Reg. No. : | | | | | | | | | | | | |

Question Paper Code :R3D01

B.E./B.Tech. DEGREE EXAMINATION, NOV 2024

Third Semester

Computer Science And Business Systems

R21UCB301-FORMAL LANGUAGE AND AUTOMATA THEORY

(Regulations R2021)

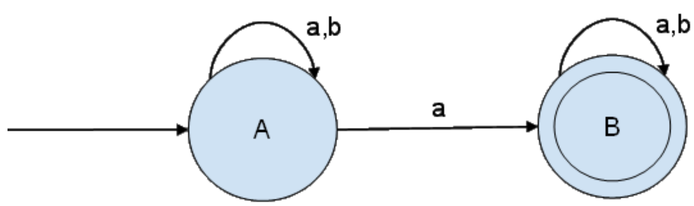
Duration: Three hours   Maximum: 100 Marks

PART A - (10 x 2 = 20 Marks)

1. Define: Finite Automaton (FA) — CO1-U
2. Construct NFA to accept all strings contains $\{a,b\}$ which starting with "*abb*". — CO2-App
3. Write Regular expression for the language — CO2-App

   (i)All strings beginning with *'11'* and ending with *'00'*

   (ii)Set of all strings that has substring *'00'*
4. Convert the following Finite Automata to Regular Grammar — CO2-App



5. List out the applications of Context Free Grammar. — CO1-U
6. Remove ε production from the given context free grammar — CO2-App

   $S \to XYX, \quad X \to 0X|\varepsilon, \quad y \to 1Y|\varepsilon$
7. Check whether $L = \{a^n b^n \mid n >= 1\}$ is CFL or not. — CO2-App
8. Define the Instantaneous Description of PDA. — CO1-U
9. Define Diagonalization language (L_d). — CO1-U
10. Design a Turing machine to compute *n mod 2* where n is represented in the tape in unary form consisting of only *0's*. — CO2-App

PART – B (5 x 16= 80 Marks)

11. (a) Construct DFA equivalent to the NFA M=({p, q, r, s },{0,1},δ, p,{ q ,s}),where T(transition function)is defined as follows, CO2-App (16)

State/input symbol	0	1
→p	{q ,s}	{q}
*q	{r}	{q, r}
r	{s}	{p}
*s	-	{p}

Or

(b) Construction from NFA to DFA using Subset construction methods. CO2-App (16)

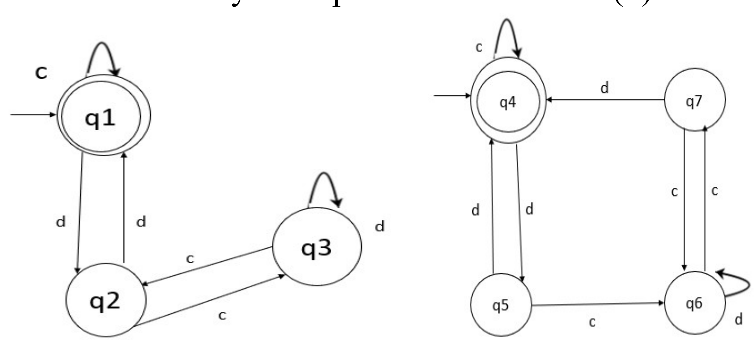
	a	b
P	{p,q}	□p}
Q	{r}	{r}
R	{s}	-
*s	{s}	{s}

12. (a) (i) Conversion from Regular expression to finite automata RE=(a|b)*abb. (8) CO2-App (16)

(ii) Conversion from Regular expression to finite automata RE=(0+1)*(00+11). (8)

Or

(b) (i) Consider the two Deterministic Finite Automata (DFA) and check whether they are equivalent or not. (8) CO2-App (16)



(ii) Construct an NFA to accept the language indicated by the following regular expression *(0+1)*(00+11)*. (8)

13. (a) Convert the following CFG to Greibach Normal Form CO2-App (16)

S->CA|BB

B->b|SB

C->b

A->a

Or

(b) Convert the following CFG to GNF CO2-App (16)

S→ASB|aB

A→B|S

B→b|ε

14. (a) Construct a context-free grammar G which accepts L(M), where M = ({q0,q1}, {a,b}, {a,z0}, δ, q0, z0, ϕ) and where δ is given by CO2-App (16)

a.δ(q0, a, z0) = { (q0,az0)}

b.δ(q0, a,a) = { (q0, aa)}

c.δ(q0, b,a) = { (q1, ε)}

d.δ(q1, b, a) = { (q1, ε)}

e.δ(q1, ε,z0) = { (q1, ε)}

Or

(b) Construct a context-free grammar G which accepts L(M), where M = ({q0,q1}, {a,b}, {z0,z1}, δ, q0, z0, ϕ) and where δ is given by CO2-App (16)

a) δ(q0, b, z0) = { (q0,z1z0)}
b) δ(q0, ε, z0) = { (q0, ε)}
c) δ(q0, b, z1) = { (q0, z1z1)}
d) δ(q0, a, z1) = { (q1, z1)}
e) δ(q1, b, z1) = { (q1, ε)}
f) δ(q1, a, z0) = { (q0, z0)}

15. (a) Construct Turing Machine for the Language L={a^n b^n}where n>=1. CO2-App (16)

Or

(b) Construct a TM for the language L = {$0^n 1^n 2^n$} where n≥1 CO2-App (16)