VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering

Fall Semester 2024-25

CAT I

SLOT: C2+TC2

Programme Name & Branch: B.Tech

Course Name & Code: BCSE307L – Compiler Design

Class Number (s): VL2024250101546, VL2024250101550, VL2024250101556, VL2024250101591, VL2024250101607, VL2024250101617, VL2024250101627, VL2024250101635, VL2024250101644, VL2024250101655, VL2024250101665, VL2024250101671, VL2024250101674, VL2024250101679, VL2024250101688, VL2024250101729, VL2024250101738, VL2024250102054, VL2024250108000

Faculty Name (s): PROF. SAHAAYA ARUL MARY S A, PROF. KANNADASAN R, PROF. VISHNUPRIYA, PROF. VETRISELVI T, PROF. BHUVANESWARI M, PROF. KANAGARAJ R, PROF. KALAIVANI K, PROF. SATHYA K, PROF. BASKARAN P, PROF. SABYASACHI KAMILA, PROF. UMA PRIYA D, PROF. MUKKU NISANTH KARTHEEK, PROF. ARUMUGA ARUN R, PROF. ISLABUDEEN M, PROF. SUGANTHINI C, PROF. NAGA PRIYADARSINI R, PROF. BHAWANA TYAGI, PROF. BAIJU B V, PROF. UMAMAHESWARI M

Exam Duration: 90 Min. **Maximum Marks: 50**

ANSWER ALL THE QUESTIONS

Q. No.	Question	Max Marks
1.	Convert the regular expression **(a|b)*abb(a|b)*** into deterministic finite automata using direct method.	10

2.	a) Identify the lexemes that make up the tokens in the following program and give reasonable attributes values for the tokens.
	
	int main()
	{ //To calculate the prime number
	for (j = 2; j <= num / 2; j++) {
	if ((num % j) == 0) {
	flag = 1;
	break;
	}
	}
		5+5

	if (flag == 0) printf("%d is a prime number \n", num); else printf("%d is not a prime number \n", num); } b) Construct the operator precedence relation table for the following CFG where S is the starting symbol. **S → SAS | (S) | S** **A→ ↑ | *** Parse the string **a↑b*c** using the relation table constructed.	
3.	a. Illustrate all phases of compilation and generate assembly language code for the ***computation of simple interest.*** b. Construct a symbol table for the above case.	10
4.	In the following context-free grammar, the symbols a, b, c and d are terminals and P is the initial symbol. **P → a | b P c P d |b Q d** **Q → P | Q P** i. Explain briefly why this grammar is not LL(1). ii. Convert this grammar to an equivalent that is LL(1). iii. For the grammar of the previous subtask, construct the complete LL(1) parsing table. **iv.** Show all the steps required to parse the input string: **bbacadabadd**	10
5.	a) Write the Recursive Procedure based Top-Down parsing for the following CFG. S → AB A → aA | bB | Ac | Ab B → b | aB b) Check whether the string w=acadb is parsed by the above CFG.	10

18CS2071 - COMPILER DESIGN
CAT-1-1 - C2 SLOT

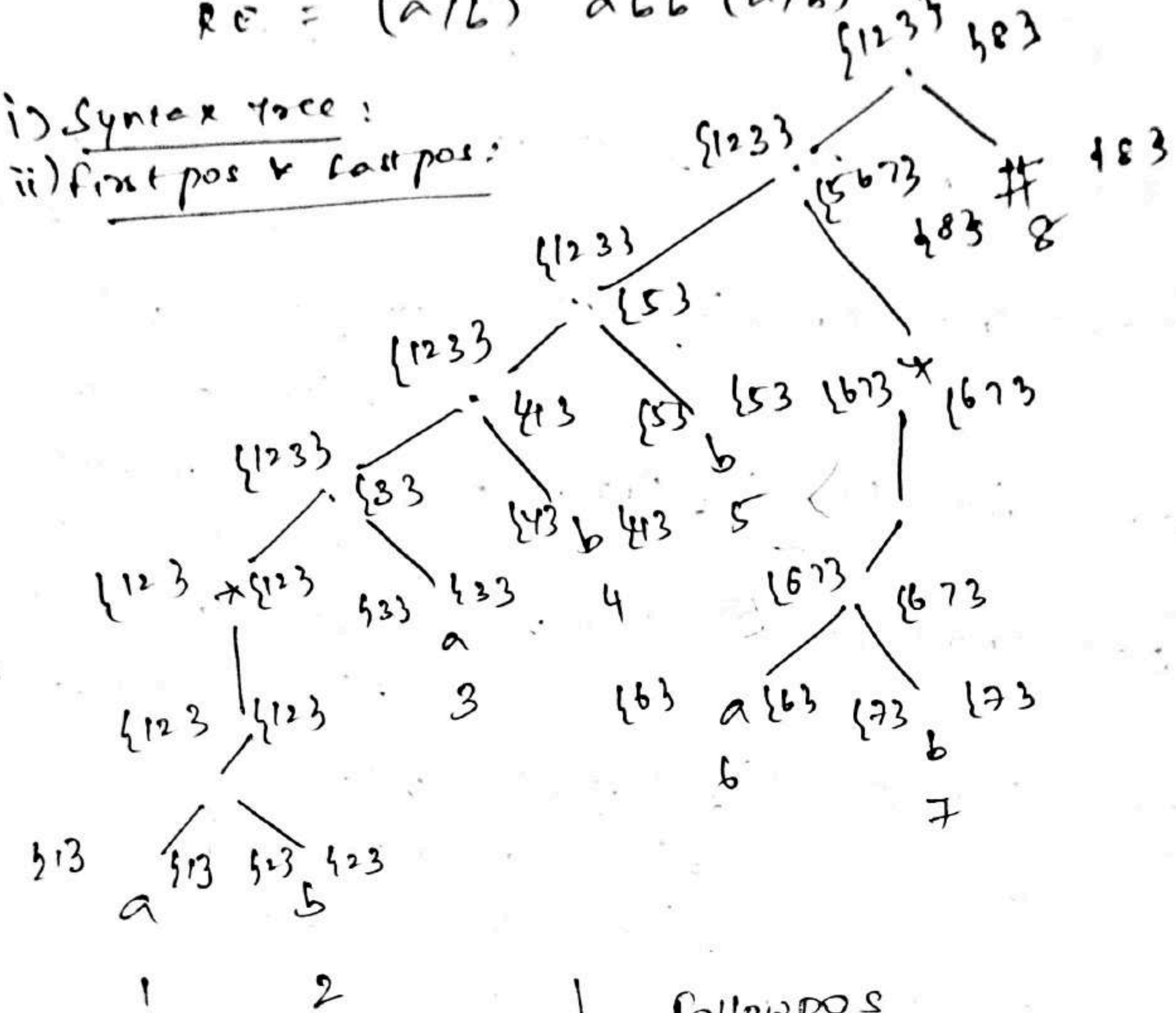
(2) RE to DFA.

$L = \{ x\,abb\,y \mid x, y \in (a+b)^* \}$

$RE = (a/b)^* \, abb \, (a/b)^*$

i) Syntax tree:

ii) firstpos & lastpos:



iii) followpos:

	followpos
1	{1 2 3}
2	{1 2 3}
3	{4}
4	{5}
5	{6 7 8}
6	{6 7 8}
7	{6 7 8}
8	#

Dtransition A = {1, 2, 3}

Dtran (move (A, a)) = {1} ∪ {4}
= {1 2 3} ∪ {4}
= {1 2 3 4} = B

Dtran (move (A, b)) = {2}
= {1 2 3} = A

Dtran (move (B, a)) = {1} ∪ {3} = {1 2 3} ∪ {4}
= {1 2 3 4} = B

Dtran (move (B, b)) = {2} ∪ {4} = {1 2 3} ∪ {5}
= {1 2 3 5} = C

Dtran (move (C, a)) = {1} ∪ {3} = {1 2 3} ∪ {4}
= {1 2 3 4} = B

Dtran (move (C, b)) = {2} ∪ {5} = {2 3} ∪ {6 7 8}
= {1 2 3 6 7 8} = D

Dtran (move (D, a)) = {1} ∪ {3} ∪ {6}
= {1 2 3} ∪ {4} ∪ {6 7 8}
= {1 2 3 4 6 7 8} = E

Dtran (move (D, b)) = {2} ∪ {7}
= {1 2 3} ∪ {6 7 8} = {1 2 3 6 7 8}
= D

Dtran (move (E, a)) = {1} ∪ {3} ∪ {6}
= {1 2 3} ∪ {4} ∪ {6 7 8}
= {1 2 3 4 6 7 8} = E

Dtran (move (E, b)) = {2} ∪ {4} ∪ {7}
= {1 2 3} ∪ {5} ∪ {6 7 8}
= {1 2 3 5 6 7 8} = F

$Dbar(move(F,a)) = \{1\} \cup \{3\} \cup \{6\}$

$= \{1,3\} \cup \{4,3\} \cup \{6,7,8\}$

$= \{1,3,4,6,7,8\} = E$

$Dbar(move(F,b)) = \{2\} \cup \{5\} \cup \{7\}$

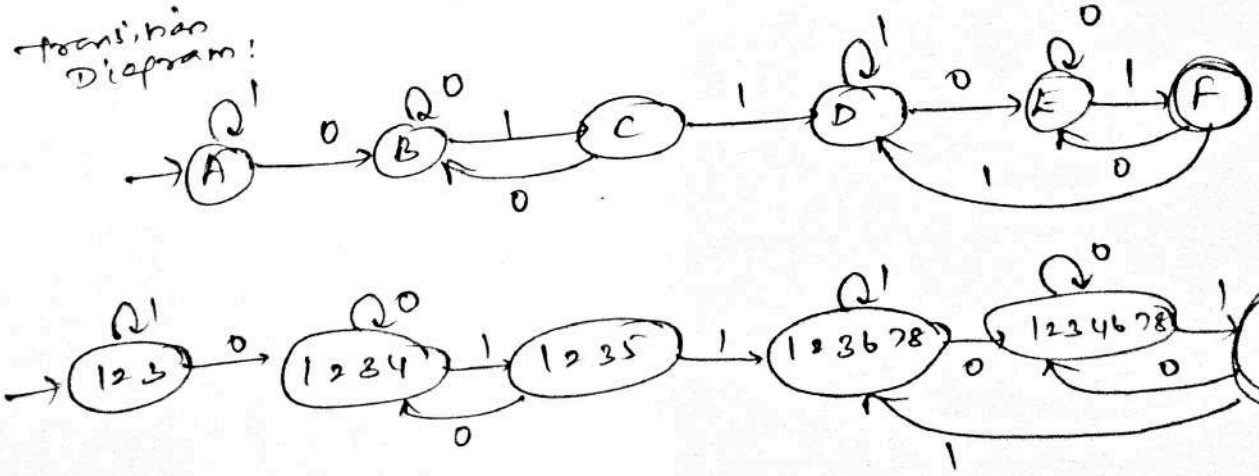
$= \{1,3\} \cup \{6,7,8\} \cup \{6,7,8\}$

$= \{1,3,6,7,8\} = D$

Transition Table :

	0	1
→ A	B	A
B	B	C
C	B	D
D	E	D
E	E	F
F	E	D

Transition Diagram :



(4)

(2)
(5)

"List few below"

Lexeme	Token	Attribute Value
init	keyword_init	keyword
main	keyword_main	keyword
(		
)		
{		
for	keyword_for	keyword
(		
j	id	1
2	Num/constant	2
;		
j	id	2
<=	relop	LE
num	id	3
/	operator	DIV
2	num/constant	2
j	id	4
++	Inc. operator	
)		
{		
if	if	—
(		
(	id	5
num		
%		
j	id	6
)		
==	Relop	EQ

(2) b) operator precedence

	id	↑	*	(	)	$
id		·>	·>		·>	·>
↑	<·	<·	·>	<·	·>	·>
*	<·	<·	·>	<·	·>	·>
(	<·	<·	<·	<·	≐	
)		·>	·>		·>	·>
$	<·	<·	<·	<·		

id ↑ id * id $

id ·> ↑ <· id ·> * <· id ·> $

(3) (a)

SI = p * n * r / 100

↓

[LEXICAL ANALYZER]

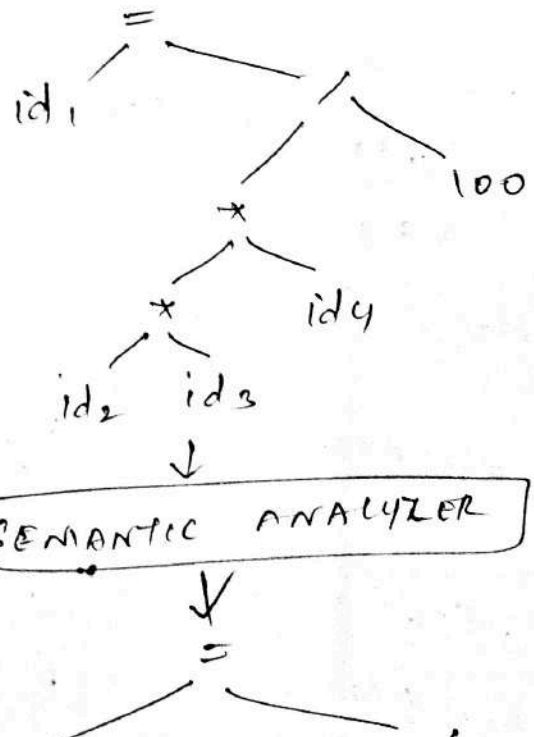
↓

$id_1 = id_2 * id_3 * id_4 / 100$

↓

[SYNTAX ANALYZER]

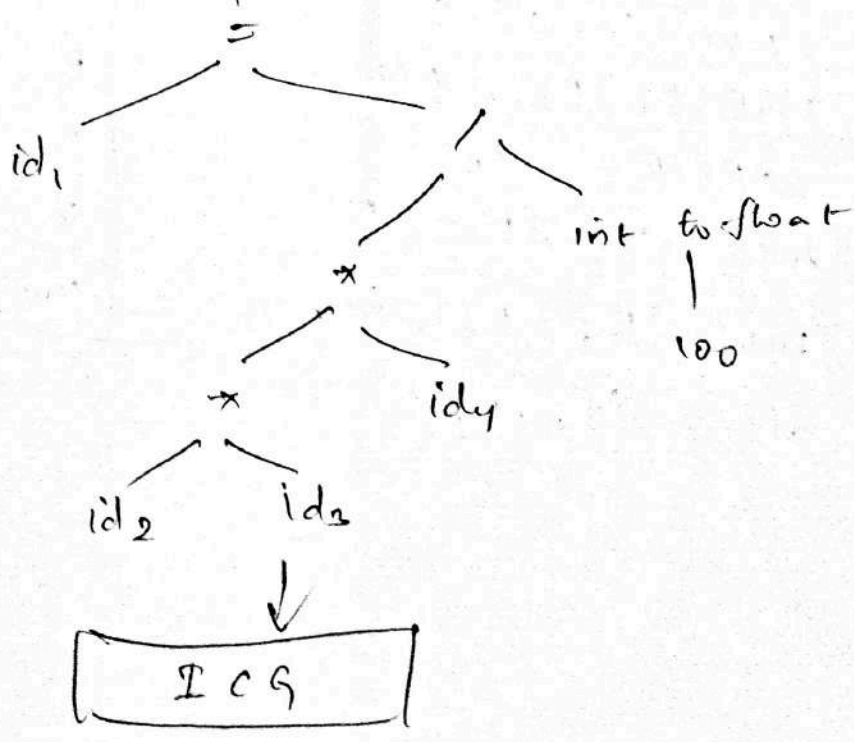
↓



↓

[SEMANTIC ANALYZER]

↓



↓

[ICG]

Note:
Depends on the proof how they have handled data type ("type checking")

(7)

INTERMEDIATE CODE GENERATION (ICG)

↓

```
t1 = int to float (100)
t2 = id4 / t1
t3 = id3 * t2
id1 = id2 * t3
```

↓

CODE OPTIMIZER

↓

```
t1 = id4 / inttofloat (100)
t2 = id3 * t1
id1 = id2 * t2
```

↓

CODE GENERATOR

↓

```
LDF  R1 , id4
DIVF R1 , R1 , #100.0
LDF  R2 , id2
MUL  R2 , R2 , R1
MUL  R2 , R2 , id2
ST   id1 , R2
```

(3b) Explanation & Symbol Table

SI	id	1	float
p	id	2	float
n	id	3	float
r	id	4	float
100	num	100	int

(4) $P \rightarrow a \mid bPcPd \mid bQd$

$Q \rightarrow P \mid QP$

I. Eliminate left factoring & Left Recursion

$P \rightarrow a \mid bP'$

$P' \rightarrow PcPd \mid Qd$

$Q \rightarrow PQ'$

$Q' \rightarrow PQ' \mid \varepsilon$

Substitute Q in P', eliminate left factoring

$P \rightarrow a \mid bP'$

$P' \rightarrow PP'' $

$P'' \rightarrow cPd \mid Q'd$

$Q' \rightarrow PQ' \mid \varepsilon$

II. finding first (X):

	first	first
P	a , b	a , b
P'	first (P)	a , b
P''	c , first (Q')	c , a , b , ε , d
Q'	first (P) , ε	a , b , ε

Find nip follow(x)

$P \rightarrow b P'$

$\text{Foll}(P') = \text{Foll}(P)$

$P' \rightarrow \underset{\alpha}{P} \underset{B}{P''}$

$\text{Foll}(P) = \text{first}(P'') - \varepsilon$

$= c, a, b, d$

$\text{foll}(P) = \text{foll}(P')$

$P' \rightarrow \underset{\alpha}{P} \underset{B}{P''}$

$\text{foll}(P'') = \text{foll}(P')$

$P'' \rightarrow \underset{\alpha}{c} \underset{B}{P} \underset{\beta}{d}$

$\text{foll}(P) = d$

$P'' \rightarrow \underset{\alpha}{Q'} \underset{B}{d}$

$\text{foll}(Q') = d.$

$Q' \rightarrow \underset{\alpha}{P} \underset{B}{Q'}$

$\text{foll}(P) = (\text{first}(Q') - \varepsilon)$

$= a, b, \varepsilon - \varepsilon$

$= a, b$

	Follow	follow
P	a, b, c, d foll(P'), $	~~foll(P)~~, $, a, b, c, d
P'	foll(P)	a, b, c, d, $
P''	foll(P')	a, b, c, d, $
Q'	d	d

III. parser table

State \ i/p symbol	a	b	c	d	$
P	$P \to aP'$	$P \to bP'$			
P'	$P' \to PP''$	$P \to PP''$			
P''	$P'' \to Q'd$	$P'' \to Q'd$	$P'' \to cPd$	$P'' \to Q'd$	
Q'	$Q' \to PQ'$	$Q' \to PQ'$		$Q' \to \varepsilon$	

IV Stack implementation

Stack	i/p	Action
$P	bbacadabadd$	$P \to bP'$
$bP'	bacadabadd$	pop P
$P'	acadabadd$	$P' \to PP''$
$P''P	acadabadd$	$P \to bP'$
$P''P'b	acadabadd$	pop P
$P''P'	cadabadd$	$P' \to PP''$
$P''P''P	cadabadd$	$P \to a$
$P''P''a	adabadd$	pop P
$P''P''	dabadd$	$P'' \to cPd$
$P''dPc	dabadd$	pop P
$P''dP	abadd$	$P \to a$

$\$ p'' d \not\alpha$	$\not\beta d a b a d d \$$	$p \circ p$
$\$ p'' \not\alpha$	$\not\alpha a b a d d \$$	$p \circ p$
$\$ p''$	$a b a d d \$$	$p'' \to Q' q$
$\$ d Q'$	$a b a d d \$$	$Q' \to p Q'$
$\$ d Q' p$	$a b a d d \$$	$p \to a$
$\$ d Q' \not\alpha$	$\not\alpha \not\beta a d d \$$	$p \circ p$
$\$ d Q'$	$b a d d \$$	$Q' \to p Q'$
$\$ d Q' p$	$b a d d \$$	$p \to b p'$
$\$ d Q' p' \not\beta$	$\not\beta a d d \$$	$p \circ p$
$\$ d Q' p'$	$a d d \$$	$p' \to p p''$
$\$ d Q' p'' p$	$a d d \$$	$p \to a$
$\$ d Q' p'' \not\alpha$	$\not\alpha d d \$$	$p \circ p$
$\$ d Q' p''$	$d d \$$	$p'' \to Q' d$
$\$ d Q' d Q'$	$d d \$$	$Q' \to \varepsilon$
$\$ d Q' \not\alpha$	$\not\beta d \$$	$p \circ p$
$\$ d Q'$	$d \$$	$Q' \to \varepsilon$
$\$ A$	$\not\alpha \$$	$p \circ p$
$\$$	$\$$	accept

(5) (12)

a) Recursive procedure - Alg.

b)

$S \rightarrow A\ B$

$A \rightarrow aA / bB / Ac / Ab$

$B \rightarrow b / aB$

$w = a\ cad\ b$

i) Need to identify the recursive descent parse tree using backtracking.

ii) Need to specify, different levels and justify where he couldn't derive the string

$w = a\ cad\ b$.