SRM Institute of Science and Technology
College of Engineering and Technology
SCHOOL OF COMPUTING

SRM Nagar, Kattankulathur – 603203, Chengalpattu District, Tamilnadu

SET C

Academic Year: 2022-23 (EVEN)

Test: CLAT-2 | **Date: 04.04.2022**
Course Code & Title: 18CSC304J COMPILER DESIGN | **Duration: 2 periods**
Year & Sem: III & V | **Max. Marks: 50**

Q. No	Question
1	Grammar of the programming is checked at which phase of compiler? A. Lexical analysis B. Syntax analysis C. Semantic analysis D. Syntax directed translation **ANS :B**
2	Which of the following regular expression operator has highest precedence A. Concatenation B. Union C. Positive closure D. Kleene closure **ANS :D**
3	An LALR(1) parser for a grammar G can have shift-reduce (S-R) conflicts if and only if A. the SLR(1) parser for G has S-R conflicts B. the LR(1) parser for G has S-R conflicts C. the LR(0) parser for G has S-R conflicts D. the LALR(1) parser for G has reduce-reduce conflicts **ANS :B**
4	The grammar $C \rightarrow CC \mid (C) \mid e$ is not suitable for predictive-parsing because the grammar is A. Ambiguous B. Left recursive C. Right recursive D. An operator grammar **ANS :B**
5	Consider the grammar $E \rightarrow E + n \mid E \times n \mid n$ For a sentence $n + n \times n$, the handles in the right-sentential form of the reduction are A. n, $E + n$ and $E + n \times n$ B. n, $E + n$ and $E + E \times n$ C. n, $n + n$ and $n + n \times n$ D. n, $E + n$ and $E \times n$ **ANS :D**
6	For the grammar rules given below what is the FIRST(S) S ->Aa|bB, A->c|€ A. b,c B. a,c

	C. a,b,c D. a,b,c,€ **ANS :C**
7	Consider the grammar defined by the following production rules, with two operators ∗ and + S --> T * P T --> U | T * U P --> Q + P | Q Q --> Id U --> Id Which one of the following is TRUE? A. +' is left associative, while '∗' is right associative B. +' is right associative, while '∗' is left associative C. Both + and ∗ are right associative D. Both + and ∗ are left associative **ANS :B**
8	The grammar A → AA | (A) | ε is not suitable for predictive-parsing because the grammar is A. ambiguous B. left-recursive C. right-recursive D. both (A) and (B) **ANS :D**
9	Consider the following grammar: S → FR R → S | ε F → id In the predictive parser table, M, of the grammar the entries M[S, id] and M[R, $] respectively. A. {S → FR} and {R → ε }

	B. {S → FR} and { }
	
	C. {S → FR} and {R → *S}
	
	D. {F → id} and {R → ε}
	
	ANS :A

10	Consider the grammar S → aAbB | bAaB | ε A → S B → S What is Follow(S) A. {a,b,$} B. {a,$} C. {b,$} D. {a} **ANS :A**

SRM Institute of Science and Technology
College of Engineering and Technology
SCHOOL OF COMPUTING
RM Nagar, Kattankulathur – 603203, Chengalpattu District, Tamilnadu
Academic Year: 2022-23 (EVEN)

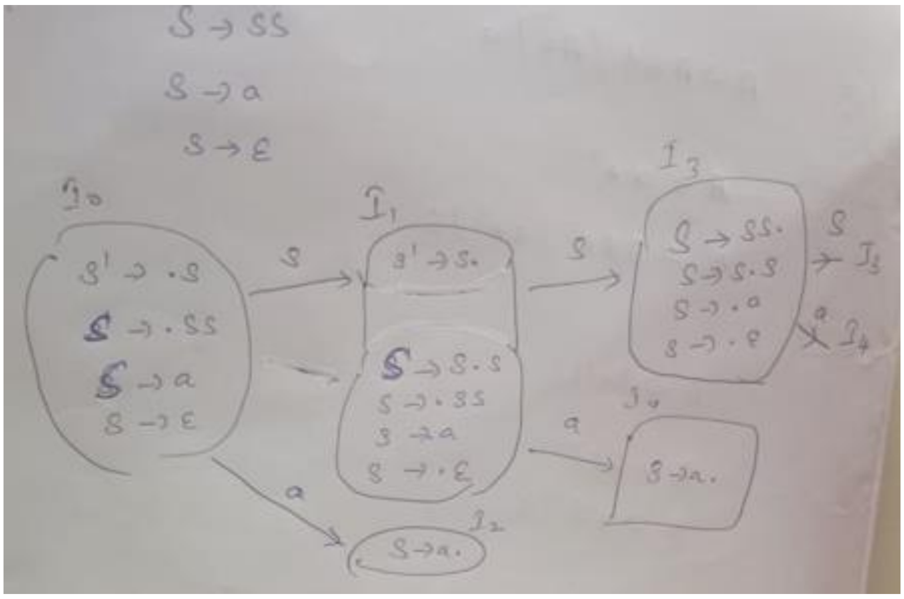
SET C

Test: CLAT-2 | **Date: 04.04.2022**
Course Code & Title: 18CSC304J COMPILER DESIGN | **Duration: 2 periods**
Year & Sem: III & V | **Max. Marks: 50**

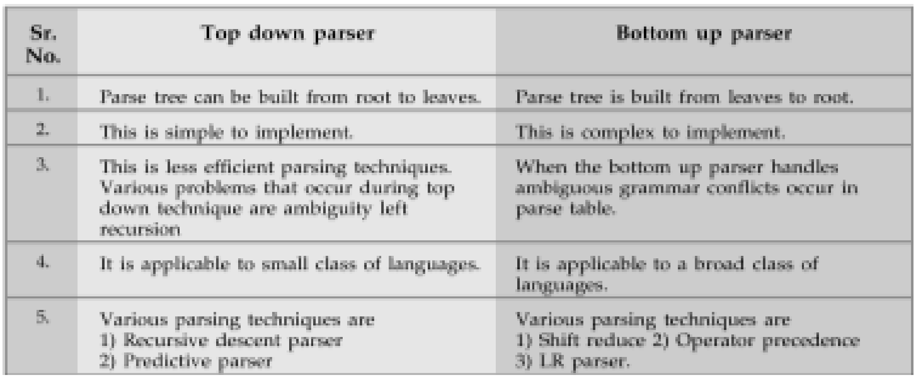
S.No.	Course Outcome	PO1	PO2	PO3	PO4	PO5	PO6	PO7
1	CO3	3	3	3				

11 **Construct the LR(0) item sets for the following grammar**

S->SS, S->a S-> ∈



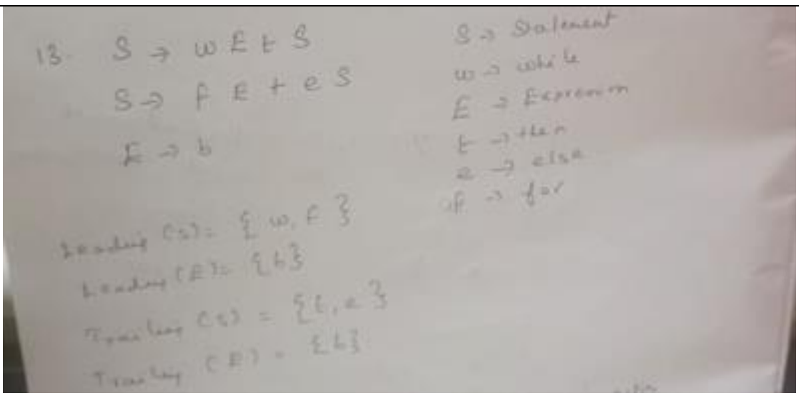
12 **Differentiate between Top down parsing and Bottom up parsing**



Sr. No.	Top down parser	Bottom up parser
1.	Parse tree can be built from root to leaves.	Parse tree is built from leaves to root.
2.	This is simple to implement.	This is complex to implement.
3.	This is less efficient parsing techniques. Various problems that occur during top down technique are ambiguity left recursion	When the bottom up parser handles ambiguous grammar conflicts occur in parse table.
4.	It is applicable to small class of languages.	It is applicable to a broad class of languages.
5.	Various parsing techniques are 1) Recursive descent parser 2) Predictive parser	Various parsing techniques are 1) Shift reduce 2) Operator precedence 3) LR parser.

13 Ragu has to find the CFG for the below pseudo code and find leading and trailing for all the non-terminals from the following grammar

```
Statement
{
While expression then statement OR
for expression then else statement
}
expression = b
```



13. S → wEtS
S → fEteS
E → b

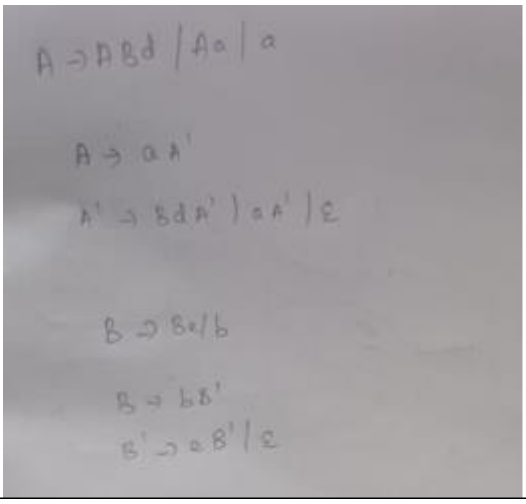
S → Statement
w → while
E → Expression
t → then
e → else
f → for

Leading (S) = {w, f}
Leading (E) = {b}
Trailing (S) = {t, e}
Trailing (E) = {b}

14 Parse the input string “ibtaea”using shift reduce parsing for the following grammar
S->iEtS/iEtSeS/a
E->b

STACK	INPUT	ACTION
$	ibtaea $	SHIFT
$i	btaea $	SHIFT
$ib	taea $	Reduce E->b
$iE	taea $	SHIFT
$iEt	aea $	SHIFT
$iEta	ea$	Reduce S->a
$iEtS	ea$	SHIFT
$iEtSe	a$	SHIFT
$iEtSea	$	Reduce S->a
$iEtSeS	$	S->iEtSeS
$S	$	success

15 Eliminate Left Recursion for the following grammar
A->ABd/Aa/a
B->Be/b



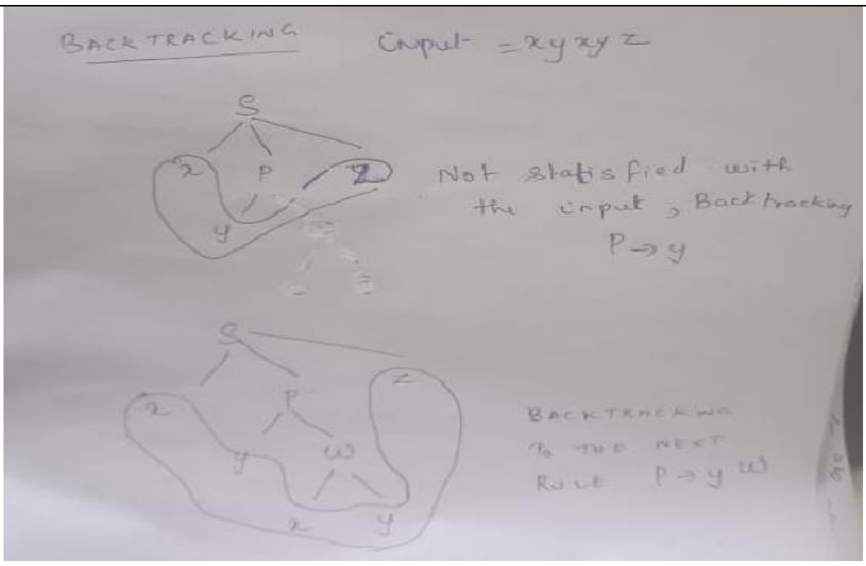
A → ABd | Aa | a

A → aA'
A' → BdA' | aA' | ε

B → Be/b

B → bB'
B' → eB' | ε

16 a) Perform the Backtracking for the input xyxyz for the following grammar (4 marks)
S->xPz
P->yw/y
W-> xy


BACKTRACKING
Input = xyxyz
S
x
P
z
y
Not satisfied with the input, Backtracking P → y
S
x
P
z
y
w
x
y
BACKTRACKING TO THE NEXT RULE P → y w

b) What are the problems in Top Down Parsing (4marks)

1) Backtracking

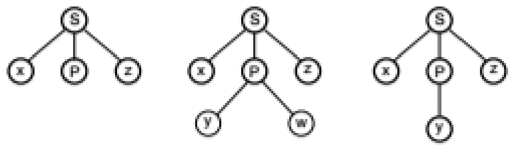
Backtracking is a technique in which for expansion of non-terminal symbol we choose one alternative and if some mismatch occurs then we try another alternative if any.

For example :

$S \rightarrow xPz$

$P \rightarrow yw \mid y$

Then



2) Left recursion

The left recursive grammar is a grammar which is as given below.

$A \overset{+}{\Rightarrow} A\alpha$

$$\left.\begin{array}{l} A \rightarrow A\alpha \\ A \rightarrow \beta \end{array}\right\}$$

Then we eliminate left recursion

$$\left.\begin{array}{l} A \rightarrow \beta A' \\ A' \rightarrow \alpha A' \\ A' \rightarrow \varepsilon \end{array}\right\}$$

3) Left factoring

If the grammar is left factored then it becomes suitable for the use. Basically left factoring is used when it is not clear that which of the two alternatives is used to expand the non-terminal. By left factoring we may be able to re-write the production in which the decision can be deferred until enough of the input is seen to make the right choice.

In general if

$A \rightarrow \alpha\beta_1 | \alpha\beta_2$

is a production then it is not possible for us to take a decision whether to choose first rule or second. In such a situation the above grammar can be left factored as

$A \rightarrow \alpha A'$

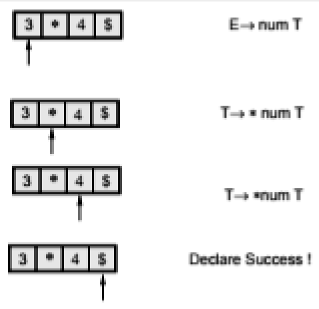
$A' \rightarrow \beta_1 | \beta_2$

c) Write the procedure of Recursive Descent Parsing (procedure 3 marks and input checking 3 marks) for the following grammar

E->num T

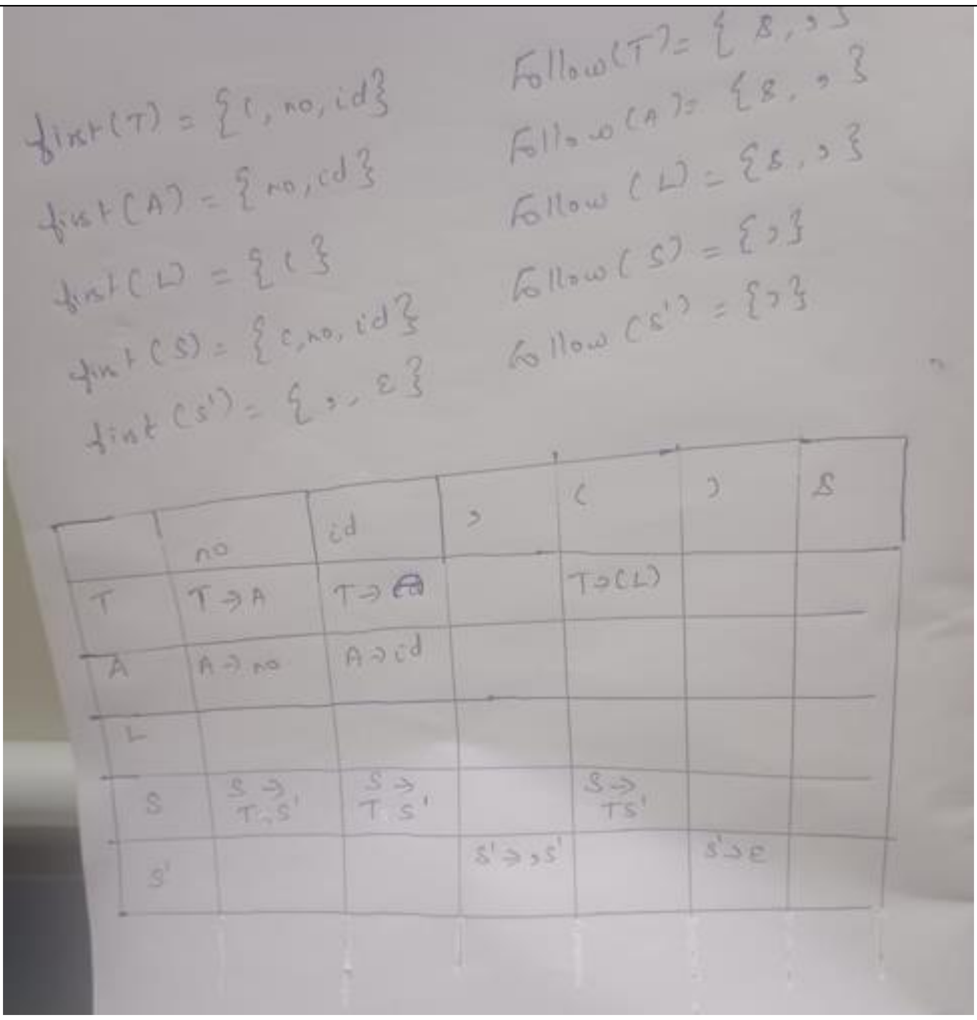
T->*num T/ ∈

```
procedure E
{
        if lookahead = num then
        {
                match(num) ;
                T;                              /* call to procedure T */
        }
        else
                error;
        if  lookahead = $
        {
                declare success;    /* Return on success */
        }
        else
                error;
}/*end of procedure E*/
procedure T
{
        if lookahead = '*'
        {
                match('*');
                if lookahead ='num'
              {
                     match(num);
                     T;
              }                         /* inner if closed*/
              else
                     error;
       }                                /* outer if closed*/
       else NULL;                       /*indicates ε Here the other alternate is
                                         combined into same procedure T */
}                                       /* end of procedure T*/
procedure match( token t )
{
       if lookahead=t
       lookahead = next  token;               /*lookahead pointer is advanced*/
       else
       error;
}                                             /*end of procedure match*/
procedure error
{
       print("Error!!!");
}                                             /*end of procedure error*/
```



17 Consider the Context free grammar
T->A/L
A-> no/id
L -> (S)
S->T,S/S

i) Find the issue in the above grammar and eliminate (4 marks)
ii) Construct the First and Follow for the Non-terminals (4marks)
iii) Construct the LL(1) parsing table (4 marks)



first(T) = {(, no, id}
first(A) = {no, id}
first(L) = {(}
first(S) = {(, no, id}
first(S') = {,, ε}

Follow(T) = {$, ,}
Follow(A) = {$, ,}
Follow(L) = {$, ,}
Follow(S) = {)}
Follow(S') = {)}

	no	id	,	(	)	$
T	T→A	T→A		T→(L)		
A	A→no	A→id				
L						
S	S→T,S'	S→T S'		S→TS'		
S'			S'→,S'		S'→ε	

18

i)Justify the below grammar is an Operator Precedence Grammar with the rules ((2 marks)

E->EAE
E->(E)
E->id
A->*
A-> / (NOTE: / represents the divide symbol)

ANSWER :

E-> E*E
E-> E/E
E-> (E)

ii)Write down the rules for Leading , Trailing and operator precedence table (5 marks)

ANSWER :

Leading(E)->{*,/, (}

Trailing(E)->{*,/, (}

iii) Construct the operator precedence table for the operators using Leading and Trailing (4 marks)

ANSWER:

E-> E*E

Trailing (E) > *

*** < leading (E)**

E-> E/E

Trailing (E) > /

/ < leading (E)

E-> (E)

(=)

Trailing (E) >)

(< leading (E)

iv) Draw the operator precedence graph using operator precedence functions and find the longest path (4 marks)

19

Show that the following grammar is LR(1) not LALR
S -> aEa | bEb | aFb | bFa
E -> e
F -> e

Item Sets Construction (5 marks) Table of LR(1) (3 marks) Table of LALR(3) and Justification (1mark)

I_0
S' -> .S [$]
S -> .aEa [$]
S -> .aFb [$]
S -> .bFa [$]
S -> .bEb [$]

Goto (I_0 ,S)
I_1
S' -> S. [$]

Goto (I_0 ,a)

I_2
S -> a.Ea [$]
S -> a.Fb [$]
E -> .e   [a]
F -> .e   [b]

I_3

Goto (I_0 ,b)
S -> b.Eb [$]
S -> b.Fa [$]
E -> .e [b]
F -> .e [a]

I_4

Goto (I_2 ,E)
S -> aE.a [$]

I_5

Goto (I_2 ,F)

S -> aF.b [$]

I_6

Goto (I_2 ,e)
E -> e. [a]
F -> e. [b]

I_7

Goto (I_3 ,E)
S -> bE.b [$]

I_8

Goto (I_3,F)

S -> bF.a [$]

I_9

Goto (I_3 ,e)
E -> e. [b]
F -> e. [a]

I_{10}

Goto (I_4 ,a)
S -> aEa. [$]

I_{11}

Goto (I_5 ,b)

S -> aFb. [$]

I_{12}

Goto (I_7 ,b)
S -> bEb. [$]

I_{13}

Goto (I_8,a)

S -> bFa. [$]

1. S -> aEa
2. S-> bEb
3. S-> aFb
4. S-> bFa
5. E -> e
6. F -> e

LR(1) TABLE

STATE	a	b	e	$	S	E	F
0	s2	s3			1	-	-
1				Accept			
2			S6			4	5
3			S9			7	8
4	S10						
5		S11					
6	r5	r6					
7		S12					
8	s13						
9	r6	r5					
10				r1			
11				r3			
12				r2			
13				R4			

LALR TABLE

STATE	a	b	e	$	S	E	F
0	s2	s3			1	-	-
1				Accept			
2			S6			4	5
3			S9			7	8
4	S10						
5		S11					
69	r5/r6	r6/r5					
7		S12					
8	s13						
10				r1			
11				r3			
12				r2			
13				R4			

SINCE REDUCE REDUCE CONFLICT IS IN 69 IT IS NOT LALR