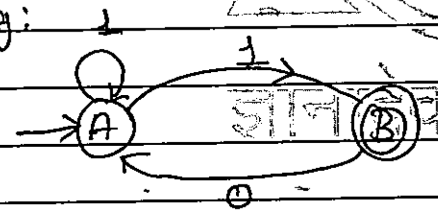
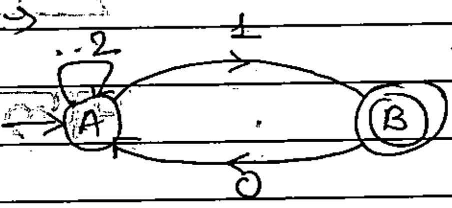
Unit :-> 1

Q.1 (a)

NFA	DFA
(1) Non-Deterministic Finite Automata can have multiple transitions from a state on a single input	Deterministic Finite Automata always has only & only one transition from a state on a single input.
(2) NFA can contain empty (i.e. ε) as input symbol	DFA can't contain empty input symbols
(3) NFA from presentation point may have multiple routes from a single state & input	DFA from presentation point ~~me~~ only has a single route from a ~~state~~ single state & input
(4) eg:  $\delta(A,1) = A,B$ $\delta(B,0) = A$	eg:->  $\delta(A,1) = B$, $\delta(A,2) = A$ $\delta(B,0) = A$

(P.T.O)

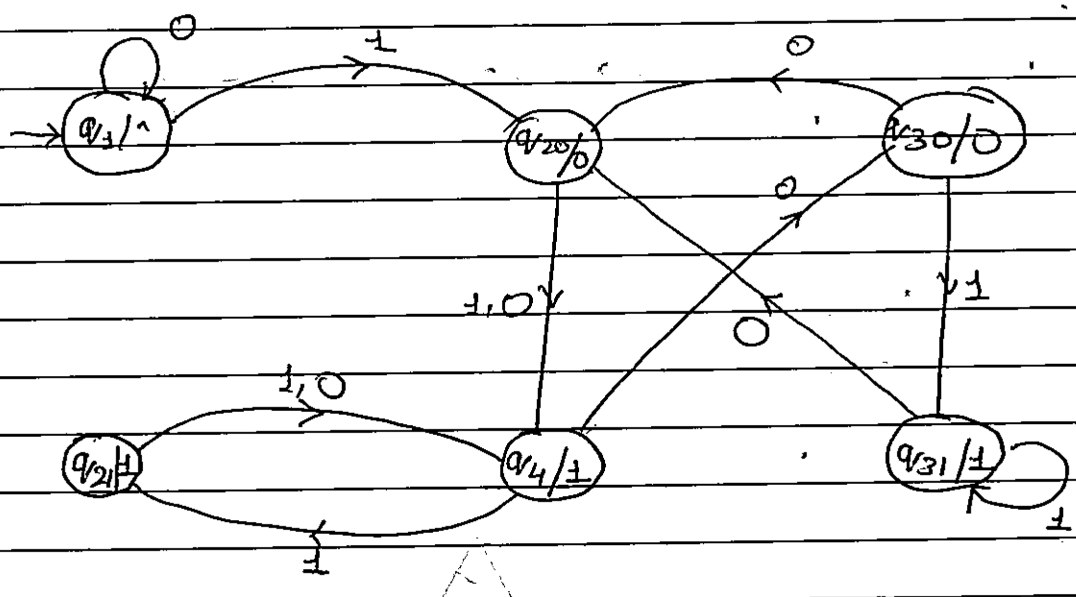
1b) from given moore machine transition table it is evident that q_3, q_2 on input $a=1$ has dissimilar output, therefore

$$q_2 \begin{cases} \to 0 \to q_{20} \to 0 \\ \to 1 \to q_{21} \to 1 \end{cases}$$

$$q_3 \begin{cases} \to 0 \to q_{30} \to 0 \\ \to 1 \to q_{31} \to 1 \end{cases}$$

Construction of Mealy Machine transition table

	in $a=0$	in $a=1$	output
$\to q_1$	q_1	q_{21}	
q_{20}	q_4	q_4	0
q_{21}	q_4	q_4	1
q_{30}	q_{20}	q_{31}	0
q_{31}	q_{20}	q_{31}	1
q_4	q_{30}	q_{21}	1



1c) Step I & II

	q_0	q_1	q_2	q_3	q_4	q_5	q_6	q_7
q_0								
q_1								
q_2	✓	✓						
q_3	✓	✓	✓					
q_4			[illegible]	✓				
q_5	✓	✓	✓	✓	✓			
q_6								
q_7								
q_8								

1 c) Step I & II Creating Half matrix & marking pairs accordingly

q_0								
q_1	✓							
q_2	✓	✓						
q_3	✓	✓						
q_4	✓		✓	✓				
q_5	✓	✓			✓			
q_6	✓	✓			✓			
q_7	✓	✓	✓	✓	✓	✓	✓	
q_8	✓	✓	✓	✓	✓	✓	✓	
	q_0	q_1	q_2	q_3	q_4	q_5	q_6	q_7

unmarked pairs are $(q_0 q_1)$, $(q_3 q_2)$, $(q_4 q_0)$, $(q_4 q_1)$

$(q_5 q_2)$, $(q_5 q_3)$, $(q_6 q_2)$, $(q_6 q_3)$, $(q_6 q_5)$

$(q_7 q_0)$, $(q_7 q_1)$, $(q_7 q_4)$, $(q_8 q_0)$

$(q_8 q_1)$, $(q_8 q_4)$, $(q_8 q_7)$

Step III checking δ transition for marking the pair for each currently unmarked pair

$q_0 q_1$

$\left.\begin{array}{l}\delta(q_0;0) = q_1 \\ \delta(q_1;0) = q_2\end{array}\right\}$ ✓

$q_3 q_2$

$\left.\begin{array}{l}\delta(q_3;0) = q_8 \\ \delta(q_2,0) = q_7\end{array}\right\}$ ✗ $\quad \left.\begin{array}{l}\delta(q_3,1) = q_7 \\ \delta(q_2,1) = q_8\end{array}\right\}$ ✗

$q_4 q_0$

$\left.\begin{array}{l}\delta(q_4,0) = q_5 \\ \delta(q_0,0) = q_1\end{array}\right\}$ ✓

$q_4 q_1$

$\left.\begin{array}{l}\delta(q_4;0) = q_5 \\ \delta(q_1,0) = q_2\end{array}\right\}$ ✗ $\quad \left.\begin{array}{l}\delta(q_4,1) = q_6 \\ \delta(q_1,1) = q_3\end{array}\right\}$ ✗

$q_5 q_2$

$\left.\begin{array}{l}\delta(q_5;0) = q_7 \\ \delta(q_2;0) = q_7\end{array}\right\}$ ✗ $\quad \left.\begin{array}{l}\delta(q_5,1) = q_8 \\ \delta(q_2,1) = q_8\end{array}\right\}$ ✗

$q_5 q_3$

$\left.\begin{array}{l}\delta(q_5,0) = q_7 \\ \delta(q_3,0) = q_8\end{array}\right\}$ ✗ $\quad \left.\begin{array}{l}\delta(q_5;1) = q_8 \\ \delta(q_3,1) = q_7\end{array}\right\}$ ✗

$q_6 q_2$

$\left.\begin{array}{l}\delta(q_6,0) = q_7 \\ \delta(q_2,0) = q_7\end{array}\right\}$ ✗ $\quad \left.\begin{array}{l}\delta(q_6,1) = q_8 \\ \delta(q_2,1) = q_8\end{array}\right\}$ ✗

$q_6 q_3$

$\left.\begin{array}{l}\delta(q_6, 0) = q_7 \\ \delta(q_3, 0) = q_8\end{array}\right\} \times$ $\quad$ $\left.\begin{array}{l}\delta(q_6, 1) = q_8 \\ \delta(q_3, 1) = q_7\end{array}\right\} \times$

$q_6 q_5$

$\left.\begin{array}{l}\delta(q_6, 0) = q_7 \\ \delta(q_5, 0) = q_7\end{array}\right\} \times$ $\quad$ $\left.\begin{array}{l}\delta(q_6, 1) = q_8 \\ \delta(q_5, 1) = q_8\end{array}\right\} \times$

$q_7 q_0$

$\left.\begin{array}{l}\delta(q_7, 0) = q_7 \\ \delta(q_0, 0) = q_1\end{array}\right\} \checkmark$

$q_7 q_1$

$\left.\begin{array}{l}\delta(q_7, 0) = q_7 \\ \delta(q_1, 0) = q_2\end{array}\right\} \checkmark$

$q_7 q_4$

$\left.\begin{array}{l}\delta(q_7, 0) = q_7 \\ \delta(q_4, 0) = q_6\end{array}\right\} \checkmark$

$q_8 q_0$

$\left.\begin{array}{l}\delta(q_8, 0) = q_8 \\ \delta(q_0, 0) = q_1\end{array}\right\} \checkmark$

$q_8 q_1$

$\left.\begin{array}{l}\delta(q_8, 0) = q_8 \\ \delta(q_1, 0) = q_2\end{array}\right\} \checkmark$

$q_8 q_4$

$$\left.\begin{array}{l}\delta(q_8,0) = q_8 \\ \delta(q_4,0) = q_5\end{array}\right\} \checkmark$$

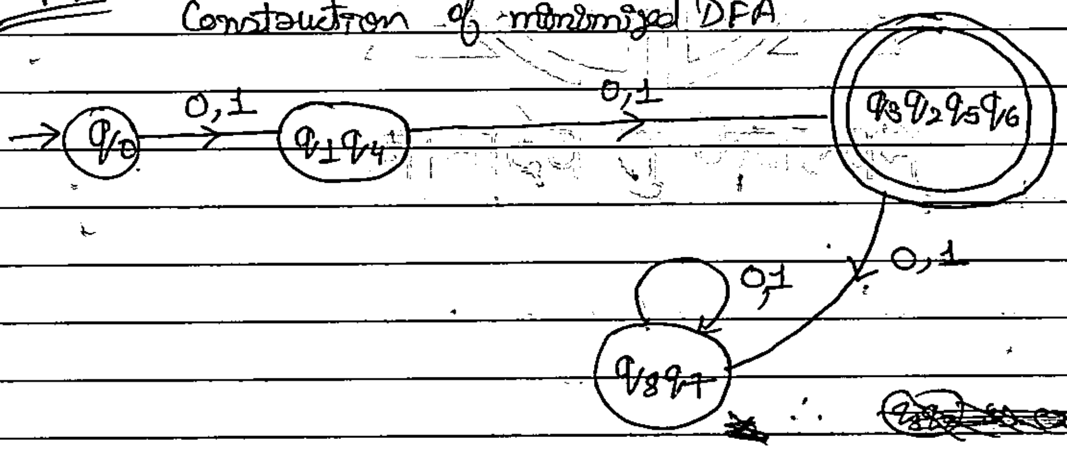
$q_8 q_7$

$$\left.\begin{array}{l}\delta(q_8,0) = q_8 \\ \delta(q_7,0) = q_7\end{array}\right\} \times \qquad \left.\begin{array}{l}\delta(q_8,1) = q_8 \\ \delta(q_7,1) = q_7\end{array}\right\} \times$$

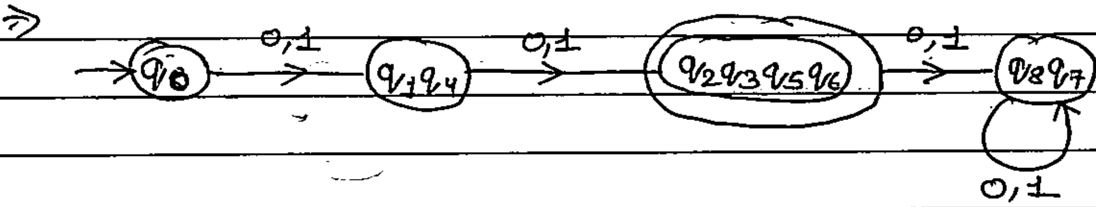
Step IV Final Unmarked pairs

$(q_3 q_2)$, $(q_4 q_1)$, $(q_5 q_2)$, $(q_5 q_3)$, $(q_6 q_2)$, $(q_6 q_3)$, $(q_6 q_5)$, $(q_8 q_7)$

Step V Construction of minimized DFA



∴ Final DFA



Unit 2

Q2b

2a) Closure property in regular grammar is explained as the different iterations possible for an input variable

Closure in regular grammar consists mainly of two types namely

1. Kleen Closure
2. +ve Closure

Kleen ~~sou~~ closure includes the empty string as a member of the set & is represented by V^*

eg:- $A^* = \{\epsilon, A, AA, AAA, AAAA, \ldots\}$

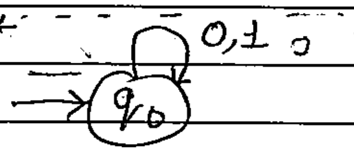
+ve ~~or~~ Positive closure include all possible string combination except ϵ or empty string, Denoted by V^+.

eg: $A^+ = \{A, AA, AAA, AAAA, \ldots\}$

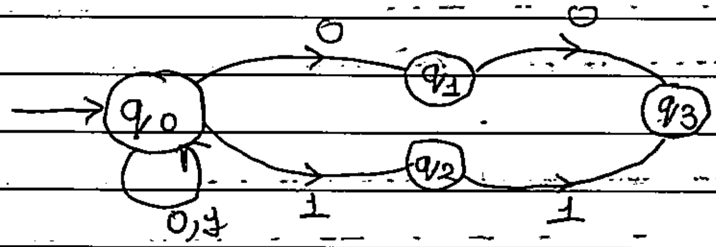
2b) $RE = (0+1)^* (00+11) (0+1)^*$

Constructing step by step Finite Automata

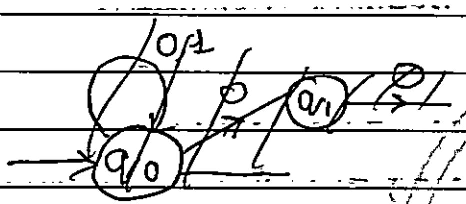
Step I $(0+1)^*$

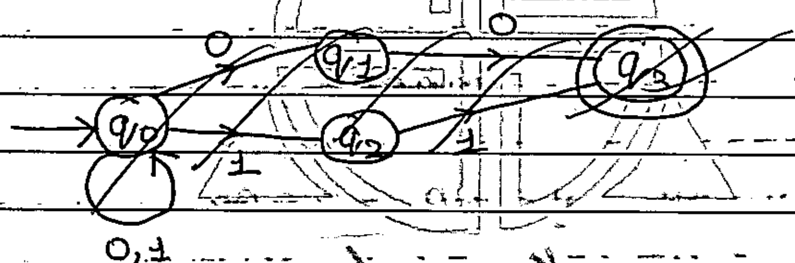


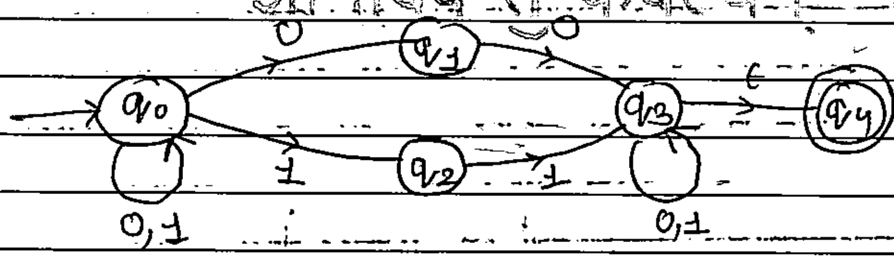
Step II $(0+1)^*(00+11)$



Step III $(0+1)^*(00+11)(0+1)^*$







Transition Functions

↳ $\delta(q_0, 0) = q_0 / q_1$ $\qquad$ $\delta(q_3, 0) = q_3$

$\delta(q_0, 1) = q_0 / q_2$ $\qquad$ $\delta(q_3, 1) = q_3$

$\delta(q_1, 0) = q_3$ $\qquad$ $\delta(q_3, \epsilon) = q_4$

$\delta(q_2, 1) = q_3$

2 (d) Using Pumping lemma for Language

$$L = \{a^n b^n, n \geq 1\}$$

& Pumping length $p = 3$

$$S = a^3 b^3$$
$$= aaabbb$$

$\because S = xyz$ for a regular language

Let

Case I $x = \epsilon$, $y = aaabbb$, $z = \epsilon$

Case II $x = a$, $y = aabbb$, $z = \epsilon$

Case III $x = aa$, $y = ab$, $z = bb$

① Checking required condition $|y| > 0$

Case I $|y| = 6$ ✓ Case II $|y| = 5$ ✓ Case III $|y| = 2$ ✓

② Checking required condition $|xy| \geq p$

Case I $|xy| = 6$ i.e. > 3 ✓
Case II $|xy| = 6$ i.e. > 3 ✓
Case III $|xy| = 4$ i.e. > 3 ✓

(3) Checking required condition $xy^iz \in L$ for $i = 1, 2, 3 \ldots$

$xy^iz \in L \quad \forall i =$ Natural no.

$i = 2$

Case I $\quad \epsilon\,(aaabbb)^2\,\epsilon \Rightarrow aaabbb\,aaabbb \notin L$

Case II $\quad a\,(aabbb)^2\,\epsilon \Rightarrow a\,aabbb\,aabbb \notin L$

Case III $\quad aa\,(ab)^2\,bb \Rightarrow aa\,ab\,ab\,bb \notin L$

$\because$ Case I, II & III can't satisfy 3rd condition of pumping lemma.

$\therefore$ L is not a regular language.

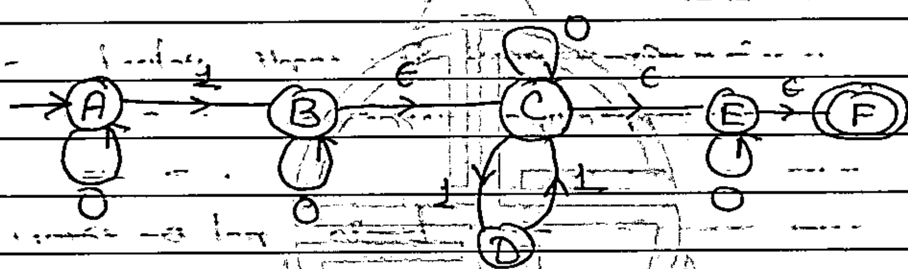
Q3.4

3a4 (1) $\Sigma = \{0,1\}$

odd no of 1's $= \cancel{0^*10^* + 0^*10^*(11)^*0^*}$

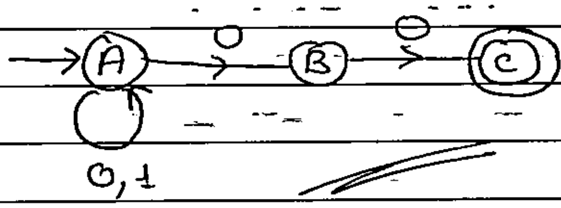
$\cancel{0^*10^* + 1(11)^*}$

$\Rightarrow \cancel{0^*10^*+}\ 0^*10^*(0+11)^*0^*$



(2) $\Sigma = \{0,1\}$

ending with 00 $\rightarrow (0+1)^*00$



3b) Chomsky Classification of grammer states that

1. A variable on LHS can derive either the empty variable "ϵ" or,

2. Two non-terminals or variables only, or,

3. only one terminal

~~This can be explained~~ Any set of grammer following the above three criteria is termed as grammer of ~~a~~ Chomsky Normal Form.

This can be explained via examples

~~general~~ $S \rightarrow AB$ $V \in \Sigma$ ~~(terminals)~~

$A \rightarrow \epsilon$

$B \rightarrow a$

This set of grammer follows all chomsky norms & hence can be classified as chomsky grammer

Any Context Free grammer can be converted into chomsky normal form using specific methods

(P.T.O.)

Sol:

$S \rightarrow 0B / 1A$

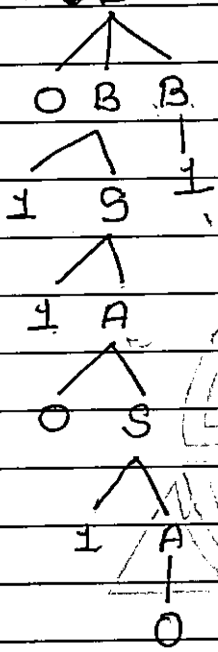
$A \rightarrow 0 / 0S / 1AA$

$B \rightarrow 1 / 1S / 0BB$

String = 00110101

~~LMD/→~~ Left Derivation tree

$S \longrightarrow 0B$



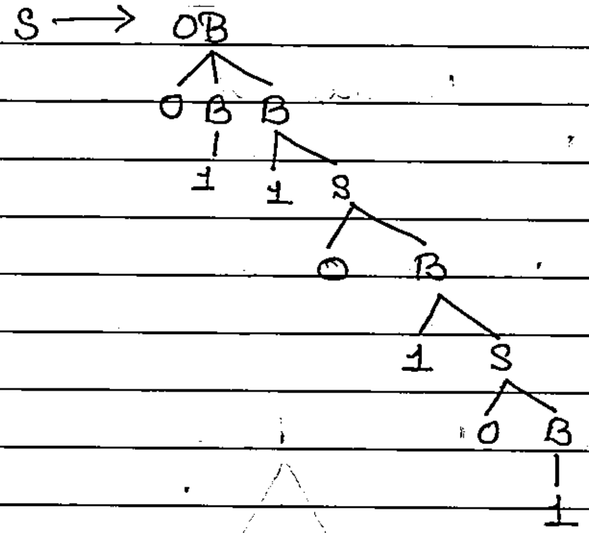
$\Rightarrow 00110101$

∴ String = 00110101

is Accepted by the given grammar when using ~~left~~ left derivation tree

P.T.O.

Right derivation tree



String = 00110101
is accepted by the given grammer
when using Right derivation tree

Unit 4

Q4
4a)

	NPDA	DPDA
1	NPDA Push down Automata has fewer states	DPDA push down automata is more states
2	NPDA is substantially faster compared to DPDA	~~NE~~ DPDA is comparatively slower
3	NPDA is ~~faster~~ harder to construct	DPDA is easier to construct

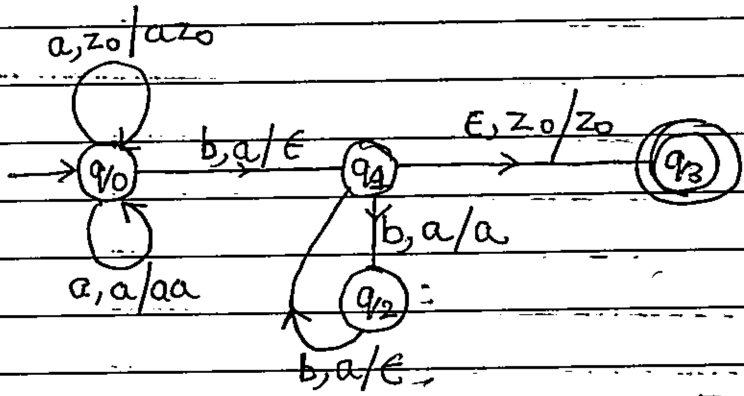
4b)

PDA = $\{Q, \Sigma, P, \cancel{\$}\ Z_0, q_0, D, \tau\}$

- Q = Set of states
- Σ = Set of input Alphabets
- P = ~~Stack~~ Production Rule
- q_0 = Initial State
- D = Stack symbol for Final State
- Z_0 ~~\$~~ = Stack Data-Structure Z_0 element
- τ = Stack Function (push, pop)

$L = \{a^n b^{2n}\}$ when $n \geq 1$ the Pushdown Automata will be constructed as

(P.T.O.)



Transition functions

$\delta(q_0, a, z_0) = (q_0, a)$, push
$\delta(q_0, a, a) = (q_0, a)$, push
$\delta(q_0, b, a) = (q_1, \epsilon)$, pop
$\delta(q_1, b, a) = (q_2, a)$, leave
$\delta(q_2, b, a) = (q_1, \epsilon)$, pop
$\delta(q_1, \epsilon, z_0) = (q_3, z_0)$, leave (Final state)

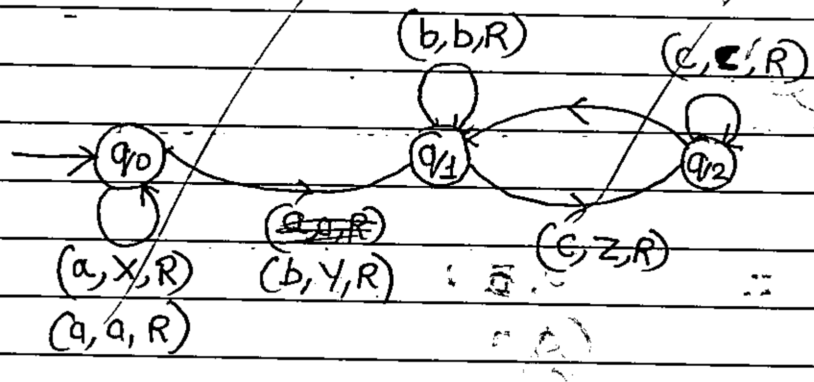
Hence PDA for Language $L = \{a^n b^{2n}\}$ $n \geq 1$ is constructed

4(c) Turing Machine = $\{Q, \Sigma, P, q_0, T, \tau, B\}$

Q = {Set of States}
Σ = {Set of input symbol}
P = Production rule
q_0 = Initial state
T = Tape
τ = Tape operation
B = Blank space for final state.

P.T.O.

for $L = a^n b^n c^n$, $n \geq 0$; Turing Machine will be constructed as follows



4d) (1) Decidable & ~~Undecidatedable~~ Undecidable Problem:

Decidable ~~po~~ problem is the term coined when there exist some turing machine that can solve ~~a problem~~ the given problem using simple functions of a recursively enumerable language, even if given a problem with complex function, it can be broken into simpler function and again a turing machine exist for that problem.

Therefore problems of such behaviour is called decidable problems.

Undecidable Problem is the term coined when ~~a given lan~~ to solve a given problem there exist not turing machine to solve that problem. The language generally used ~~as~~ here is ~~undecidable~~ not decidable although it may or may not be partially decidable but since no turing machine exist is called undecidable problem.

~~(2) Turing Machine~~

(2) Halting Problem of Turing machine.

- The UTM or Universal Turing Machine or any turing machine suffers a simple drawback that it halts every time after giving a response or output.

The Turing machine after it needs to restart the calculation ~~unt~~ until a blank space is found in tape & it halts again. The output maybe false or true depending on the input but it halt after every response.

If an input string s when $s \notin L$ (Language not accepted by turing machine) then it will halt in a non-final state and if $s \in L$ it will always ~~halt~~ ~~halt in~~ halt in the final state giving a positive response

Unit 5

Q5 4

5a4 Partial function is defined as a function that is undefined at any certain point.

eg:

$f(x) = 1/x$ here the function $f(x)$ is undefined when $x=0$

Therefore $f(x)$ is a Partial function.

Initial function is defined as a function defined on the set of natural numbers with output on every point ~~ie~~ i.e. $N = 1,2,3,4,5, \ldots\ldots$ and so on

~~The initial funct~~ Any function can be termed as initial function if it follows the above criteria.

eg: $f(x,y) = x+y$

5b4 ① $f(x,y) = x*y$

for $y=0$

$$f(x,0) = x*0$$
$$= 0$$

↑

zero function ✓

for $y = y+1$

$$f(x,y+1) = x*(y+1)$$
$$= x*y + x$$
$$= f(x,y) + x$$ → not progressive

$$U_i^n \{f(x,y), f(x,y)+x\} = f(x,y)+x$$

$$U_2 = f(x+y)+x$$ → recursive funt

can also be defined as Projection function

∴ $f(x,y)$ here is both zero & ~~projection~~ recursive

hence

$f(x,y) = x*y$ is a Primitive recursive function.

② $f(x,y) = x^y$

for $y = 0$

$$f(x,0) = x^0 = 1$$ → not zero funct..

for $y = y+1$

$$f(x,y+1) = x^{y+1}$$
$$= x^y . x$$
$$= f(x,y) . x$$ → recursive function

$\therefore \cancel{f(x,y) = f(x,y) \cdot x}$

$\therefore f(x, y+1) = f(x,y) \cdot x$

& $U(1, x, x^2, x^3 \ldots) = f(x,y) = x^y$

F.c. can be written as projection function

$\therefore f(x,y) = x^y$ is a primitive recursive function
since it is
recursive & projectable.

5(c) Space & Time Complexity is the bargain one has to do between Space & Time so as to get either ~~more speed on the cost of more~~ faster calculation on the cost of more space or ~~less speed on the cost of less space~~ aquire less space on the cost of more time.

The Space & Time complexity Dicephora is called space & time complexity.

In the current scenario ~~is~~ space ~~is~~ or memory is cheaper & hence we prefer more speed

ज्ञानादेव तु कैवल्यम्

ज्ञानादेव तु कैवल्यम्

ज्ञानादेव तु कैवल्यम्

ज्ञानादेव तु कैवल्यम्