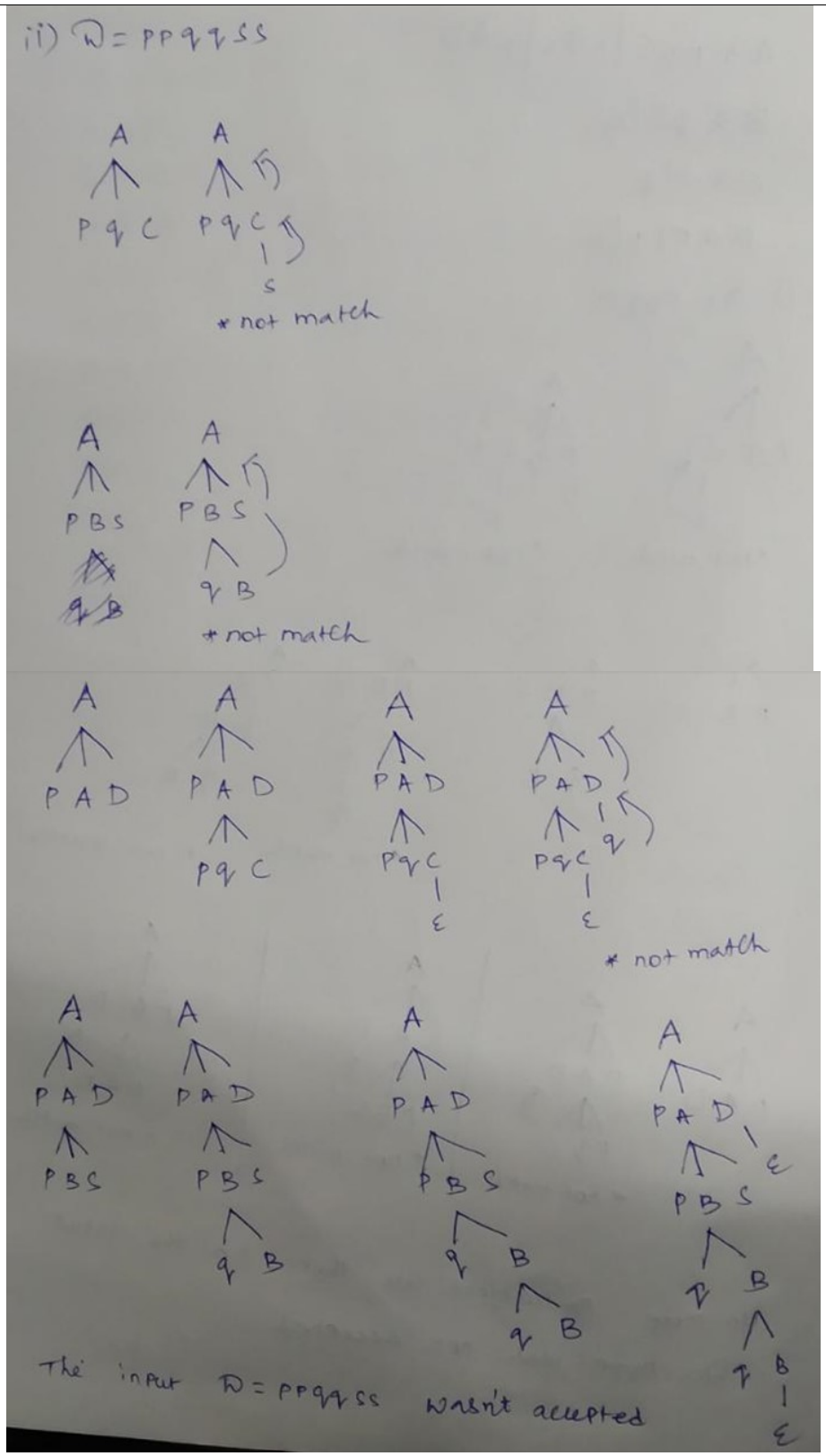
D → p | q | ε

Check whether the following inputs are accepted or not by the grammar using recursive decent parsing:

i) pqqp - 6 marks

ii) ppqqss - 6 marks



	(OR)					
17.	Election commission has announced the MLA election for Kanchipuram constituency. In view of this, applications are invited for the MLA election nomination. A candidate should produce proof for Age, Qualification, and any Work experience. The basic criteria for age limit is Age>20 and Age<50. The academic qualification can be UG or PG or Diploma or no qualification. Then, it includes whether the candidate has a work experience or not. Construct CFG for	**12**	**3**	**3**	**4**	**4.6.2**

	the given scenario and parse the below input strings using Shift Reduce parsing. Input: i) 31 dip no ii) 20 ug yes **Grammar:** **- 6 marks** M → AQE A → 2X | 3Y | 4Y X → 1 | 2 | 3 | … | 9 Y → 0 | X Q → *ug* | *pg* | *dip* | ε E → *yes* | *no* **Parsing the inputs:** i) 31 dip no - 3 marks **Stack** / **Input** / **Action** $ / 31 dip no $ / Shift $ 3 / 1 dip no $ / Shift $ 31 / dip no $ / Reduce X → 1 $ 3X / dip no $ / Reduce Y → X $ 3Y / dip no $ / Reduce A → 3Y $ A / dip no $ / Shift $ A dip / no $ / Reduce Q →dip $ AQ / no $ / Shift $ AQ no / $ / Reduce E → no $ AQE / $ / Reduce M →AQE $ M / $ / Accept ii) 20 ug yes - 3 marks **Stack** / **Input** / **Action** $ / 20 ug yes $ / Shift $ 2 / 0 ug yes $ / Shift $ 20 / ug yes $ / Reduce Y → 0 $ 2Y / ug yes $ / Shift $ 2Y ug / yes $ / Reduce Q → ug $ 2YQ / yes $ / Shift $ 2YQ yes / $ / Reduce E → yes $ 2YQE / $ / Not Accepted					
18.	Consider the following grammar: S → (L) | a L → L,S | S Construct operator precedence parsing table using Leading and Trailing, precedence graph and precedence function	**12**	**2**	**3**	**3**	**3.8.2**

18.

$S \to (L)$

$S \to a$

$L \to L, S$

$L \to S$

LEADING()

$(S) = \{ (, a \}$

$(L) = \{ , , (, a \}$

TRAILING()

$(S) = \{), a \}$

$(L) = \{ , ,), a \}$

Three types of precedence = $\{ <\cdot, \cdot>, = \}$

Rule ①

set $ < a ; for all 'a' Lead(S)

set b > $; for all 'b' Trail(S)

$ < Lead(S)

$ < { (, a }

Trail(S) > $

{), a } > $

Rule ②

$A \to xy$; if xy are terminal set x=y

NOT APPLICABLE

Rule ③

$A \to xBy$; if 'xy' are terminal 'B' is nonTerminal

set x = y

$S \to (L)$

(=)

Rule (4)

$A \rightarrow \alpha x B \beta$; if 'x' is terminal and 'B' non Terminal

set $x < Lead(B)$

$S \rightarrow (L)$ $L \rightarrow L, S$

$(< \{), (, a \}$ $, < \{ (, a \}$

Rule (5)

$A \rightarrow \alpha B x \beta$; if 'x' is terminal and 'B' non Terminal

set $Trail(B) > x$

$S \rightarrow (L)$ $L \rightarrow L, S$

$\{),), a \} >)$ $\{),), a \} >)$

Rule (6)

$, $ = Accepted

	a	,	(	)	$
a		>		>	>
,	<	>	<	>	
(	<	<	<	=	
)		>		>	>
$	<		<		Accepted

(OR)

19.	Consider the following grammar: S → L = R | R L → *R | id R → L Construct the SLR parser table for the grammar. Show action of the parser, for the input string ***id=id**	**12**	**3**	**3**	**3**	**3.8.2**

Given Grammar:

S → L = R
S → R
L → *R
L → id
R → L

Step-1: Augmented grammar - 1 mark

S' → S
S → L = R
S → R
L → *R
L → id
R → L

Step-2: Find LR(0) collections - 3 marks

I_0:
S' →.S
S → .L = R
S → .R
L → .*R
L → .id
R → .L

I_1: goto(I0, S)
S^1 →S.

I_2: goto(I0, L)
S → L. = R
R → L.

I_3: goto(I0, R)
S → R.

I_4: goto(I0, *)
L → *.R
R → .L
L → .*R
L → .id

I_5: goto(I0, id)
L → id.

I_6: goto(I2, =)
S → L =. R
R → .L
L → .*R
L → .id

I_7: goto(I4, R)
L → *R.

I_8: goto(I4, L)
R → L.

I_4: goto(I4, *)
L → *.R
R → .L
L → .*R
L → .id

I_5: goto(I4, id)
L → id.

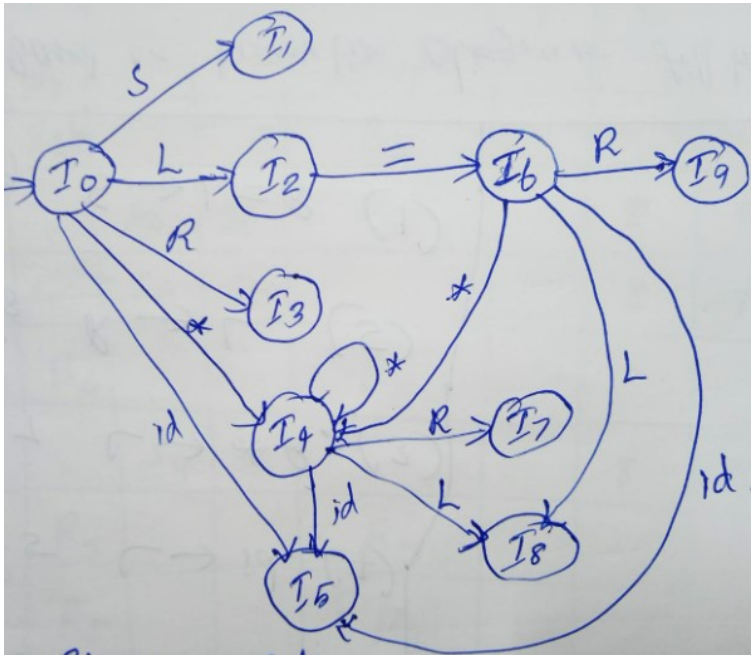
I_9: goto(I6, R)
S →L=R.

I_8: goto(I6, L)
R → L.

I_4: goto(I6, *)
L → *.R
R → .L
L → .*R
L → .id

I_5: goto(I6, id)
L → id.

Transition Diagram of goto function - 2 marks



Step-3: Find FOLLOW() - 1 marks

FOLLOW(S') = { $ }
FOLLOW(S) = { $ }
FOLLOW(L) = { =, FOLLOW(R) } = { =, $ }
FOLLOW(R) = { FOLLOW(S), FOLLOW(L) } = { $, = }

Step-4: Construct the parsing table - 3 marks

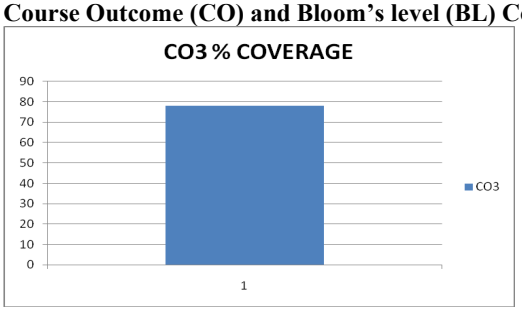
	=	*	id	$		S	L	R
0		S4	S5			1	2	3
1				Accept				
2	S6/R5			R5				
3				R2				
4		S4	S5				8	7
5	R4			R4				
6		S4	S5				8	9
7	R3			R3				
8	R5			R5				
9				R1				

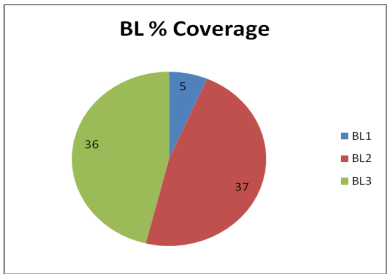
Parsing the input: ***id = id** **- 2 marks**

Stack	Input	Action
0	*id = id $	Shift 4
0 * 4	id = id $	Shift 5
0 * 4 id 5	= id $	Reduce by L→ id
0 * 4 L 8	= id $	Reduce by R → L
0 * 4 R 7	= id $	Reduce by L → * R
0 L 2	= id $	Shift 6
0 L 2 = 6	id $	Shift 5
0 L 2 = 6 id 5	$	Reduce by L → id
0 L 2 = 6 L 8	$	Reduce by R → L
0 L 2 = 6 R 9	$	Reduce by S → L = R
0 S 1	$	Accept

***Performance Indicators are available separately for Computer Science and Engineering in AICTE examination reforms policy**

Course Outcome (CO) and Bloom's level (BL) Coverage in Questions





Approved by the Audit Professor/Course Coordinator