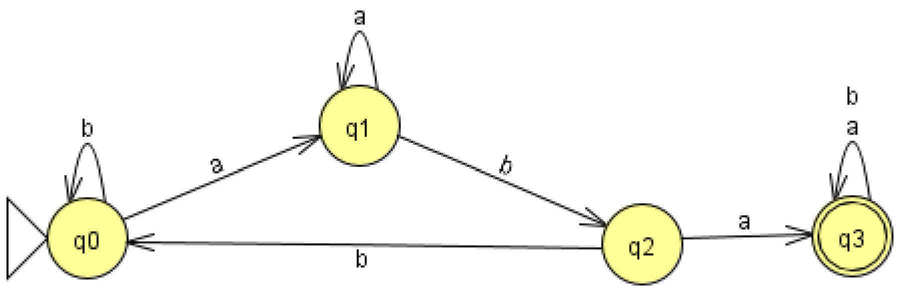
## VTU QP Solution Dec-Jan 2025

| Sub: | Theory of Computation | Sub Code: | BCS503 | Branch: | CSE |
|---|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| 1<br>(a) | Define i)Language ii) String iii)powers of alphabet<br>i) A Set of all strings which are chosen over some Σ* where Σ is a particular alphabet.<br>If L⊆ Σ* , then L is a language over Σ<br>Eg. 1. Set of all strings consisting of n 0's followed by n 1s, n ≥ 0<br>{ε,01,0011,000111,…<br>2. Set of binary numbers whose value is prime<br>{01,10,11,101,111,…}<br>ii) A string is a finite sequence of of symbols chosen from an alphabet.<br>Eg. w=00100 from Σ={0,1}<br>An empty string is denoted by ε<br>Length of a string is the number of positions for symbols in a string<br>Eg. \|w\| = 5<br>iii) Powers of alphabet<br>If Σ is an alphabet, we can express the set of all strings of a certain length from that alphabet as $\Sigma^k$<br>$\Sigma^0 = \{\varepsilon\}$<br>Σ={0,1}, then $\Sigma^1 = \{0,1\}$, $\Sigma^2 = \{00,01,10,11\}$, $\Sigma^3 = \{000,001,010,011,100,101,110,111\}$<br>$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \dots$ | **3M** | **CO 1** | **L1** |
| (b) | Define DFA, Draw DFA to accept<br>Definition<br>A deterministic finite automaton consists of:<br>1. A finite set of states, often denoted Q.<br>2. A finite set of input symbols, often denoted Σ.<br>3. A transition function that takes as arguments a state and an input symbol and returns a state.<br>transition function is represented by δ.<br>transition function, δ - arcs between states and the labels on the graph representation, δ - arcs.<br>If q is a state, a is an input symbol, δ(q,a) is a that state p such that there is an arc labeled a from q to p.<br>4. A start state, one of the states in Q.<br>5. A set of final or accepting states, F. F is a subset of Q<br>input symbols<br>transition function     F ⊆ Q.<br>$A = (Q, \Sigma, \delta, q_0, F)$ set of accepting states. | **10** | **CO 1** | L3 |

i) The set of all strings that contain aba



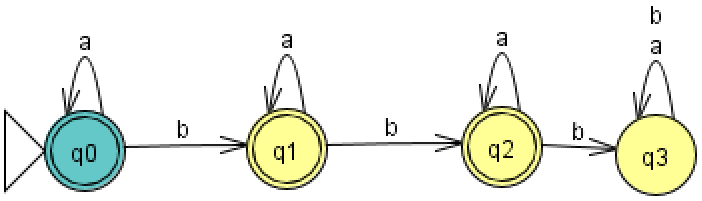
The definition of the resulting DFA is

DFA P = (Q,Σ, δ, $q_0$,F)

= ({q0,q1,q2,q3,},{a,b}, **$\delta_D$,** {q3})

| | $\delta_D$ | a | b |
|---|---|---|---|
| → | q0 | q1 | q0 |
| | q1 | q1 | q2 |
| | q2 | q3 | q0 |
| | *q3 | q3 | q3 |

ii) To accept the strings of a's and b's that contain no more than 2 b's



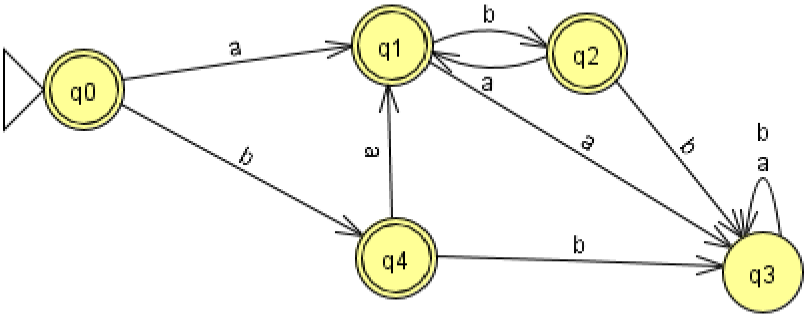
The definition of the resulting DFA is

DFA P = (Q,Σ, δ, $q_0$,F)

= ({q0,q1,q2,q3,},{a,b}, **$\delta_D$,** {q0,q1,q2})

| | $\delta_D$ | a | b |
|---|---|---|---|
| → | q0 | q0 | q1 |
| | q1 | q1 | q2 |
| | q2 | q2 | q3 |
| | *q3 | q3 | q3 |

iii) L={w ∈{a,b}*} No two consecutive characters are same in w.



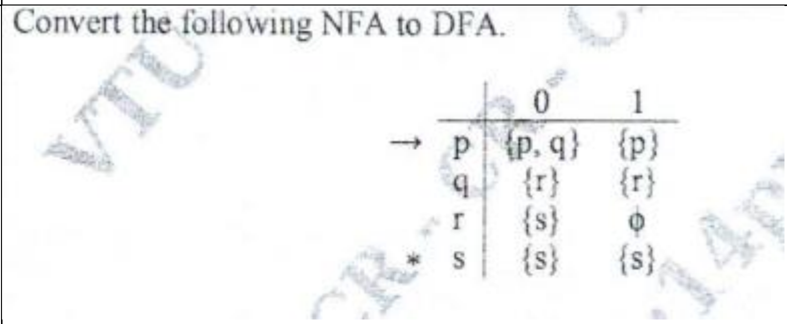
The definition of the resulting DFA is

DFA P = (Q,Σ, δ, $q_0$,F)

= ({q0,q1,q2,q3,q4},{a,b}, **$\delta_D$,** {q0,q1,q2,q4})

| | $\delta_D$ | a | b |
|---|---|---|---|
| → | *q0 | q1 | q4 |
| | *q1 | q3 | q2 |
| | *q2 | q1 | q3 |
| | q3 | q3 | q3 |
| | *q4 | q1 | q3 |

c)



Convert the following NFA to DFA.

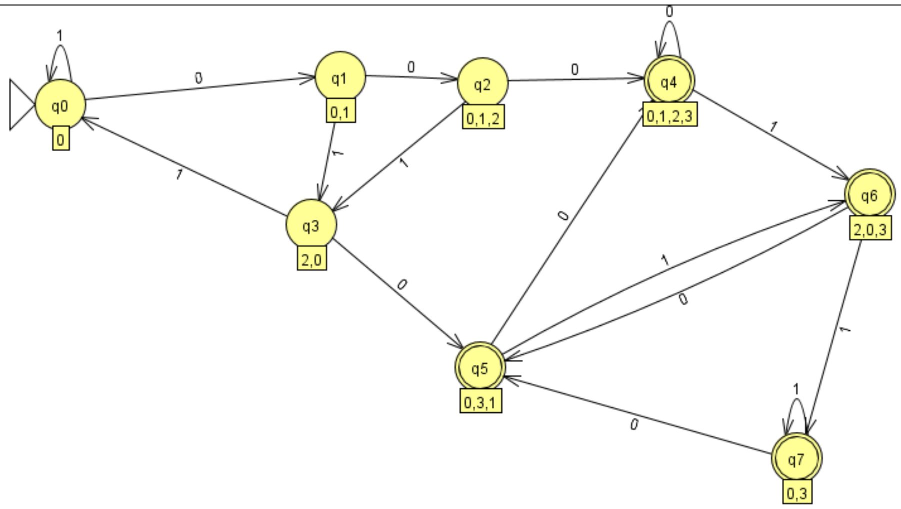
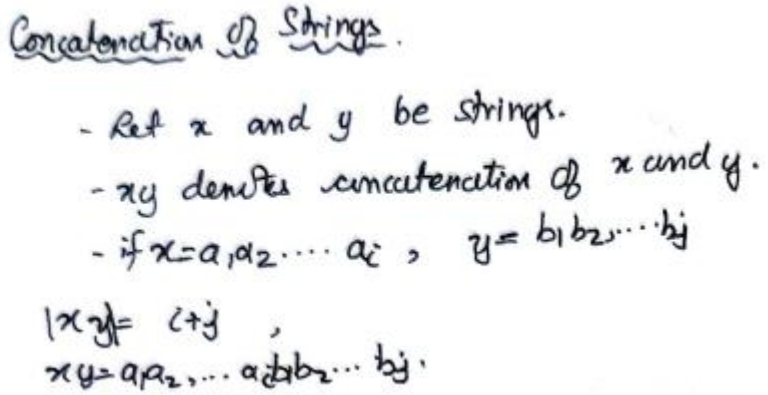
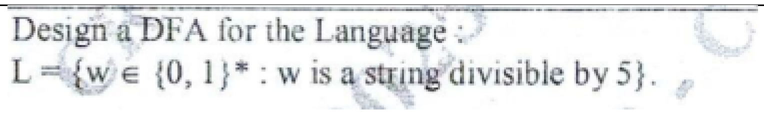
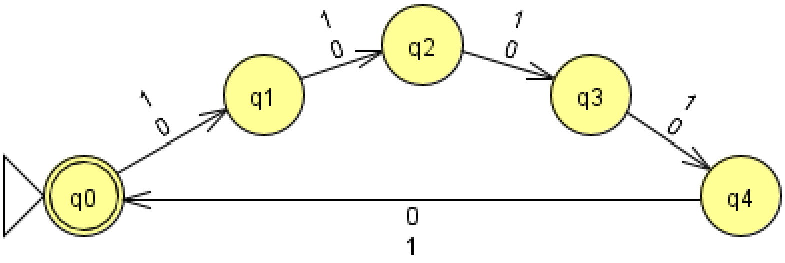
| | | 0 | 1 |
|---|---|---|---|
| → | p | {p, q} | {p} |
| | q | {r} | {r} |
| | r | {s} | φ |
| * | s | {s} | {s} |

| | | 0 | 1 |
|---|---|---|---|
| → | p | {p,q} | {p} |
| | {p,q} | {p,q,r} | {p,r} |
| | {p,q,r} | {p,q,r,s} | {p,r} |
| | {p,q,r,s} | {p,q,r,s} | {p,r,s} |
| | {p,r} | {p,q,s} | {p} |
| * | {p,q,s} | {p,q,r,s} | {p,r,s} |
| * | {p,r,s} | {p,q,s} | {p,s} |
| * | {p,s} | {p,q,s} | {p,s} |

**7**

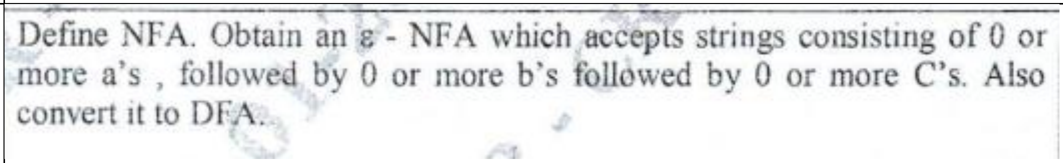
**CO 1**

L3

| | | | | |
|---|---|---|---|---|
| |  q0<br>0<br>q1<br>0,1<br>q2<br>0,1,2<br>q3<br>2,0<br>q4<br>0,1,2,3<br>q5<br>0,3,1<br>q6<br>2,0,3<br>q7<br>0,3<br>0<br>1 | | | |
| 2 (a) | Define i)Alphabet ii)Reversal of string iii) Concatenation of languages<br>Alphabet - An alphabet is a finite non-empty set of symbols. Σ denotes an alphabet.<br>1. Σ={0,1} is the binary alphabet<br>2. Σ={a,b,…,z} is the set of lower case letters<br>3. The set of all printable ASCII characters<br>ii)Reversal of strings (denoted by sR) is a string obtained by reversing t he order of symbols in s. For example,$1011^R$=1101<br>Concatenation Of Strings.<br>- Let $x$ and $y$ be strings.<br>- $xy$ denotes concatenation of $x$ and $y$.<br>- if $x=a_1a_2\cdots a_i$ , $y= b_1b_2,\cdots b_j$<br>$\|xy\|= i+j$ ,<br>$xy=a_1a_2,\ldots a_ib_1b_2\cdots b_j$. | 3 | **CO 1** | L1 |
| (b) | Design a DFA for the Language :<br>L = {w ∈ {0, 1}* : w is a string divisible by 5}.<br> <br>The definition of the resulting DFA is<br>DFA P = (Q,Σ, δ, $q_0$,F) | 7 | **CO 1** | L3 |

= ({q0,q1,q2,q3,q4},{0,1}, **$\delta_D$,** {q0})

| $\delta_D$ | 0 | 1 |
|---|---|---|
| *q0 | q1 | q1 |
| q1 | q2 | q2 |
| q2 | q3 | q3 |
| q3 | q4 | q4 |
| q4 | q0 | q0 |

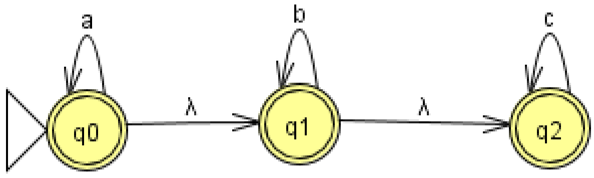


Define NFA. Obtain an ε - NFA which accepts strings consisting of 0 or more a's , followed by 0 or more b's followed by 0 or more C's. Also convert it to DFA.

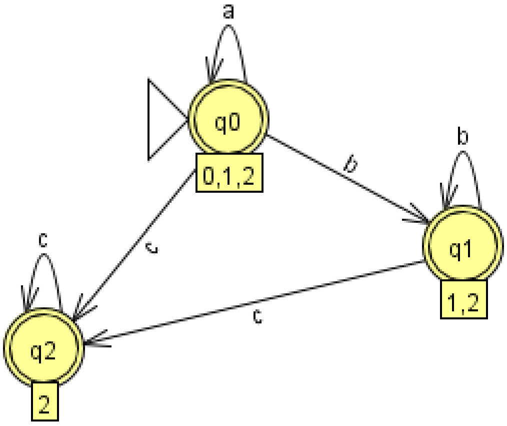
A= (Q,Σ,δ,$q_0$,F)

1. Q is the finite set of states
2. Σ is the finite set of input symbols
3. $q_0 \in Q$ is the start state
4. δ is the transition function that takes a state in Q and an input symbol in Σ and returns a subset of Q. If there is no state, it returns Φ.

(c)



DFA



**10**

**CO 1**

L2

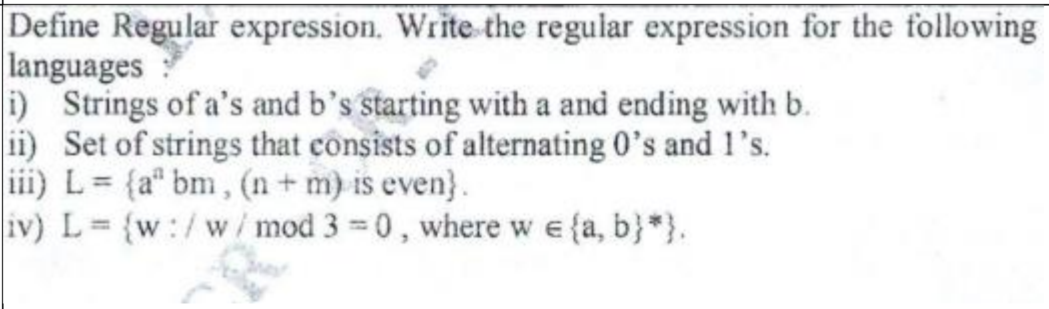
|  | $\delta_D$ | a | b | c |
|---|---|---|---|---|
| → | *{q0,q1,q2} | {q0,q1,q2} | {q1,q2} | {q2} |
|  | *{q2} | Φ | Φ | {q2} |
|  | *{q1,q2} | Φ | {q1,q2} | {q2} |

The definition of the resulting DFA is

DFA P = ({,Σ, δ, $q_0$,F)

= ({ {q0,q1,q2}, {q2},{q1,q2} },{a,b,c}, $\delta_D$, {{q0,q1,q2}, {q2},{q1,q2}} )

3(a)



Define Regular expression. Write the regular expression for the following languages :

i) Strings of a's and b's starting with a and ending with b.

ii) Set of strings that consists of alternating 0's and 1's.

iii) L = {$a^n$ bm , (n + m) is even}.

iv) L = {w : / w / mod 3 = 0 , where w ∈ {a, b}*}.

i) a(a+b)*b

ii) (01)* + (10)*+0+1+ε

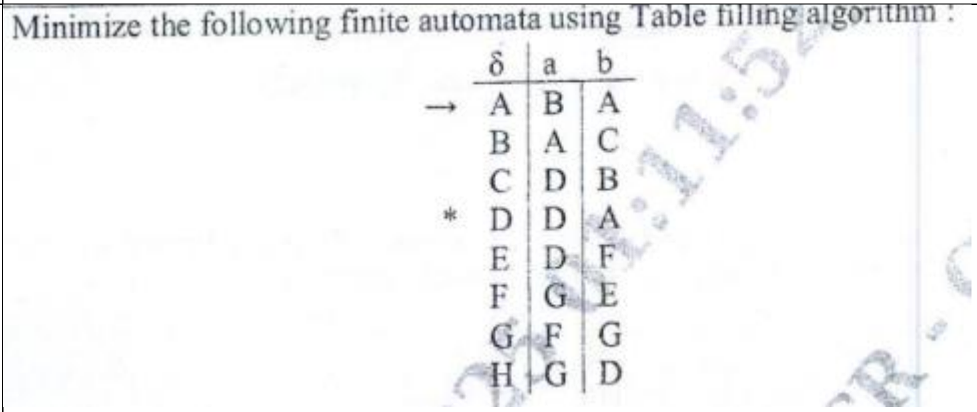
iii) Even+even =even, odd +odd = even

(aa)*(bb)* + a(aa)*b(bb)*

iv) ((a+b)(a+b)(a+b))*

10 CO 2 L2

(b)



Minimize the following finite automata using Table filling algorithm :

|  | δ | a | b |
|---|---|---|---|
| → | A | B | A |
|  | B | A | C |
|  | C | D | B |
| * | D | D | A |
|  | E | D | F |
|  | F | G | E |
|  | G | F | G |
|  | H | G | D |

E,F,G and H are not reachable

|  |  |  |  |
|---|---|---|---|
| B | X |  |  |
| C | X | X |  |
| D | x | x | x |
|  | A | B | C |
|  |  |  |  |
|  | a | b |  |
| **A,B** | A,B | A,C | X |
| **A,C** | B,D | A,B | X |
| **B,C** | A,D |  | X |

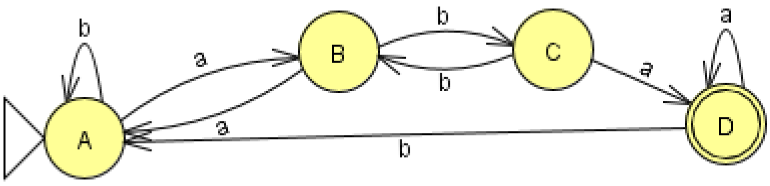
10 CO 2 L2

No further minimization required.

The definition of the resulting DFA is

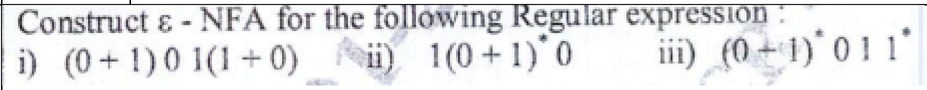
DFA P = (Q,Σ, δ, $q_0$,F)

= ({A,B, C,},{a,b}, $\boldsymbol{\delta_D}$**,** {D})



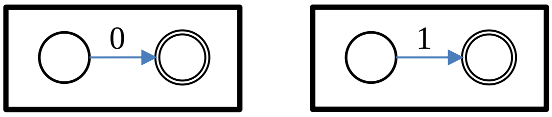
| $\delta_D$ | a | b |
|---|---|---|
| **A** | B | A |
| **B** | A | C |
| **C** | D | B |
| ***D** | D | A |

4 (a)

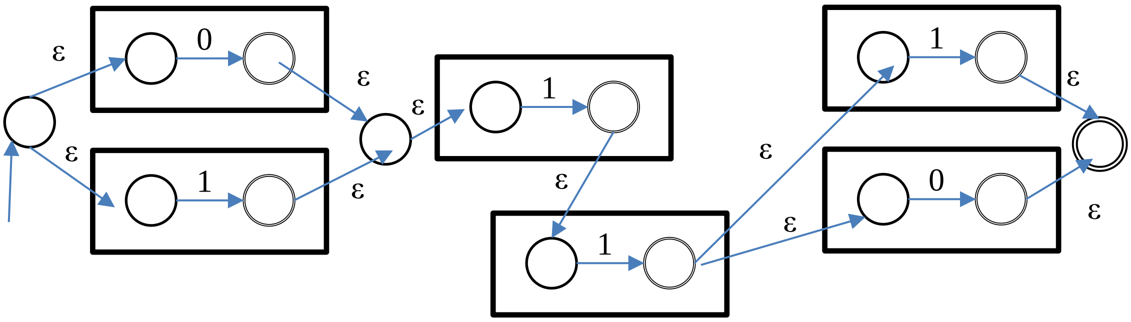

Construct ε - NFA for the following Regular expression :

i) $(0+1)\,0\,1(1+0)$ ii) $1(0+1)^*\,0$ iii) $(0+1)^*\,0\,1\,1^*$

Basis :



i)

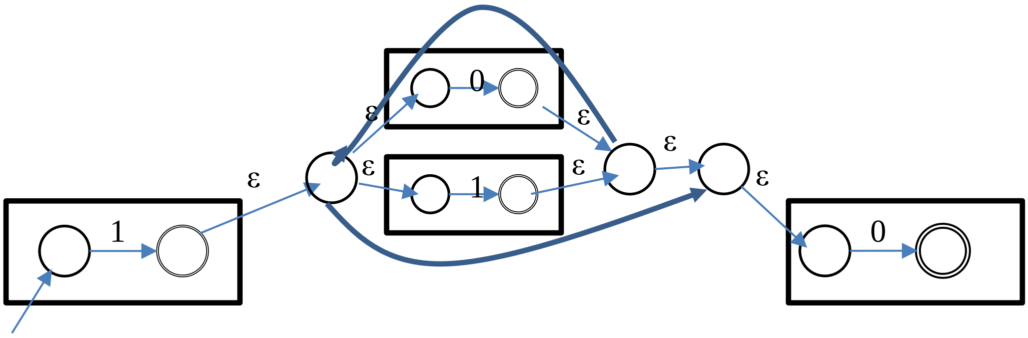
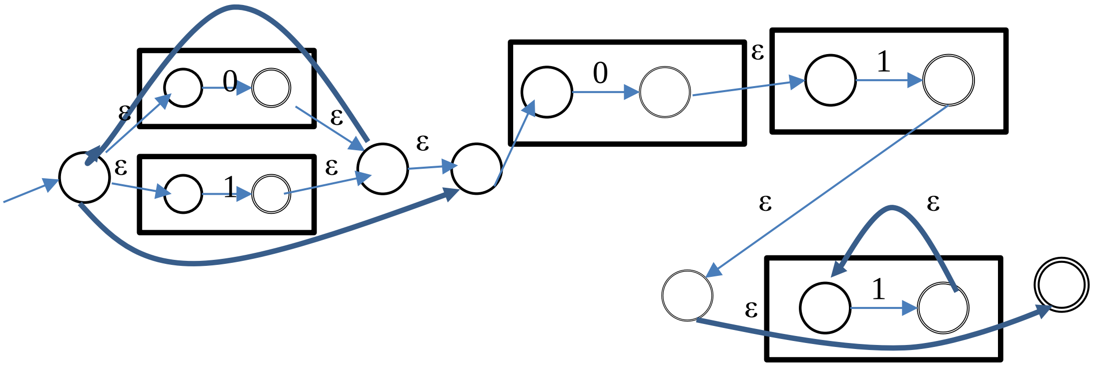
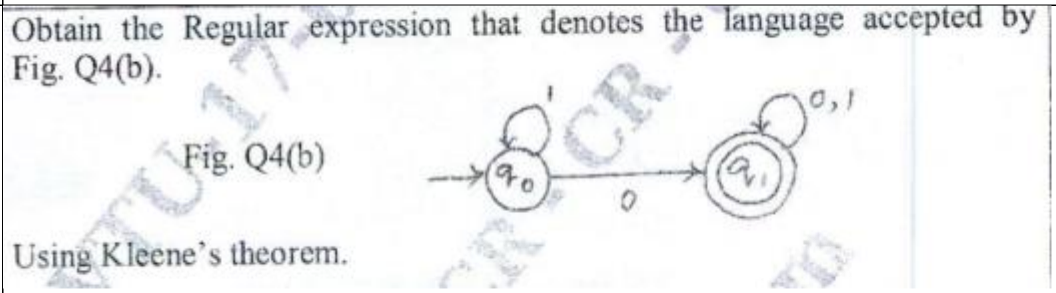


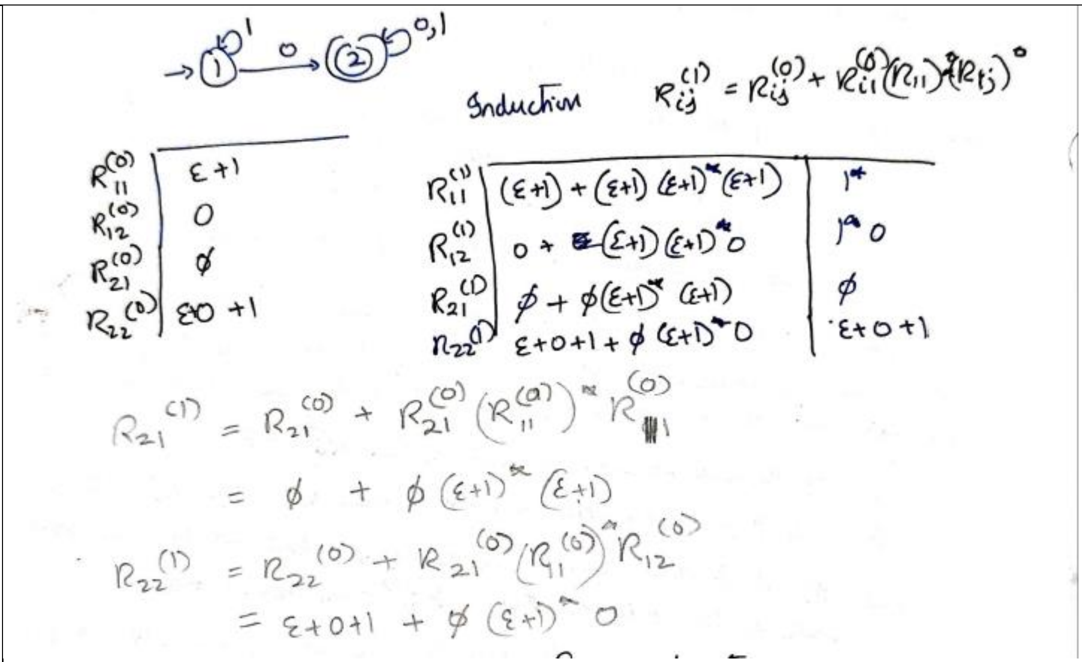
ii)

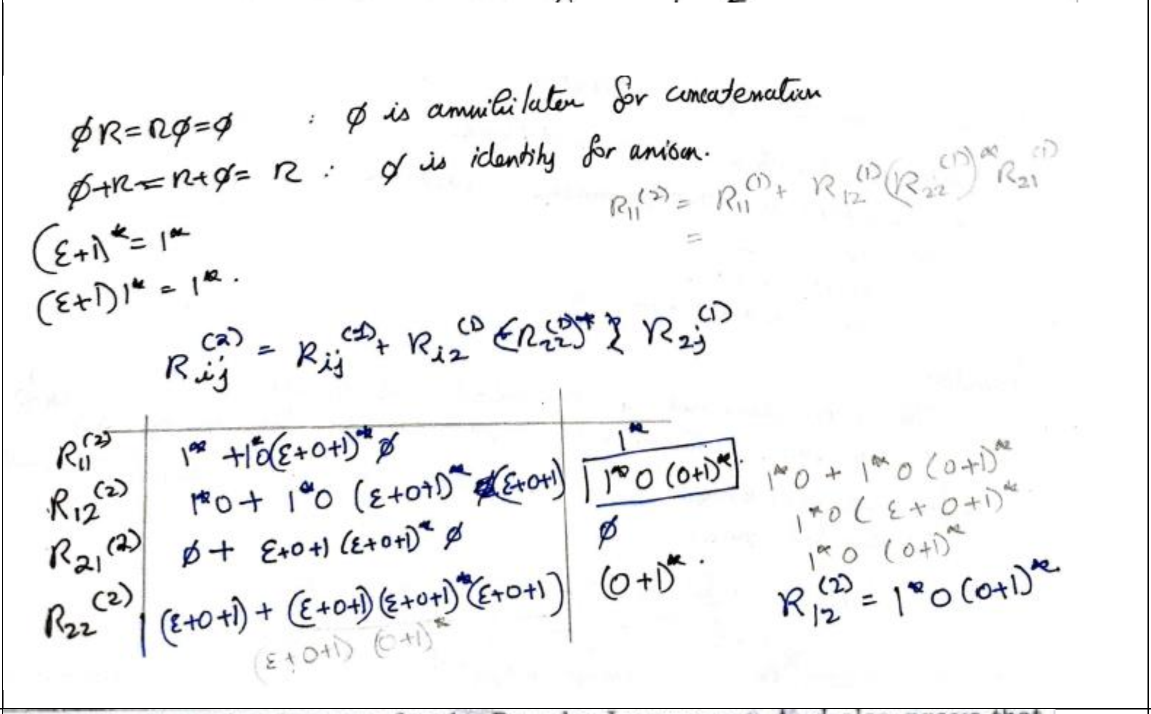
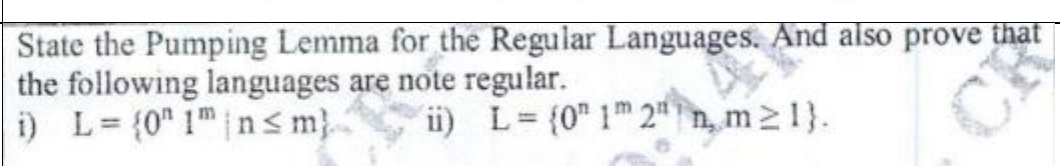
6

CO 2

L1

| | | | | |
|---|---|---|---|---|
| |  <br>iii)<br> ε<br>0<br>ε<br>ε<br>1<br>ε<br>ε<br>0<br>ε<br>1<br>ε<br>ε<br>ε<br>1 | | | |
| (c) | Obtain the Regular expression that denotes the language accepted by Fig. Q4(b).<br>Fig. Q4(b)<br> <br>Using Kleene's theorem. | **8** | **CO 2** | L1 |

<table>
<tr>
<td></td>
<td>


Induction $R_{ij}^{(1)} = R_{ij}^{(0)} + R_{i1}^{(0)}(R_{11})^{*}(R_{1j})^{0}$
<table>
<tr><td>$R_{11}^{(0)}$</td><td>$\varepsilon+1$</td></tr>
<tr><td>$R_{12}^{(0)}$</td><td>$0$</td></tr>
<tr><td>$R_{21}^{(0)}$</td><td>$\phi$</td></tr>
<tr><td>$R_{22}^{(0)}$</td><td>$\varepsilon 0+1$</td></tr>
</table>
<table>
<tr><td>$R_{11}^{(1)}$</td><td>$(\varepsilon+1)+(\varepsilon+1)(\varepsilon+1)^{*}(\varepsilon+1)$</td><td>$1^{*}$</td></tr>
<tr><td>$R_{12}^{(1)}$</td><td>$0+(\varepsilon+1)(\varepsilon+1)^{*}0$</td><td>$1^{*}0$</td></tr>
<tr><td>$R_{21}^{(1)}$</td><td>$\phi+\phi(\varepsilon+1)^{*}(\varepsilon+1)$</td><td>$\phi$</td></tr>
<tr><td>$R_{22}^{(1)}$</td><td>$\varepsilon+0+1+\phi(\varepsilon+1)^{*}0$</td><td>$\varepsilon+0+1$</td></tr>
</table>
$R_{21}^{(1)} = R_{21}^{(0)} + R_{21}^{(0)}(R_{11}^{(0)})^{*}R_{11}^{(0)}$
$= \phi + \phi(\varepsilon+1)^{*}(\varepsilon+1)$
$R_{22}^{(1)} = R_{22}^{(0)} + R_{21}^{(0)}(R_{11}^{(0)})^{*}R_{12}^{(0)}$
$= \varepsilon+0+1 + \phi(\varepsilon+1)^{*}0$

$\phi R = R\phi = \phi$ : $\phi$ is annihilator for concatenation
$\phi + R = R + \phi = R$ : $\phi$ is identity for anion.
$R_{11}^{(2)} = R_{11}^{(1)} + R_{12}^{(1)}(R_{22}^{(1)})^{*}R_{21}^{(1)}$
$=$
$(\varepsilon+1)^{*} = 1^{*}$
$(\varepsilon+1)1^{*} = 1^{*}.$
$R_{ij}^{(2)} = R_{ij}^{(1)} + R_{i2}^{(1)}\{(R_{22}^{(1)})^{*}\}R_{2j}^{(1)}$
<table>
<tr><td>$R_{11}^{(2)}$</td><td>$1^{*}+1^{*}0(\varepsilon+0+1)^{*}\phi$</td><td>$1^{*}$</td></tr>
<tr><td>$R_{12}^{(2)}$</td><td>$1^{*}0+1^{*}0(\varepsilon+0+1)^{*}(\varepsilon+0+1)$</td><td>$\boxed{1^{*}0(0+1)^{*}}$</td></tr>
<tr><td>$R_{21}^{(2)}$</td><td>$\phi+\varepsilon+0+1(\varepsilon+0+1)^{*}\phi$</td><td>$\phi$</td></tr>
<tr><td>$R_{22}^{(2)}$</td><td>$(\varepsilon+0+1)+(\varepsilon+0+1)(\varepsilon+0+1)^{*}(\varepsilon+0+1)$<br>$(\varepsilon+0+1)(0+1)^{*}$</td><td>$(0+1)^{*}.$</td></tr>
</table>
$1^{*}0 + 1^{*}0(0+1)^{*}$
$1^{*}0(\varepsilon+0+1)^{*}$
$1^{*}0(0+1)^{*}$
$R_{12}^{(2)} = 1^{*}0(0+1)^{*}$
</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(c)</td>
<td>

State the Pumping Lemma for the Regular Languages. And also prove that the following languages are note regular.
i) $L = \{0^n 1^m \mid n \le m\}$ ii) $L = \{0^n 1^m 2^n \mid n, m \ge 1\}$.

According to the pigeon hole principle, if there are k states and there is k+1 input, then it must stay in a state multiple number of times.
The pumping lemma can prove this.
Theorem:
Let L be a regular language. Then there exists a constant n (which depends on L ) such that for every string w in L such that $|w| \ge n$, we can break w into 3 strings, w=xyz such that

1. $y \ne \varepsilon$
2. $|xy| \le n$
3. For all $k \ge 0$, the string $xy^kz$ is also in L
</td>
<td><b>8</b></td>
<td><b>CO 2</b></td>
<td>L1</td>
</tr>
</table>

That is, we can always find a non-empty string y not too far from the beginning of w that can be “pumped”, i.e., repeating y any number of times or deleting it keeps the resulting string in the language L.

i) $\{0^n1^m, n<=m\}$

Let’s assume a DFA exists for L with number of states of at most =2m, . Lets take a string of length of string 2m where m=2. Let |w| = 4

Let y=aa

Let |xy| <=2k (4)

| **0** | **011** | **ε** |
|---|---|---|
| **x** | **y** | **z** |

Let us pump y 2 times. The resulting string would be $0\ (011)^2 = 0011011 \notin L$

Hence it is proved that the language $\{0^n1^m, n<=m\}$is not **regular.**

ii) $\{0^n1^m2^n, n,m>=1\}$

Let’s assume a DFA exists for L with number of states of at most =2n+1, . Lets take a string with n=2 with number of states 5. Let |w| = 8

Let y=aa

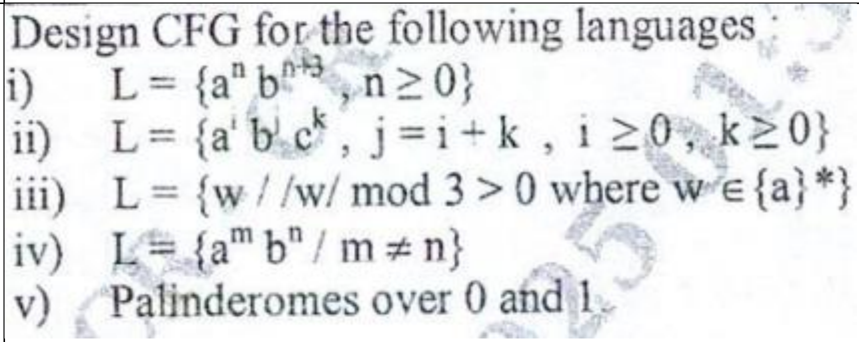
Let |xy| <=2k (4)

| **00** | **011** | **000** |
|---|---|---|
| **x** | **y** | **z** |

Let us pump y 2 times. The resulting string would be $0\ 0(011)^2\ 000= 00011010001 \notin L$

Hence it is proved that the language $\{0^n1^m2^n, n,m>=1\}$is not **regular.**

5(a) | **10** | **L2** | CO 3



Design CFG for the following languages :

i) $L = \{a^n b^{n+3} , n \geq 0\}$

ii) $L = \{a^i b^j c^k , j = i + k , i \geq 0 , k \geq 0\}$

iii) $L = \{w / /w/ \bmod 3 > 0$ where $w \in \{a\}^*\}$

iv) $L = \{a^m b^n / m \neq n\}$

v) Palinderomes over 0 and 1.

i) $L = a^nb^{n+3}$: n>=0

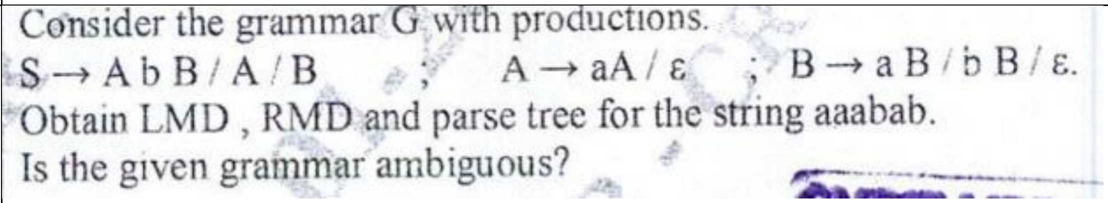
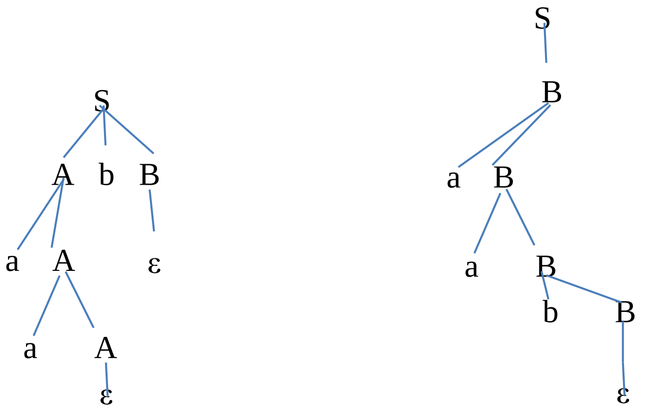
$S \rightarrow aSb \mid bbb$

$G= (V,T,S,P) = (\{S\}.\{a,b\},\{S\rightarrow aSb, S\rightarrow bbb\},S)$

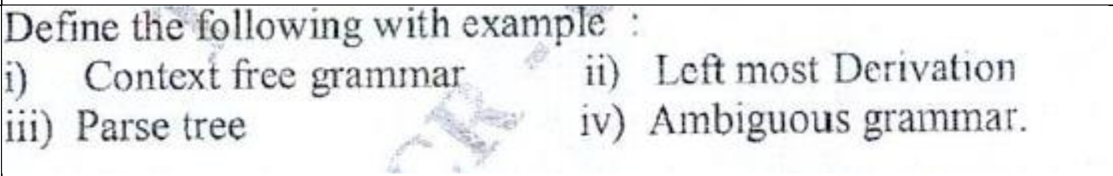
ii) $S\rightarrow aSc \mid A \mid \varepsilon$

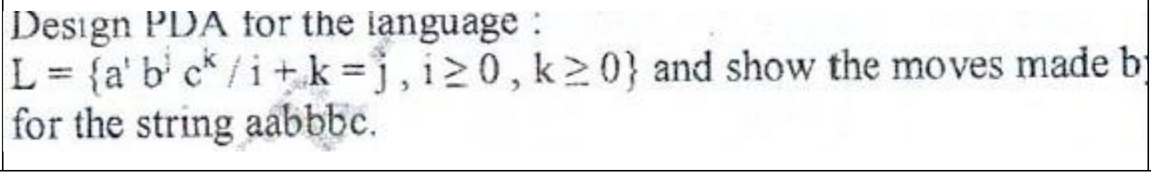
$A \rightarrow bAc \mid \varepsilon$

iii)$S\rightarrow aA$

<table>
<tr><td></td><td>A→aA<br>B→aS | ε<br>iv) S→aSb|A|B<br>A→aA|a<br>B→Bb|b<br>v) S→0S0 | 1S1| 0| 1| ε</td><td></td><td></td><td></td></tr>
<tr><td>(b)</td><td><br>Consider the grammar G with productions.<br>S → A b B / A / B ; A → aA / ε ; B → a B / b B / ε.<br>Obtain LMD , RMD and parse tree for the string aaabab.<br>Is the given grammar ambiguous?<br>LMD<br>S=>AbB => aAbB => aaAbB=> aaaAbB =>aaabB => aaabaB=>aaababB=>aaabab<br>RMD<br>S=> AbB=>AbaB=>AbabB=>Abab=>aAbab=>aaAbab=>aaaAbab=>aaabab<br><br>Is the given grammar ambigous<br>W= aab, there are two distinct parse trees<br>Hence the grammar is ambiguous<br><br>

S<br>A b B<br>a A<br>ε<br>a A<br>ε<br>S<br>B<br>a B<br>a B<br>b B<br>ε

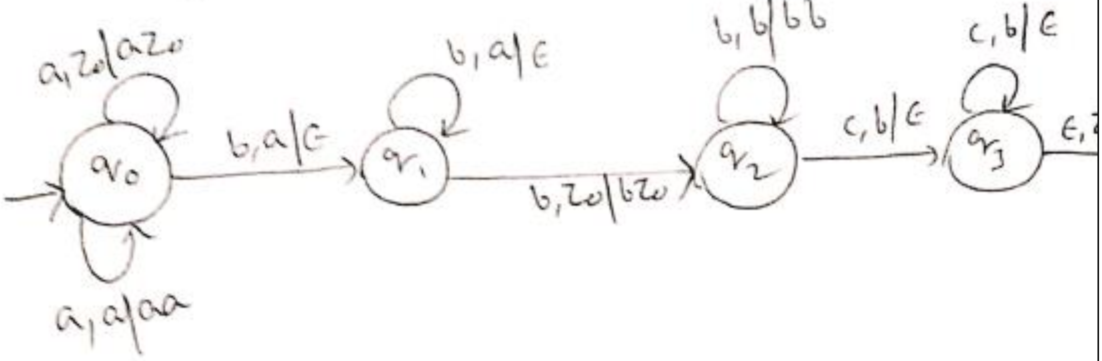
</td><td><b>10</b></td><td><b>L2</b></td><td>CO 3</td></tr>
<tr><td>6 (a)</td><td><br>Define the following with example :<br>i) Context free grammar ii) Left most Derivation<br>iii) Parse tree iv) Ambiguous grammar.<br>i) A context-free grammar (CFG) consisting of a finite set of grammar rules is a quadruple <b>(V, T, P, S)</b> where</td><td><b>4</b></td><td><b>L1</b></td><td>CO 3</td></tr>
</table>

| | | | | |
|---|---|---|---|---|
| | V is a set of non-terminal symbols.<br>**T** is a set of terminals<br>**P** is a set of rules,<br>**S** is the start symbol.<br>**Ex: The grammar ({A}, {a, b, c}, P, A),**<br>**P : A → aA,**<br>**A → abc**<br><br>(ii) **Leftmost derivation** − A leftmost derivation is obtained by applying production to the leftmost variable in each step.<br>(iii) A derivation tree or **parse tree** is an ordered rooted tree that graphically represents the semantic information a string derived from a context-free grammar.<br>(iv) A Context-Free Grammar (CFG) is called **ambiguous** if there is a string that can have more than one valid derivation tree. This means the string can be generated in different ways, either through different LeftMost Derivations (LMDT) or RightMost Derivations (RMDT).<br>Example 1. Let us consider this grammar: E-> E +E \| E*E\| id, We can create 2 parse tree from this grammar to obtain a string **id + id * id.** | | | |
| (b) |  Design PDA for the language :<br>$L = \{a^i b^j c^k / i + k = j , i \geq 0 , k \geq 0\}$ and show the moves made b<br>for the string aabbbc. | **10** | **L3** | CO 3 |

$$L = \{ a^i b^j c^k \mid i+k = j ,\ i \geq 0 ,\ k \geq 0 \}$$

$$a^i \; b^{i+k} \; c^k$$

$$a^i \; b^i \; b^k c^k$$



① $\delta(q_0, a, Z_0) = (q_0, aZ_0)$

② $\delta(q_0, a, a) = (q_0, aa)$

③ $\delta(q_0, b, a) = (q_1, \epsilon)$

④ $\delta(q_1, b, a) = (q_1, \epsilon)$

⑤ $\delta(q_1, b, Z_0) = (q_2, bZ_0)$

⑥ $\delta(q_2, b, b) = (q_2, bb)$

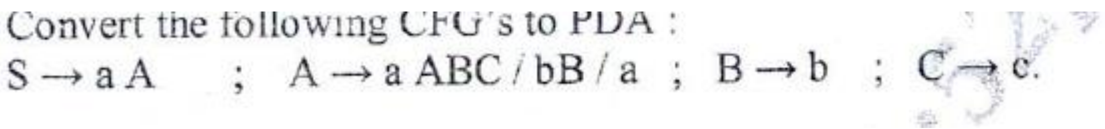
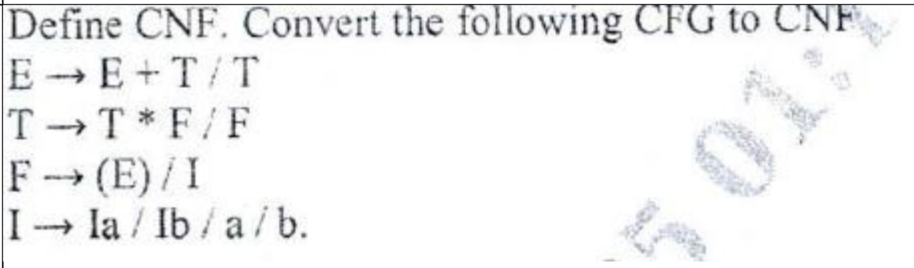
⑦ $\delta(q_2, c, b) = (q_3, \epsilon)$

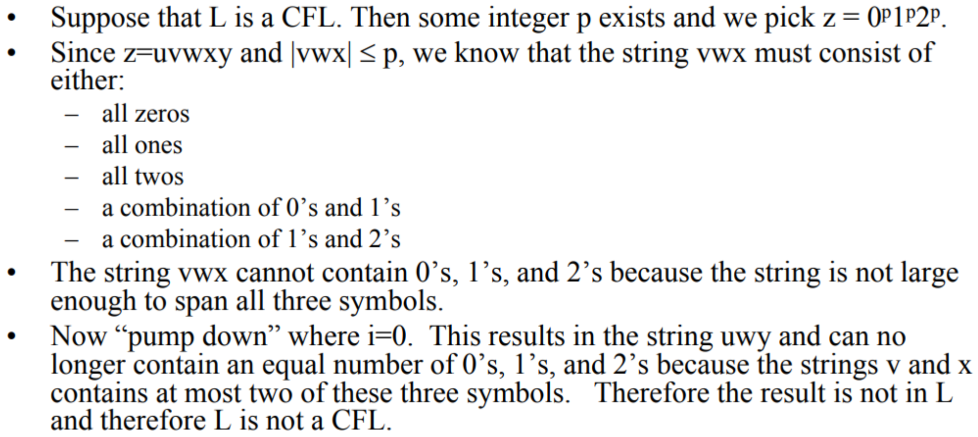
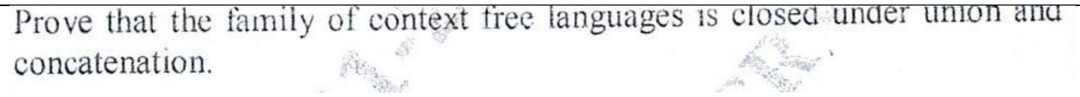
⑧ $\delta(q_3, c, b) = (q_3, \epsilon)$

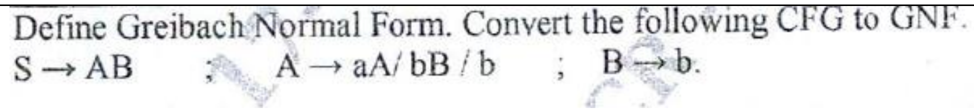
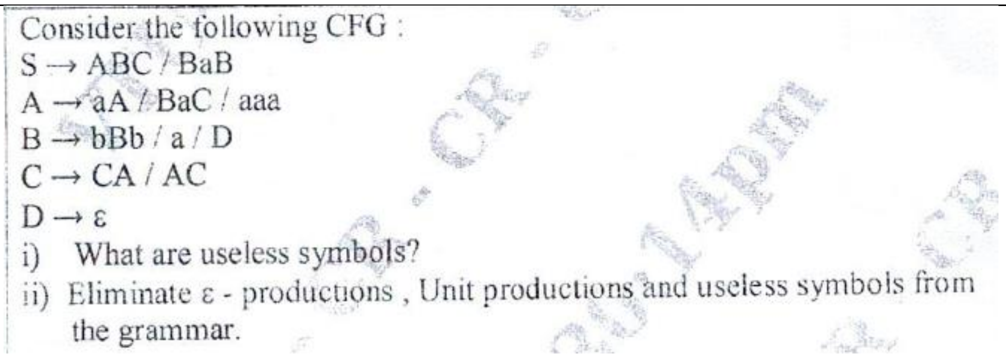
⑨ $\delta(q_3, \epsilon, Z_0) = (q_4, \epsilon)$

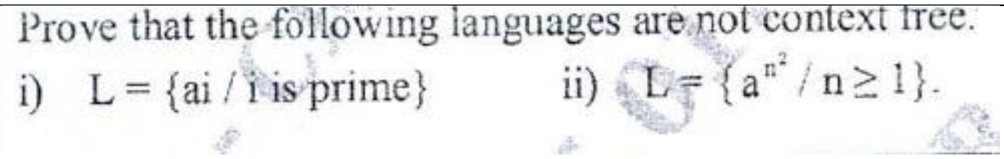
Moves made by PDA for the input w =aabbbc is given below.

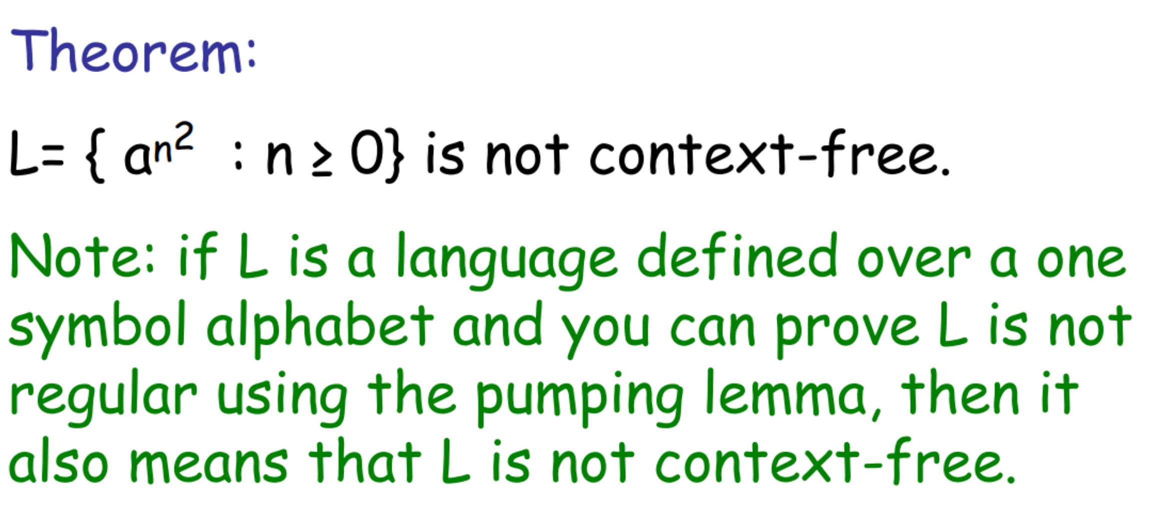
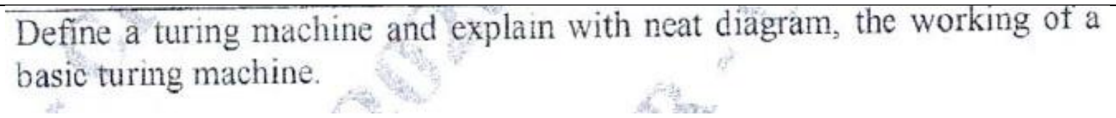
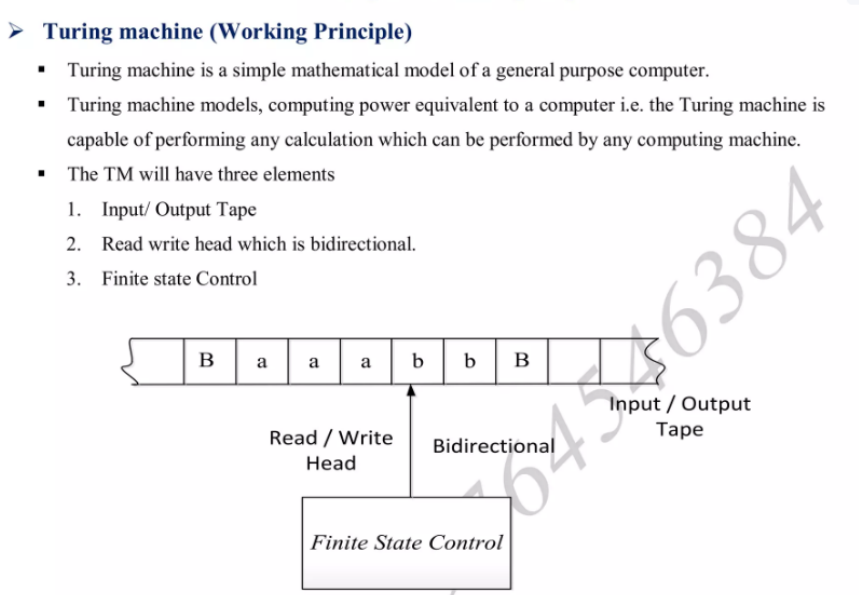
($q_0$,aabbbc,$Z_0$ ) |- ($q_0$,abbbc,a$Z_0$ ) |- ($q_0$, bbbc, aa$Z_0$ ) |- ($q_1$, bbc, a$Z_0$ ) |- ($q_1$, bc, $Z_0$ ) |- ($q_2$, c, b$Z_0$ ) |- ($q_3$, Ɛ, $Z_0$ ) |- ($q_4$, Ɛ, Ɛ ) **ACCEPTED**

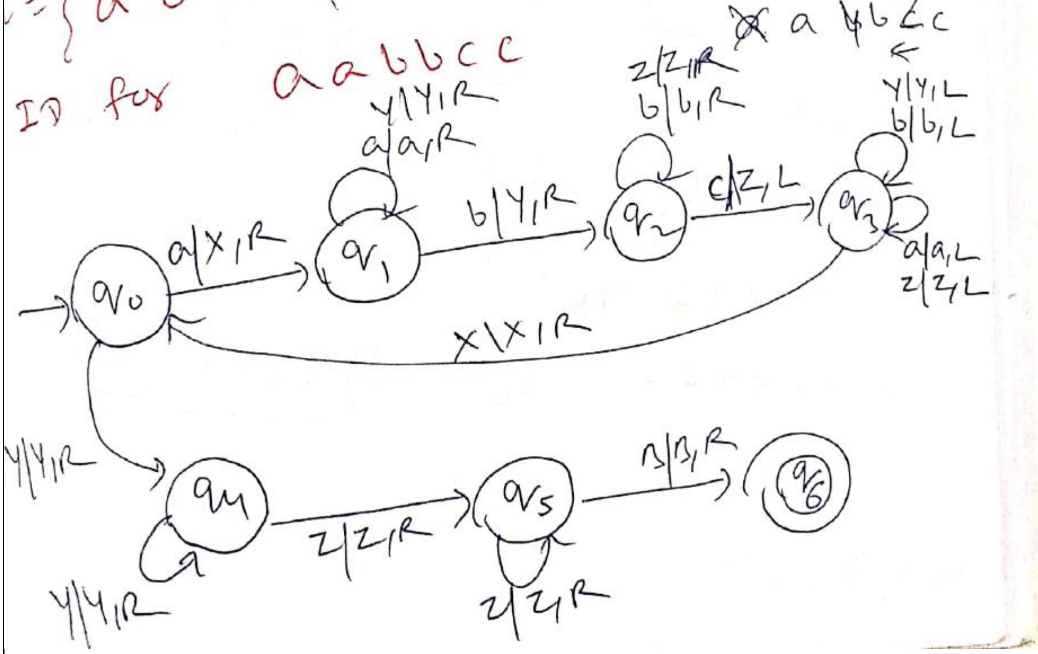
| | | | | |
|---|---|---|---|---|
| (C) | <br>Convert the following CFG's to PDA :<br>S → a A ; A → a ABC / bB / a ; B → b ; C → c.<br>1. δ (q,Ɛ,S)=(q,aA)<br>2. δ (q,Ɛ,A)= {(q,aABC), (q,bB), (q,a)}<br>3. δ (q,Ɛ,B)=(q,b)<br>4. δ (q,Ɛ,C)=(q,c)<br>5. δ (q,a,a)=(q,Ɛ)<br>6. δ (q,b,b)=(q,Ɛ)<br>7. δ (q,c,c)=(q,Ɛ) | **6** | **L2** | CO 3 |
| 7. (a) | <br>Define CNF. Convert the following CFG to CNF<br>E → E + T / T<br>T → T * F / F<br>F → (E) / I<br>I → Ia / Ib / a / b.<br>A context free grammar (CFG) is in Chomsky Normal Form (CNF) if all production rules satisfy one of the following conditions:<br>- A non-terminal generating a terminal (e.g.; X->x)<br>- A non-terminal generating two non-terminals (e.g.; X->YZ)<br>- Start symbol generating ε. (e.g.; S-> ε)<br>Step 1: No epsilon production<br>Step 2: No useless symbol<br>Step 3: After removing Unit production<br>E → E+T \| T*F \| (E) \| Ia \| Ib \| a \| b<br>T → T*F \| (E) \| Ia \| Ib \| a \| b<br>F→ (E) \| Ia \| Ib \| a \| b<br>I → Ia \| Ib \| a \| b<br>Step 4: Converting to CNF<br>E→EL \| TM \| RN \| IX \| IY \|a \|b<br>T→TM \| RN \| IX \| IY \| a \| b<br>X→a S→) L→PT M→QF N→ES<br>Y→b<br>P→+<br>Q→*<br>R→( | **10** | **L2** | CO 4 |

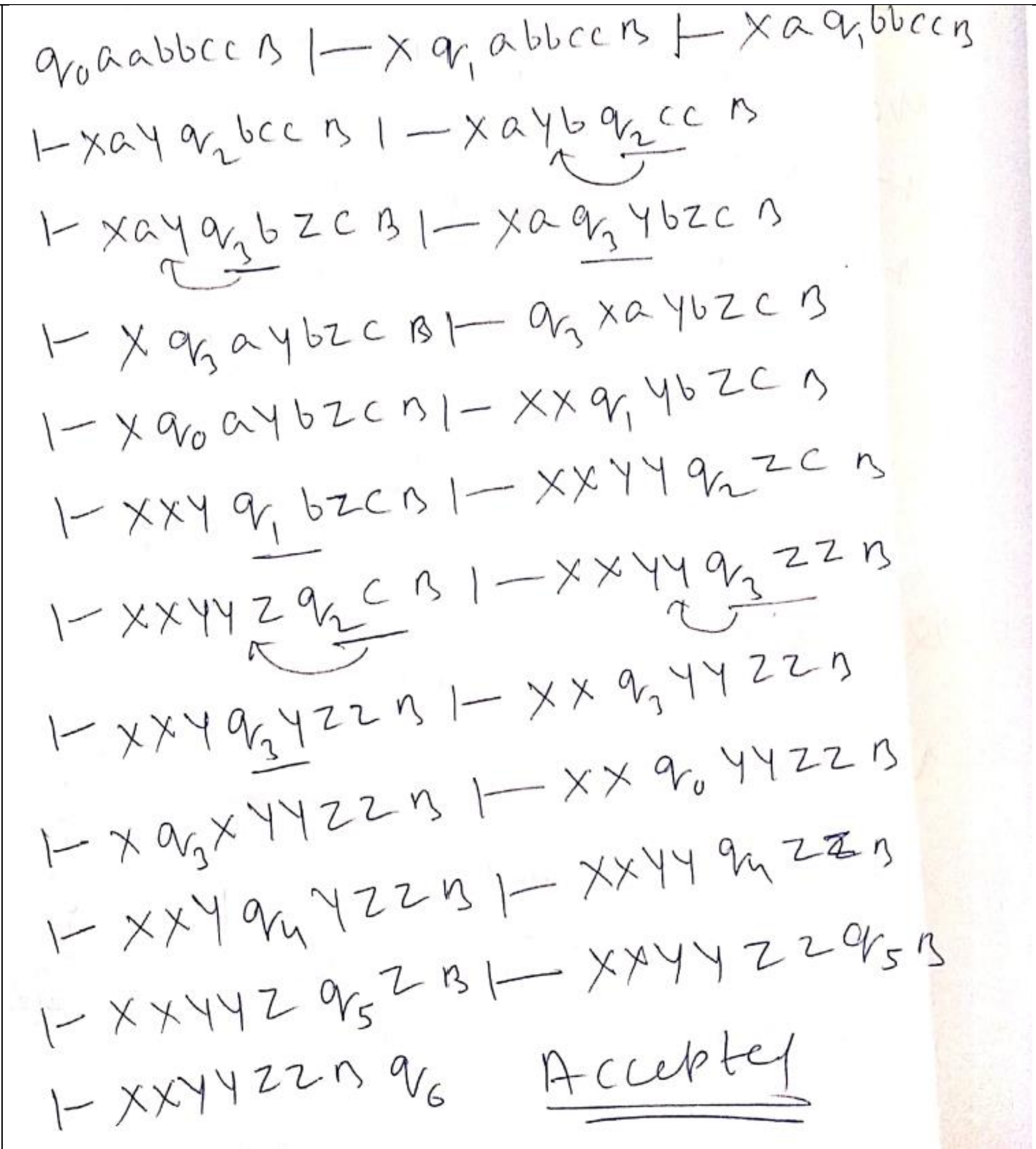
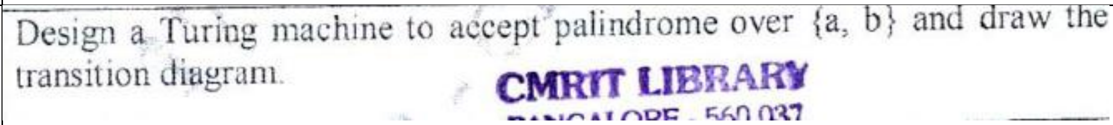
| | | | | |
|---|---|---|---|---|
| | | | | |
| 7<br>(b) | **Show that L ={$0^n1^n2^n$ \| n>=1} is not context free**<br><br>• Suppose that L is a CFL. Then some integer p exists and we pick z = $0^p1^p2^p$.<br>• Since z=uvwxy and \|vwx\| ≤ p, we know that the string vwx must consist of either:<br>– all zeros<br>– all ones<br>– all twos<br>– a combination of 0's and 1's<br>– a combination of 1's and 2's<br>• The string vwx cannot contain 0's, 1's, and 2's because the string is not large enough to span all three symbols.<br>• Now "pump down" where i=0. This results in the string uwy and can no longer contain an equal number of 0's, 1's, and 2's because the strings v and x contains at most two of these three symbols. Therefore the result is not in L and therefore L is not a CFL. | **4** | **L2** | CO<br>4 |
| ( c<br>) | <br>Prove that the family of context free languages is closed under union and concatenation.<br>**1. Union**<br>If **L** and **M** are two context-free languages, then their union **L ∪ M** is also a CFL.<br>**Construction:**<br>1. Consider two context-free grammars, **G** and **H**, for **L** and **M** respectively.<br>2. Assume that **G** and **H** have no common variables (this can be ensured by renaming variables if needed).<br>3. Introduce a new start symbol **S** and add the rule: **S** → **$S_1$ \| $S_2$** Here, **$S_1$** and **$S_2$** are the start symbols of **G** and **H**, respectively.<br>4. The resulting grammar generates **L ∪ M**, proving that CFLs are closed under union.<br>**Example:** Let **$L_1$ = { $a^nb^nc^m$ \| m ≥ 0, n ≥ 0 }** and **$L_2$ = { $a^nb^mc^m$ \| n ≥ 0, m ≥ 0 }**.<br>• ***$L_1$*** *enforces that the number of* ***a's*** *equals the number of* ***b's****.*<br>• ***$L_2$*** *enforces that the number of* ***b's*** *equals the number of* ***c's****.*<br>• *Their* ***union*** *states that either of these conditions must be satisfied, making the resulting language context-free.*<br>**Note:** So CFL are closed under Union.<br><br>**2. Concatenation**<br>If **L** and **M** are CFLs, then their concatenation **LM** is also a CFL.<br>**Construction:**<br>1. Let **G** and **H** be the context-free grammars for **L** and **M**, respectively.<br>2. Assume that **G** and **H** have no common variables. | **6** | **L1** | CO<br>4 |

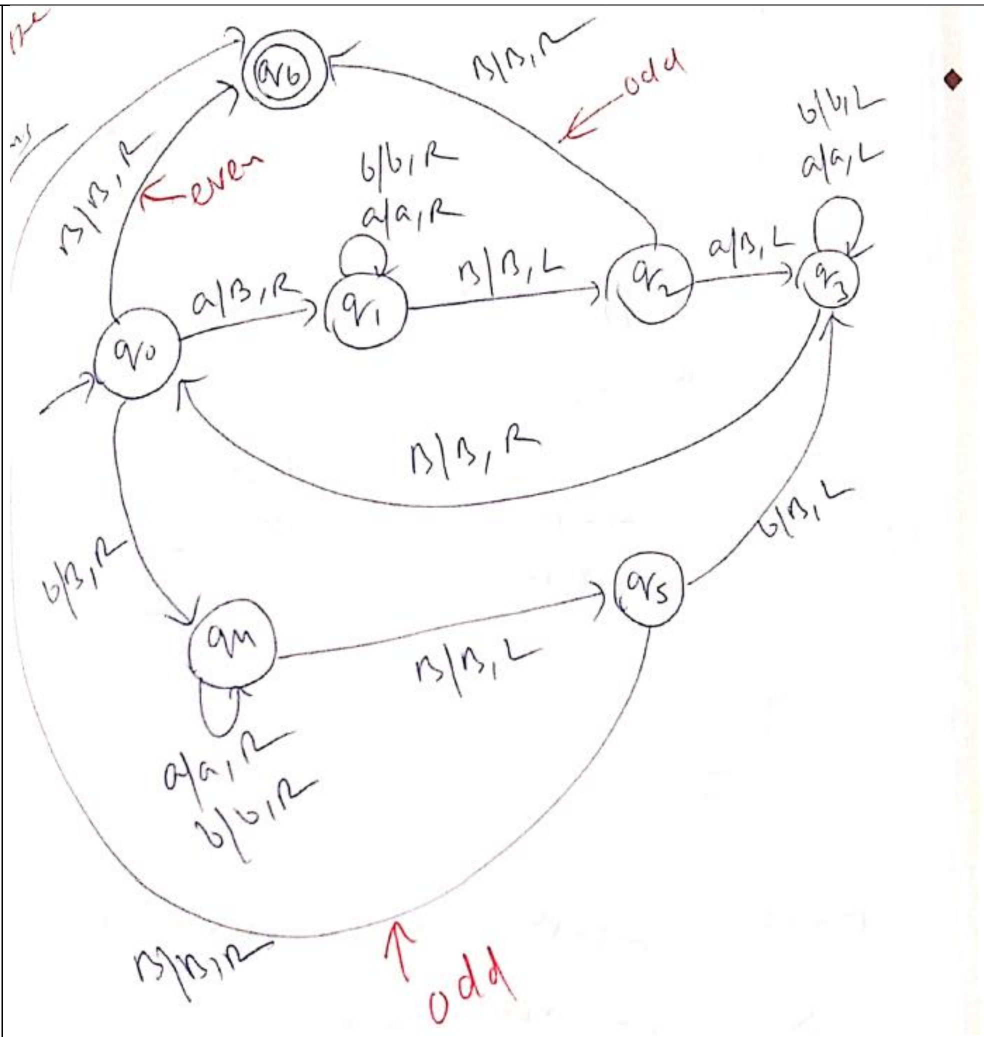
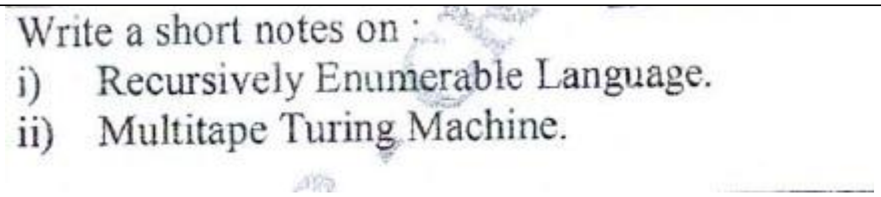
| | | | | |
|---|---|---|---|---|
| | 3. Introduce a new start symbol **S** and add the production:**S** → **S**₁ **S**₂Here, **S**₁ and **S**₂ are the start symbols of **G** and **H**, respectively.<br>4. This ensures that any derivation from **S** will first generate a string in **L** and then a string in **M**, proving that CFLs are closed under concatenation. | | | |
| 8 (a) | <br>Define Greibach Normal Form. Convert the following CFG to GNF.<br>$S \rightarrow AB$ ; $A \rightarrow aA/ bB / b$ ; $B \rightarrow b$.<br><br>Note: Out of Syllabus. Students got gress mark for this | **6** | **L2** | CO 4 |
| (b) | <br>Consider the following CFG :<br>$S \rightarrow ABC / BaB$<br>$A \rightarrow aA / BaC / aaa$<br>$B \rightarrow bBb / a / D$<br>$C \rightarrow CA / AC$<br>$D \rightarrow \varepsilon$<br>i) What are useless symbols?<br>ii) Eliminate ε - productions , Unit productions and useless symbols from the grammar.<br>Ans:<br>A non terminal is useless if it cann't be reached from start state or it cann't generate terminals.<br>Step 1: Elimination of epsilon production:<br>Null Set ={B, D}<br>$S \rightarrow ABC \mid BaB \mid Ba \mid aB \mid a \mid AC$<br>$A \rightarrow aA \mid BaC \mid aaa \mid aC$<br>$B \rightarrow bBb \mid a \mid D \mid bb$<br>$C \rightarrow CA \mid AC$<br>Step2: Removal of Useless Symbol<br>C & D are useless<br>$S \rightarrow BaB \mid Ba \mid aB \mid a$<br>$A \rightarrow aA \mid aaa$<br>$B \rightarrow bBb \mid a \mid bb$<br>Now A is useless<br>$S \rightarrow BaB \mid Ba \mid aB \mid a$<br>$B \rightarrow bBb \mid a \mid bb$ | **10** | **L3** | CO 4 |

| | | | | |
|---|---|---|---|---|
| 8 ( c ) | <br>Prove that the following languages are not context free.<br>i) L = {ai / i is prime}   ii) $L = \{a^{n^2} / n \geq 1\}$.<br>Theorem: L= { $a^p$ : p is a prime number} is not context-free.<br>Assume L is context-free and is generated by a context-free grammar G. Then there exists some constant k dependent on G such that for all strings w in L of length at least k the Pumping Theorem holds. Choose w= $a^p$ for some prime number p ≥ k.<br>Factor $a^p$ in all ways as u v x y z:<br>$w = a^i a^j a^r a^s a^t$ where $p= i + j + r + s + t$,<br>$j + s \geq 1$, $v=a^j$ and $y= a^s$.<br>Pumping n+1 times yields:<br>$a^i (a^j)^{n+1} a^r (a^s)^{n+1} a^t = a^i a^r a^t (a^j)^{n+1} (a^s)^{n+1}$<br>$= a^{p-j-s + (j+s)(n+1)}$<br>Let x= j+s.<br>Pumping n+1 times yields: $a^{p+xn}$<br>$w = a^i a^j a^r a^s a^t$<br>Let x= j+s.<br>Pumping n+1 times yields: $a^{p+xn}$<br>Pump p+1 times:<br>Gives $a^{p+xp} = a^{p(x+1)}$<br>Since x ≥ 1, (x+1) ≥ 2 and so p (x+1)<br>Cannot be prime. | **4** | **L2** | CO 3 |

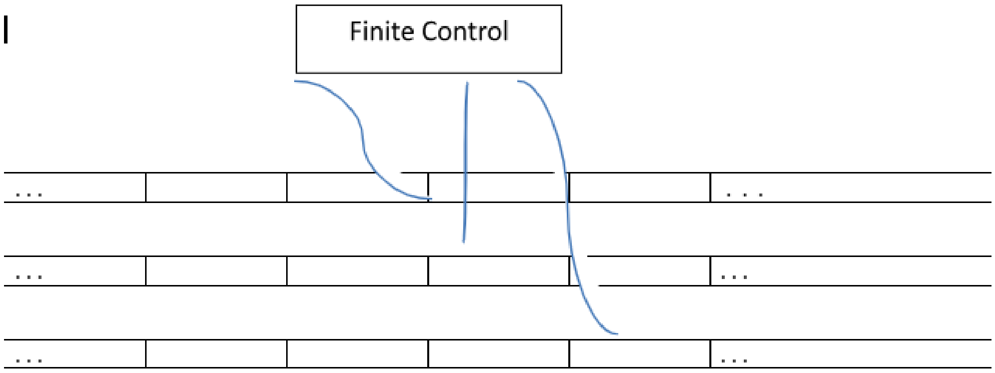
| | | | | |
|---|---|---|---|---|
| | <br>Theorem:<br>L= { $a^{n^2}$ : n ≥ 0} is not context-free.<br>Note: if L is a language defined over a one symbol alphabet and you can prove L is not regular using the pumping lemma, then it also means that L is not context-free. | | | |
| 9 ( a) | <br>Define a turing machine and explain with neat diagram, the working of a basic turing machine.<br><br>➢ **Turing machine (Working Principle)**<br>▪ Turing machine is a simple mathematical model of a general purpose computer.<br>▪ Turing machine models, computing power equivalent to a computer i.e. the Turing machine is capable of performing any calculation which can be performed by any computing machine.<br>▪ The TM will have three elements<br>1. Input/ Output Tape<br>2. Read write head which is bidirectional.<br>3. Finite state Control<br> B a a a b b B<br>Read / Write Head<br>Bidirectional<br>Input / Output Tape<br>Finite State Control | **6** | **L1** | CO 4 |

| | | | | |
|---|---|---|---|---|
| | ▪ The Turing machine can be thought of as a finite automata connected to read/ write head which is bidirectional i.e. can be move left to write and write to left.<br>▪ It has o ne input / output tape which is divided in many cells. At each cell only one input symbol is placed.<br>▪ The input and output on tape affected by the read/ write head which can examine one cell in one move.<br>▪ In one move, the machine examine the present symbol under the read/ write head on the tape and the present state of an automaton to determine<br>☞ What to do with tape present symbol i.e. a new symbol to be written or no change of symbol on the tape in the cell.<br>☞ In which direction head move i.e. either the head moves one cell left (L), or one cell right (R), or stay at the same cell (N).<br>☞ The next state of machine.<br>▪ TM is more powerful than other all machines.<br>▪ TM is used for recognizing all type languages especially for Type-0 and Type-1. | | | |
| (b) | Design a Turing machine to accept the language, L = {$a^n$ $b^n$ $c^n$ / n ≥ 1}. Draw the transition diagram and show the moves for the string aabbcc.<br>Design a TM to accept L={$a^n$ $b^n$ $c^n$ \| n ≥ 1}. Show the ID for aabbcc<br> ~~X~~ a ~~Y~~ b Z c<br>←<br>Y\|Y,R<br>a\|a,R<br>Z\|Z,R<br>b\|b,R<br>Y\|Y,L<br>b\|b,L<br>a\|X,R<br>b\|Y,R<br>c\|Z,L<br>a\|a,L<br>Z\|Z,L<br>X\|X,R<br>q0<br>q1<br>q2<br>q3<br>Y\|Y,R<br>q4<br>Y\|Y,R<br>Z\|Z,R<br>q5<br>Z\|Z,R<br>B\|B,R<br>q6 | **14** | **L4** | CO 4 |

| | | | | |
|---|---|---|---|---|
| | <br>$q_0$aabbccB ⊢ x$q_1$abbccB ⊢ xa$q_1$bbccB<br>⊢ xaY$q_2$bccB ⊢ xaYb$q_2$ccB<br>⊢ xaY$q_3$bZcB ⊢ xa$q_3$YbZcB<br>⊢ X$q_3$aYbZcB ⊢ $q_3$xaYbZcB<br>⊢ x$q_0$aYbZcB ⊢ XX$q_1$YbZcB<br>⊢ XXY$q_1$bZcB ⊢ XXYY$q_2$ZcB<br>⊢ XXYYZ$q_2$cB ⊢ XXYY$q_3$ZZB<br>⊢ XXY$q_3$YZZB ⊢ XX$q_3$YYZZB<br>⊢ X$q_3$XYYZZB ⊢ XX$q_0$YYZZB<br>⊢ XXY$q_4$YZZB ⊢ XXYY$q_4$ZZB<br>⊢ XXYYZ$q_5$ZB ⊢ XXYYZZ$q_5$B<br>⊢ XXYYZZB$q_6$ Accepted | | | |
| 10 (a) | <br>Design a Turing machine to accept palindrome over {a, b} and draw the transition diagram.<br> | **12** | **L4** | CO 5 |

| | | | | |
|---|---|---|---|---|
| |  q6<br>B\|B,R<br>odd<br>B\|B,R<br>even<br>b\|b,R<br>a\|a,R<br>a\|B,R<br>q1<br>B\|B,L<br>q2<br>a\|B,L<br>b\|b,L<br>a\|a,L<br>q3<br>q0<br>B\|B,R<br>b\|B,R<br>q5<br>b\|B,L<br>q4<br>B\|B,L<br>a\|a,R<br>b\|b,R<br>B\|B,R<br>odd | | | |
| (b) | <br>Write a short notes on :<br>i) Recursively Enumerable Language.<br>ii) Multitape Turing Machine.<br>**Recursive Enumerable (RE) or Type -0 Language**<br>RE languages or type-0 languages are generated by type-0 grammars. An RE language can be accepted or recognized by Turing machine which means it will enter into final state for the strings of language and may or may not enter into rejecting state for the strings which are not part of the language. It means TM can loop forever for the strings which are not a part of the language. RE languages are also called as Turing recognizable languages.<br>**MULTITAPE TURING MACHINE**<br>Multi-tape Turing Machines have multiple tapes where each tape is accessed with a separate head. Each head can move independently of the other heads. Initially the input is on tape 1 and others are blank. At first, the first tape is occupied by the input and the other | **8** | **L1** | CO 5 |

tapes are kept blank. Next, the machine reads consecutive symbols under its heads and the TM prints a symbol on each tape and moves its heads.



A Multi-tape Turing machine can be formally described as a 7-tuple (Q,Σ,Γ, B, δ, $q_0$, F) where −

- **Q** is a finite set of states
- Σ is a finite set of inputs
- Γ is the tape alphabet
- **B** is the blank symbol
- **δ** is a relation on states and symbols where

  $\delta: Q \times \Gamma^k \rightarrow Q \times (\Gamma \times \{\text{Left, Right, Stationary}\})^k$

  where there is **k** number of tapes
- **$q_0$** isl state
- **F** is the set of fina the initial states

  In each move the machine M:

  (i) Enters a new state

  (ii) A new symbol is written in the cell under the head on each tape

  (iii) Each tape head moves either to the left or right or remains stationary.