Programme: B E – Computer Science and Engineering (AI&ML) & Computer Science and Engineering (Cyber Security)

Internal Assessment – I

TERM : 4th OCT 2023 to 27th JAN 2024	COURSE NAME: AUTOMATA THEORY AND COMPILER DESIGN
DATE : 01-12-2023 TIME: 11.00am-12.00pm	COURSE CODE : CY53/CI53
MAX MARKS: 30	PORTIONS : 2.5 UNITS

Mobile Phones are banned

Instructions to Candidates: **Answer any two full questions.** Marks: **15x2=30**

Q. NO	**Questions**	**Blooms Levels (L1 to L6)***	**CO**	**Marks**
1.a	For the expression grammar **E→E*F | F+E | F** **F→F – | id** Obtain the following i. Leftmost derivation for the input string ‘**id-+id*id**’ ii. Rightmost derivation for the input string ‘**id-+id*id**’ iii. Is this Grammar ambiguous? If so, Eliminate the ambiguity.	**L4**	**CO3**	**1+1+2=4**
b	Consider the grammar given below. If not suitable for Predictive parser make necessary changes and construct predictive parsing table, M for the grammar given below. If LL(1), do parsing for the input “**idid$**” **S→FR** **R→S | ε** **F→id**	**L3**	**CO2**	**5**
c	Compute FIRST and FOLLOW of all Non-terminals for the grammar G. Show the traces of computation. **i.** **S→ABC** **A→BS | b** **B→ bS | CA | ε** **C→ ε** **ii.** **S→ACB | cbB |Ba** **A→da |BC** **B→g | ϵ** **C→h | ϵ**	**L3**	**CO4**	**6**
2.a	i. Construct a transition diagram for **her, she, he, him** ii. Specify the input and output of each phase of a C compiler by translating the given assignment statement. (Assume all variables are of type int) **val=sqrt(a*a+b*b);**	**L2, L3**	**CO1, CO4**	**2+3=5**

B	Demonstrate how tokens are identified using "sentinels" technique. Identify the tokens and its appropriate lexemes for the code fragment. **int fact(int n){** **// computing factorial** **if(n==1)** **printf("the factorial is 1");** **if(n>1)** **return(n*fact(n-1));** **}**	**L2**	**CO2**	**2+2 =4**
C	i. What is an Augmented Grammar? Specify the reasons for the necessity of augmenting a grammar for LR Parsers. ii. Identify the initial item in the item set for LR parsers. iii. Construct LR(0) set of items for the following grammar **G:- S→ +SS | *SS | (S) | a**	**L4**	**CO2**	**2+1 +3= 6**
3.a	i. Describe the actions of a Shift Reduce Parser. ii. Show the moves made by shift reduce parser for the input string "**bab$".** Identify handles for the given input string. **G: S → AaBb | BaAb** **A → b** **B → ε** iii. If A→α is a production of a CFG and |α|=n. Then how many items can be generated using **A→ α.**	**L3 ,L4**	**CO2**	**1+3 +1= 5**
B	Design a Predictive Parsing Table for the Grammar given below. Modify the Grammar if required. **G: S→ fES | wES | id** **E→ E:T | E,T | ε** **T→num**	**L3**	**CO2**	**6**
c	Write regular expressions for i. Accepting the Subject Code for semester 3,4,5,6 ii. Accepting Email id for gmail/yahoo	**L4**	**CO1 , CO3**	**2+2 =4**

*** L1** – Remember, **L2** – Understand, **L3**- Apply, **L4**- Analyze, **L5**-Evaluate, **L6**-Create

Course Outcomes (COs):

1. Understand the core concepts in automata theory and Theory of Computation. (PO 1, 2, PSO 1,2)
2. Design and develop lexical analyzers, parsers and code generators. (PO1,2, 3, 9, PSO 1,2)
3. Design Grammars and Automata (recognizers) for different language classes and become knowledgeable about restricted models of Computation (Regular, Context Free) and their relative powers. (PO 1, 2,3,4, PSO 1,3)
4. Gain the knowledge of the structure of a Compiler and Apply concepts automata theory and Theory of Computation to design Compilers. (PO 1,2,3,4,9, PSO 1, 2)

Sl.No.	SCHEME AND SOLITIONS	MARKS
1 a.	**E→E*F | F+E | F**	
	F→F – | id	
	Obtain the following	
	i. Leftmost derivation for the input string ‘**id-+id*id**’	
	E→F+E→F-+E→id-+E→id-+E*F→id-+F*F→id-+id*F→id-+id*id	1M
	ii. Rightmost derivation for the input string ‘**id-+id*id**’	
	E→F+E→F+E*F→F+E*id→F+F*id→F+id*id→F-+id*id→id-+id*id	1M
	iii. Is this Grammar ambiguous? If so, Eliminate the ambiguity.	
	Yes.	
	E→FT|F+ET	2M
	T→*FT| ε	
	F→idR	
	R→-R| ε	
b.	**S→FR**	
	R→S | ε	
	F→id	
	OR	
	ELR	
	S→FR	
	R→FR | ε	
	F→id	
	The grammar is suitable for PP. ---	1M

NT	FIRST	FOLLOW
S	{id}	{$}
F	{id}	{id,$}
R	{id, ε}	{$}

PARSE TABLE M (2M)

NT/INPUT SYMBOL	id	$
S	S→FR	
F	F→id	
R	R→S	R→ ε

Parsing on input: idid (2M)

Matched string	Stack	Input	Action
	S$	idid$	
	FR$	idid$	Reduce S→FR
	idR$	idid$	Reduce F→id
id	R$	id$	Matched id
	S$	id$	Reduce R→S
	FR$	id$	Reduce S→FR
	idR$	id$	Reduce F→id
idid	R$	$	Reduce R→ ε
idid	$	$	Accept

Sl.No.	SCHEME AND SOLITIONS	MARKS
c	**Compute FIRST and FOLOOW**	
	i. S→ABC	
	A→BS | b	
	B→ bS | CA | ε	
	C→ ε	

NT	FIRST	FOLLOW
S	{b}	{$,b}
A	{b}	{b,$}
B	{b, ε }	{b,$}
C	{ε}	{b,$}

1.5+1.5=3M

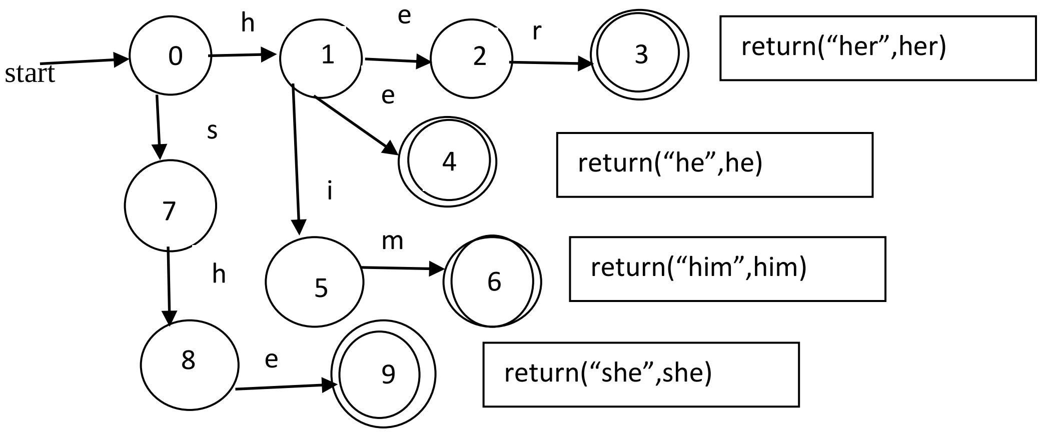
ii. **S→ACB | cbB |Ba**
A→da |BC
B→g | ϵ
C→h | ϵ

NT	FIRST	FOLLOW
S	{a,c,g,d,h, ε }	{$}
A	{d,g,h, ε }	{h,g,$}
B	{g, ε}	{a,h,g,$}
C	{h, ε}	{h,g,$}

1.5+1.5=3M

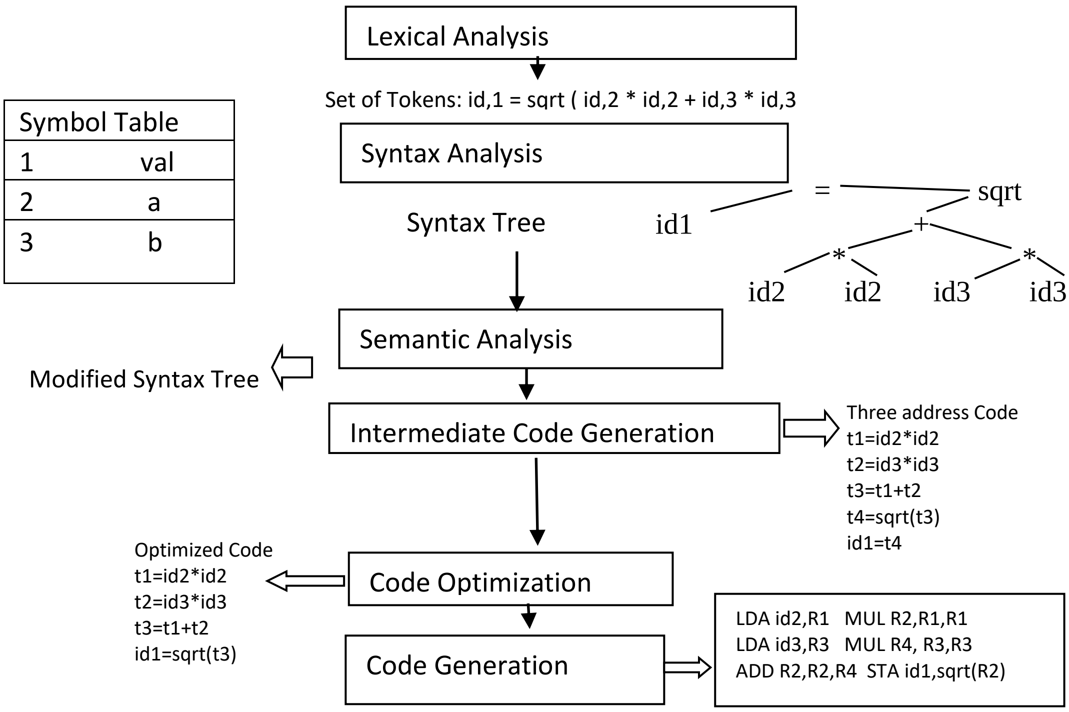
2.a. i. Construct a transition diagram for **her, she, he, him**

2M



ii.

val=sqrt(a*a+b*b);



3M
Each phase other than code optimization 1M

Symbol Table 1M

b.

Sentinels: //eof char should be the sentinel char. Lexeme Begin and Forward pointer-------------- 1M

		a	+	eof

1M

b	+	c	eof		eof

Lexemes	**Tokens**
Chars i,n,t	int
Chars f,a,c,t	id,1
Char (	(
Char n	id,2
Char)	)
Char {	{
Char i,f	if
Char ==	==
1	1
Char p,r,i,n,t,f	printf
“the factorial is 1”	literal
Char >	>
Char r,e,t,u,r,n	return
Char *	*
Char -	-

2M

c.

i. What is an Augmented Grammar? Specify the reasons for the necessity of augmenting a grammar for LR Parsers.

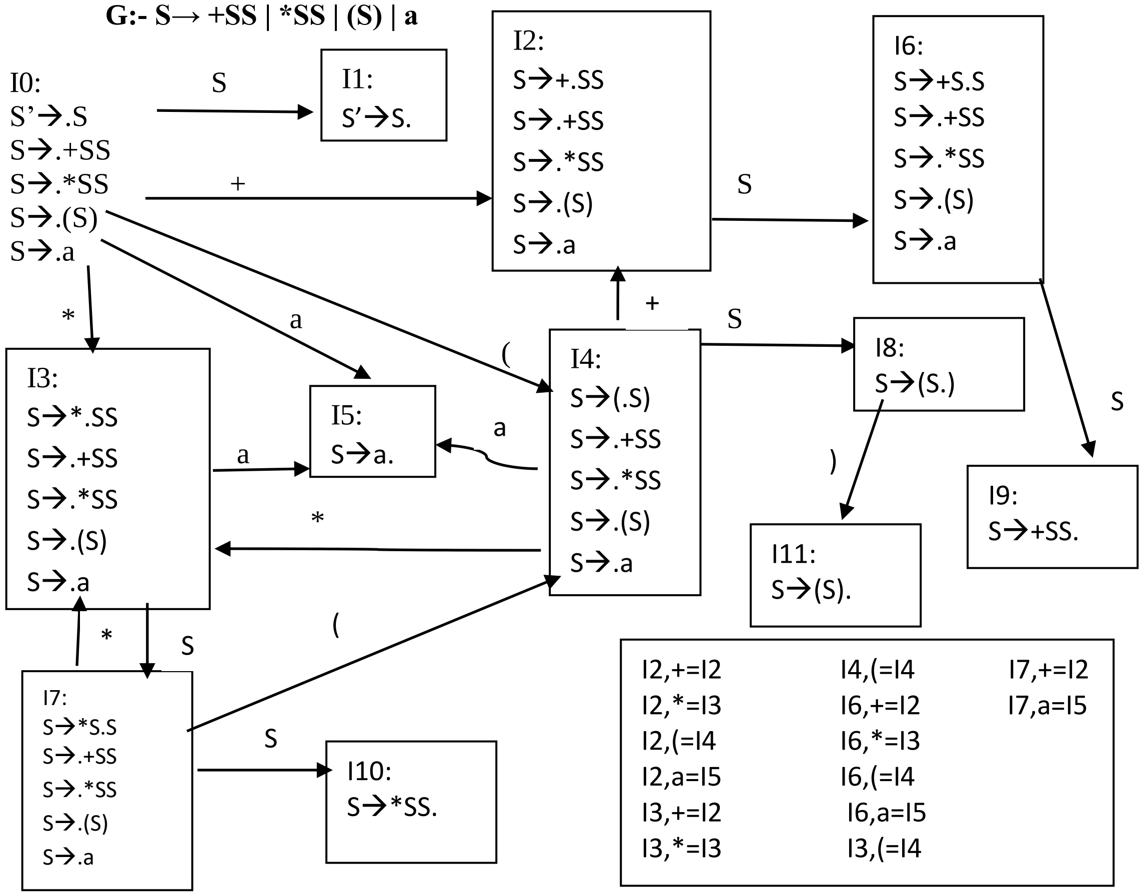
For Bottom Up Parsing, LR Parsers will stop parsing only after getting start symbol as handle (on top of stack). For specifying start symbol as handle 2M

ii. Identify the initial item in the item set for LR parsers.

Initial Item: S’→.S 1M

iii. Construct LR(0) set of items for the following grammar

G:- S→ +SS | *SS | (S) | a 3M



3 a.

i. Describe the actions of a Shift Reduce Parser.
Actions of a Shift reduce parser: Shift, Reduce, Accept, Error — 1M

ii. Show the moves made by shift reduce parser for the input string “**bab$”.** Identify handles for the given input string.

G: S → AaBb | BaAb
A → b
B → ε

Stack	**Input**	**Handle**	**Action**
$	**bab$**		
$b	**ab$**	**b**	**shift**
$A	**ab$**		**Reduce A→b**
$Aa	**b$**		**shift**
$AaB	**b$**	**ε**	**Reduce B→ ε**
$AaBb	**$**		**shift**
$S	**$**	**AaBb**	**Reduce S → AaBb**
			Accept

3M

iii. If A→α is a production of a CFG and |α|=n. Then how many items can be generated using **A→ α.** **Answer:** n+1 — 1M

b.

Design a Predictive Parsing Table for the Grammar given below. Modify the Grammar if required.

G: S→ fES | wES | id
E→ E:T | E,T | ε
T→num

Solution: **Modification required:**
Elimination of left recursion
E→E’
E’→:TE’ | ,TE’ | ε — 1M

New G:
S→ fES | wES | id
E→E’
E’→:TE’ | ,TE’ | ε
T→num

Follow(E)={f,w,id}
Follow(E’)={f,w,id} — 1M

Parse Table, M

NT\input symbol	**id**	**num**	**f**	**w**	**:**	**,**
S	S→id		S→fES	S→wES		
E	E→E’		E→E’	E→E’	E→E’	E→E’
E’	E’→ ε		E’→ ε	E’→ ε	E’→:TE’	E’→,TE’
T		T→num				
The Grammar is LL(1)						

4M

c.

Write regular expressions for
i. Accepting the Subject Code for semester 3,4,5,6
ii. Accepting Email id for gmail/yahoo

Answer:
i. (CI|ci)(CY|cy)/((L|l)?[3-6][1-8] — 2M
ii. [a-zA-z0-9_]@(yahoo)|(gmail)\.(com|”co.in”) — 2M