Affiliated to GGSIP University, New Delhi
Approved by AICTE & Council of Architecture

Delhi Technical Campus Greater Noida

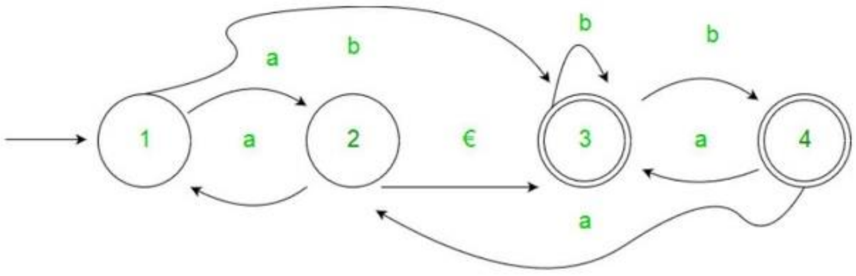
Question Bank Unit 1&2

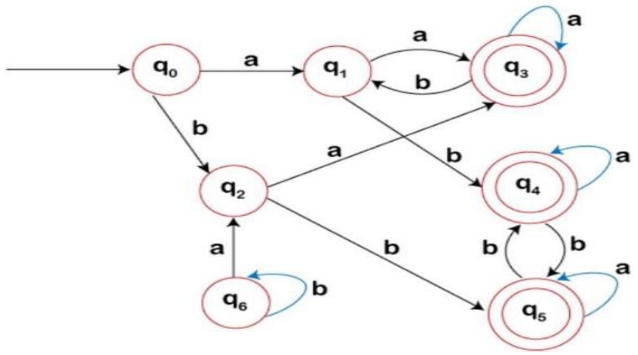
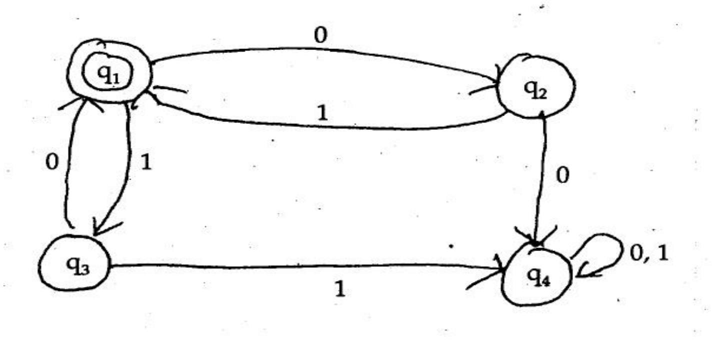
Subject: COMPILER DESIGN **Subject Code: CIC-303**

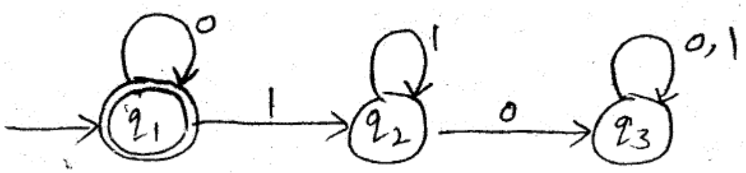
Class: B.Tech CSE 5th **Faculty Name: Dr. Seema Verma/ Sweta Rai**

Date of Issue: **Date of Submission:**

1.	Define term: a) Cross compiler b) Dirty Compiler c) Lexeme d) Token e) Context free Grammar f) Pattern
2.	Discuss the advantages and disadvantages for single and multipass compiler.
3.	Explain various phases of compiler with diagram. List various compiler writing tools.
4.	Differentiate between pass and phase in compiler design.
5.	What is the need of look ahead pointer in Lexical analyzer.
6.	Explain buffer management in lexical analyzer.
7.	Write tools for lexical analyzer.
8.	Differentiate between Top down and bottom-up parser
9.	Write the rules for finding First and Follow in a given grammar with example.
10.	What is ambiguous CFG? Explain with Example. How it can be removed.
11.	What is operator grammar? Give example. What is operator precedence parser?
12.	Consider the following grammar: S-> a| ^ | (T) T -> T,S |S In the above grammar, find leftmost and rightmost derivation for a) (a, (a, a)) b) (((a, a), ^, (a)), a)

13.	Write the step by step (showing input and output of each phase) compiler translation of the statement: - X: =Y*Z+10. Take another example also if required.
14.	Specify the function of each phase of the compiler using the statement **Amount=principal(1+rate*time)**
15.	Draw a deterministic finite automaton (DFA) that recognizes the language of all strings over the alphabet {0, 1} that start and end with the same symbol.
16.	Draw a deterministic finite automate which either starts with 01 or end with 01 of a string containing 0, 1
17.	Draw a DFA containing even no of 0’s and even no of 1’s of string containing 0,1
18.	Draw a DFA which accept a string containing “ing” at the end of a string in a string of {a-z}
19.	Translate the regular expression (0+1)01(0+1) into a nondeterministic finite automaton (NFA). Present the NFA using a state transition diagram.
20.	A. Write the regular expression for- a) Set of string of a’s and b’s of any length including the null string. So L={€,a,,b,aa,ab,bb,ba,aaa…………} b) Set of strings of a’s and b’s ending with the string abb. So L={abb,aabb,babb,aaabb,ababb,…………………} B. Design a FA from the given regular expression 10+(0+11)0*1.
21.	Convert the following into regular expression: 
22.	Construct the NFA for the following regular expression: R=(a|b)*abb
23.	Construct a minimal state DFA for the following regular expression. a) (a|b)*|(ab)*b|a*(bb)*

<table>
<tr><td></td><td>b) (0+1)*(0+1)10
c) (a|b)*a(a|b)(a|b)(a|b)</td></tr>
<tr><td>24.</td><td>Minimize the following DFA:

</td></tr>
<tr><td>25.</td><td>Eliminate left recursion from the given following Grammar:
a) S -> Aa|b, A->Ac|Sd| ε
b) E-> E+T |T, T-> T*F|F, F->(E)|id</td></tr>
<tr><td>26.</td><td>Explain and remove the ambiguity from the following CFG.
E-> E+E
E -> E-E
E -> E/E
E -> E*E
E -> (E)
E-> -E
E -> id</td></tr>
<tr><td>27.</td><td>What is symbol Table? Explain in detail about its contents and data structure.</td></tr>
<tr><td>28.</td><td>Find the regular expression correspondence to

</td></tr>
<tr><td>29.</td><td>Write a CFG for the regular expression r = 0* 1 (0+1)*</td></tr>
<tr><td>30.</td><td>Consider the Grammar
S->aSbS | bSaS | ε</td></tr>
</table>

	Show that this grammar is ambiguous by constructing two different leftmost derivations for the sentence abab.
31.	Construct a DFA with reduced states equivalent to the regular expression 10+(00+1)0*10
32.	Design the regular expression for given transition diagram: 
33.	What is Regular Expression? Write the regular expression for: a. R=R1+R2 (Union operation) b. R=R1.R2 (concatenation Operation) c. R=R1* (Kleen Clouser) d. R=R+ (Positive Clouser) e. Write a regular expression for a language containing strings which end with “abb” over Σ= {a,b}. f. Construct a regular expression for the language containing all strings having any number of a’s and b’s except the null string
34.	a) Explain the process of bootstrapping in compiler design with example. b) What is left recursion and left factoring? Explain with example. c) Difference between top down and bottom up parsers with example. d) Explain shift reduce parsing with the help of an example. e) Explain buffer management in lexical analyzer. f) Write tools for lexical analyzer. g) Differentiate between Top down and bottom-up parser h) What is ambiguous CFG? Explain with Example. How it can be removed. i) Explain Top down parser and the associated problems. j) What is left recursion? How it can be removed? k) What is backtracking problem in Top-down parser? How it can be removed? l) What is recursive descent parser? Give example m) Write the rules for finding First and Follow in a given grammar. n) Write the rules to design predictive parsing table. o) What is LL (1) Parser?

p) What is shift reduce parser? Explain with example.

q) Define handle and handle pruning.

r) What is operator grammar? Give example.

s) Explain Leading and trailing. What is their significance?

t) What is operator precedence parser?

u) Write the production rules to eliminate the left recursion and left factoring problems.

v) Write Rules to construct FIRST Function and FOLLOW Function.

35. For the grammar given below Construct the LL(1) parsing table.

E->TE’
E’->+TE’|€
T->FT’
T’->*FT’|€
F->(E) | ID

36. Check whether the following grammar is LL(1) or not.

i) S->A|a, A->a

ii) S->aSA|€, A->c|€

37. a) What do you mean by Handle? Check whether the grammar E->E+T|T, T->a is LR(0) or not.

38. Consider the following Grammar:

a) S->Aa |b
   A->Ac |Sd |e

b) A-> ABd|Aa|a
   B-> Be|b

Remove left recursion.

39. Do left factoring in the following grammar:

A-> aAB|aA|a
B-> bB|b

40. Design the predictive Parsing Table for the following grammars and check whether the given grammar is LL(1) or not:

a) S→ACB|CbB
   A→da|BC

	B→g|Є C→h|Є b) S→AaAb|BbBa A→ Є B→Є c) S →1AB| Є A →1AC| 0C B→0S C→1 d) #=end marker S →S# S →qABC A →a|bbD B →a| Є C →b| Є, D →c| Є e) S -> i C t S E | a E -> e S | € C -> b
41.	Design LR(0) and LR(1) parsing table for the following: a) S→Aa|bAc|Bc|bBa A->d B->d b) S→A A→AB| Є A→aB| b c) S→xAy|xBy|xAz A→aS|q B→q
42.	Show the following Grammar: S-> AaAb|BbBa A-> € B-> € Is LL(1) and parse the input string “ba”.

43.	Write short note on Lex and YACC?
44.	Draw the parse Tree to generate the given string. Based on the tree design the shift-reduce the operator precedence table. **E→E+T/T** **w=id+id *id** **T→T*F/F** **F→(E)/id**
45.	What is LALR parser? Construct the set of LR(1) items for this grammar: S-> CC C-> aC C->d
46.	Show the following grammar **S->Aa|bAc|Bc|bBa** **A->d** **B->d** Is LR(1) but not LALR(1).
47.	Write the comparison among SLR Parser, LALR parser and Canonical LR Parser.
48.	Compute First, follow each non-terminal and draw a predictive parsing table. **E→E+T/T** **T→T*F/F** **F→(E)/id**
49.	Construct an LR(1) Parsing table for the given context-free grammar. **S→AA** **A→aA/b**
50.	Consider following grammar E→E+T/T T→T*F/F F→(E)/id Construct SLR parsing table

ALL THE BEST FOR EXAMINATION