Tree *-A tree is a finite set of nodes with one specially designated node called the root and the remaining nodes are partitioned into n >= 0 disjoint sets T, and T, which are called the sub trees of the root.*

Binary Trees *-A binary tree is a finite set of nodes, with one special node called the root and the other nodes are partitioned into two disjoint sets; which are binary trees called its left and right subtrees. It is a tree where every node can have at the most two branches.*

Operations on Binary Search Tree -1)Create: This operation creates a tree with n nodes. 2)**Traverse**: This operation visits each and every node of the tree.3)**Search**: This operation searches for a specific element in the tree.4)**Insert**: This operation inserts a new element in the tree. In case of binary search tree, t new element will be inserted according to its value.5)**Delete**: This operation deletes a specific node from the tree.6)Counting total node.7)Counting leaf nodes.8)Copying a ree to another.9)Mirror of a tree.

Tree Traversal -1)Preorder Traversal -void preorder (NODE root) { NODE *temp = root; if (temp! NULL) { Printf("%d\t",temp->info); Preorder (temp->left); Preorder (temp->right);}} 2)**Inorder Traversal** -Void inorder (NODE * root) { NODE *temp = root; if (temp != NNULL { Inorder (temp->left); printf("%d\t", temp->info); inorder (temp->right);}} 3)**Postorder Traversal** -Void postorder (NODE root) { NODE *temp = root;if (temp != NULL) { postorder (temp->left); postorder (temp->right); printf("%d\t", temp->info); }}

Heap -A heap is a complete binary tree such that the key in the parent is greater than or equal to the keys in the children. This means that the largest element is in the root node (max heap). Heaps where the parent key is less than or equal to (S) the child keys are called min-heaps. Heaps are used to represent priority queues.

Static representation -1)Uses array.2)Separate memory is not required to store links to Children.3)Fixed size.4)Efficient only if the tree is perfect or complete.5)Any node can be directly accessed by calculating the index. **Dynamic representation** -1)Uses linked list.2)Separate memory is required to store links to children.3)Dynamic size.4)Efficient for any type of tree.5)A node cannot be accessed directly. To reach a node, we have to start from the root and use the links to that node.

Applications of Trees -1)Representing hierarchical information like directory structure of a computer.2)Representing dictionaries and symbol tables.3)In databases to store index and key values.4)Game trees.5)Heap sort.6)Expression representation and evaluation.7)Decision making (Decision trees).8)Text compression and encoding.

Heap Sort -Heapsort is one application of binary tree. This sorting method uses a special type of binary tree called "Heap". A max-heap is a full or complete binary tree such that each parent has a value greater than its children. This implies that the largest element is in the root node.

Greedy Strategy -The Greedy strategy is an important algorithm design strategy that is used in problem solving. It is used to solve optimization problems where the goal is to maximize (or minimize) an objective.The greedy strategy builds the solution by choosing one option at a time out of the many available options so as to get the optimal solution. As the name suggests, the greedy strategy always makes the choice that seems to be the best at that moment. It tries to find the locally optimal solution hoping that this will lead to a globally optimal solution.

Huffman Encoding -Encoding is the process of assigning a binary value to a symbol so that it can be stored and transmitted on a computer network.It is used in data compression, JPEG image format, ZIP lossless

Page No.:
Date: \ \

Q Design a PDA for accepting the set of strings over {a,b} with an equal number of a's & b's. The string should be accepted both by ① final state ② Empty stack.

$\delta(b,a/\epsilon)$
$\delta(b,Z_0/bZ_0)$ $\delta(a,b/\epsilon)$
$\delta(a,Z_0/aZ_0)$ $\delta(b,b/bb)$
$\delta(a,a/aa)$

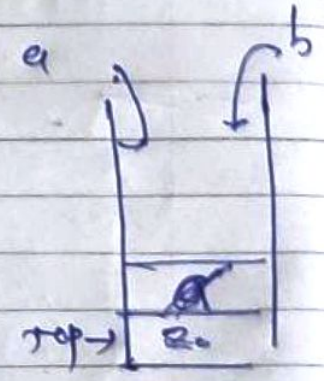
② → q_0

$\delta(\epsilon,Z_0/\epsilon)$ ← to make stack empty.

$\delta(b,a/\epsilon)$
$\delta(a,a/aa)$
$\delta(a,Z_0/aZ_0)$

① → q_0 —$\delta(\epsilon,Z_0/Z_0)$→ q_4

$\delta(b,Z_0/bZ_0)$
$\delta(b,b/bb)$
$\delta(a,b/\epsilon)$

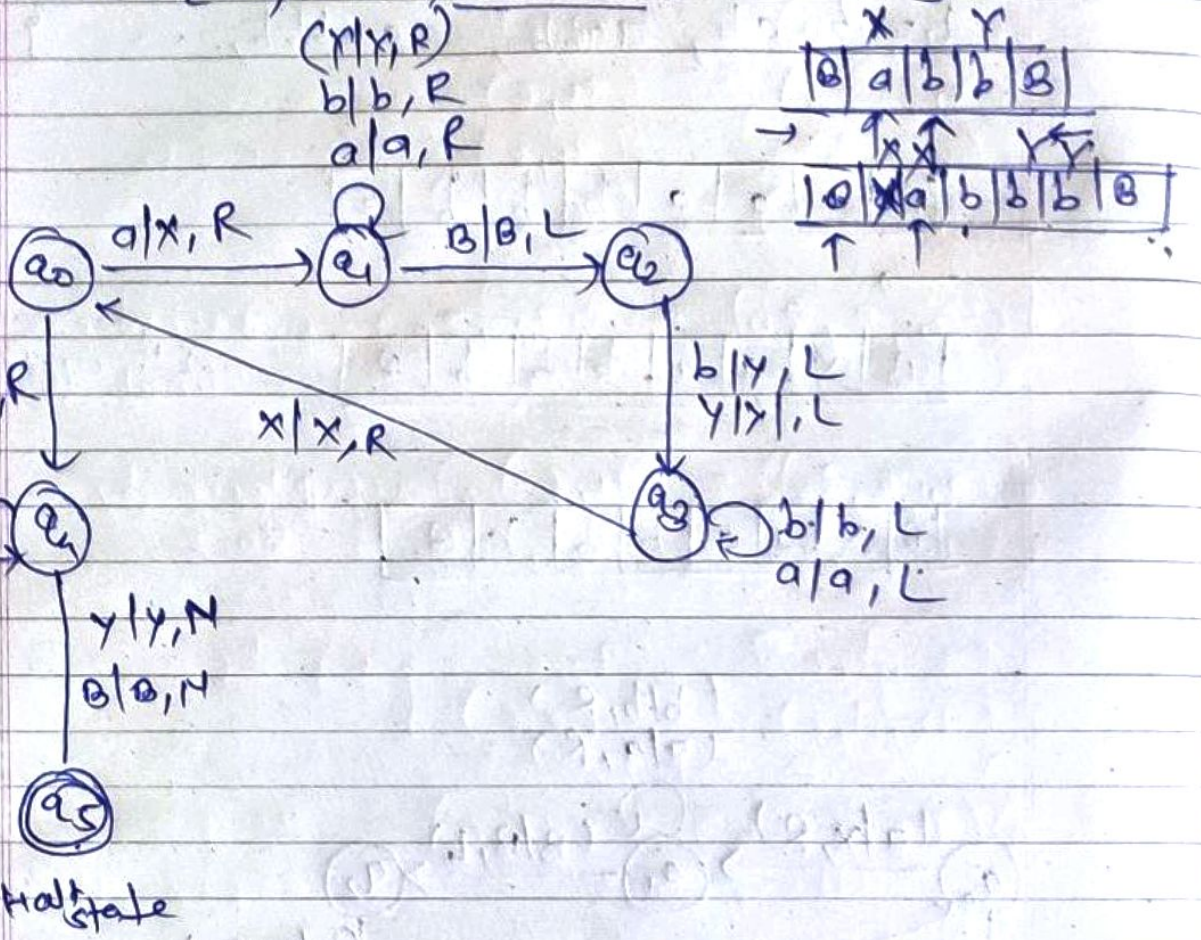


abababϵ

Page No.:
Date: \ \

Q. $L = \{ a^i b^j \mid i < j \}$

$L = \{ b, abb, aabbbb, \ldots \}$



Q. Construct TM for $L = \{ a^n b^{2n+1} \mid n \geq 1 \}$

$n=1$, $n=2$, $n=3$

$abbb$, $aabbbbb$, $aaabbbbbbb$, ...

In this leftmost a is written as X & two rightmost b's are written as BB

compression etc.*In Huffmans encoding method, the input is the characters and their frequencies (or probabilities and the output is a set of unique codewords, one for each* character.Huffman coding is a variable-length encoding mechanism. Huffman coding is a lossless data compression algorithm which is based on the Greedy strategy.

Advantages of Huffman Coding -1)Characters having higher frequency are assigned smaller codewords, thereby reducing the Number of bits. 2)Very efficient lossless compression method. 3)The codes follow the prefix property. Hence decoding is easy. **Disadvantages of Huffman Coding** -1)Frequencies of occurrences must be known in advance. 2)The codeword table must be shared with the receiver for decoding.3)Time consuming process.

Balanced tree -A balanced binary search tree is a tree that automatically keeps its height small (guaranteed to be logarithmic) for a sequence of insertions and deletions. A tree is perfectly height-balanced if the left and right subtrees of any node are the same height.

AVL Tree -An AVL tree is a binary search tree with balanced left and right subtrees. The tree is said to be balanced if the difference between the heights of left and right subtree.**Advantages** -1)The height of the AVL tree is always balanced. The height never grows beyond log N, where N is the total number of nodes in the tree.2)It gives better time complexity for operations like search, and insert when compared to simple Binary Search trees.3)AVL trees have self-balancing capabilities.**Disadvantages** -1)The implementation of AVL trees is more complex compared to binary search tree.2)Self-balancing requires frequent rotations.3)Deletion in AVL tree is very complicated and more time consuming.

Red Black tree -A red-black tree is a self-balancing binary search tree. Each node stores an extra bit representing “color” (“red” or “black”). These colors are used to ensure that the tree remains balanced during insertions and deletions.**Advantages** -1)Red black tree is useful when we need frequent insertion and deletion.2)Red-black trees are self-balancing, so operations are guaranteed to be O(log n).3)Less number of rotations as compared to AVL tree because it is not strictly balanced.**Disadvantages** -1)Implementation is complex.2)Insert and delete operations are more complex than simple binary search tree and AVL Tree.3)Requires 1 bit of color information in each node.

Splay tree -Splay Tree is defined as a self- adjusting binary search tree in which every operation on an element rearranges the tree so that the element is placed at the root position of the tree. This ensures that recently accessed elements are quick to access again.

B tree -1)All internal and leaf nodes have data pointers.2)Supports only random access. 3)Searching for some data is a slower process since data can be found on internal nodes as well as on the leaf nodes.4)No duplicate keys are maintained in the tree.5)Leaf nodes are not linked to each other. **B+ tree** -1)Only leaf nodes have data pointers.2)Supports random as well as sequential access. 3)Searching is comparatively faster as data can only be found on the leaf nnode.4)Duplicate keys are maintained in the tree and All keys are in leaf nodes.5)Leaf nodes are implemented as linked list.

Graph -A graph, G, is a collection of two sets V and E. V is a finite non-empty set of vertices (or sodes) and E is a finite non-empty set of edges (or arcs) connecting a pair of vertices. An edge is represented by two adjacent vertices. G is represented as G = (V,E).

Types of Graphs -1)Undirected graph: A graph is an undirected graph if the pairs of vertices that make up the edges are unordered pairs, i.c., an edge (Vi. Vj) is the same as (Vj). Vi). The graph G, shown above is an

Turing Machine

Turing machine M is 7-tuple machine

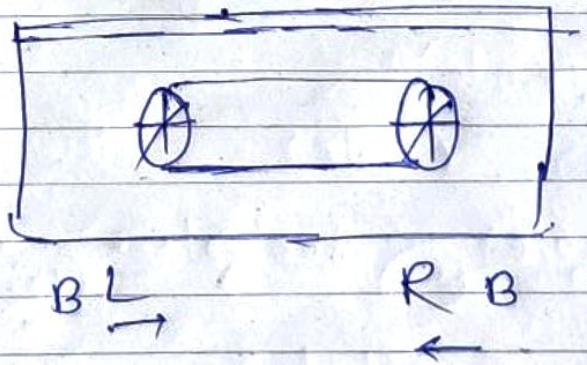
$$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$$

Q = is finite set of state
Σ = finite set of input alphabet not containing B
Γ = finite set of tape symbol Tape symbol include B

q_0 = Q is the initial ~~symbol~~/state
B = is a special symbol represting an empty cell
F = $\subseteq Q$ set of final states,

δ transition function
$Q \times \Gamma$ or $Q \times \Gamma (L, R, N)$.

$\delta(q_0, a) = (q, b, R)$,



undirected graph.2)**Directed graph**: In a directed graph, each edge is represented by a pair of ordered vertices, i.e., an edge has a specific direction. In such a case, edge (Vi, Vj) =(Vj, Vi).

Degree of Vertex -The degree of a vertex in an undirected graph is the total number of edges that vertex Connected to.

Indegree of a Vertex -If G is a directed graph, the indegree of a vertex is the number of edges for which it is head, i.c. the number of incoming edges to the vertex. A node whose indegree is 0, is called a source Node.

Outdegree of a Vertex -If G is a directed graph, the outdegree of a vertex is the number of edges for which it is the tail, Le.. the number of edges going out of it. A node whose outdegree is 0, is called a sink node.

Graph Representation -1)Adjacency Matrix -The adjacency matrix A of a graph G is a two dimensional array of n x n elements (where n is the numbers of vertices).2)**Adjacency List** -Adjacency list is a linked representation of a graph.In this representation, there is one list for each vertex in the graph. Thus, for a graph with 'n' vertices, there are 'n' lists. The list for vertex 'I' stores all vertices adjacent to vertex i.3)**Adjacency Multilist** -In the adjacency list representation, an undirected edge V, V, will be represented twice, once in the list of V, and once in the list for Vj.

Graph Traversals- 1)Depth First Search -This method is similar to the preorder tree traversal method. Starting from a particular vertex, we follow a path in the graph as deeply as we can go; marking the vertices in the path as 'visited". When the path ends, i.e., we reach a vertex which does not have any adjacent unvisited vertices, we proceed backwards (back track) to a vertex in the path which has an 'unvisited" adjacent vertex and proceed from this vertex. The process continues till all vertices have been visited.2)**Breadth First Search-** The Breadth First Search (BFS) is the level-wise traversal of a graph. In the Breadth first search method, we start at vertex v. We then visit all unvisited vertices 'w' which are adjacent to "v". For each vertex 'w', its unvisited adjacent vertices are visited and so on. The process continues till there are no more unvisited adjacent vertices left to visit.

Depth First Search-1)Traverses a graph depth-wise.2)Uses a stack for traversal.3)Exploration of a node is suspended As Soon As Another unexplored is found. 4)DFS is faster and requires lesser memory.5)The DFS traversal tree is narrow and long.**Breadth First Search** -1)Traverses a graph level-wise.2)Uses a queue for traversal.3)A node is fully explored before any other can begin.4)BFS is slower and requires more memory.5)The BFS traversal tree is wide and short.

Applications of Graphs-1)Representing maps and route calculation.2)Representing electrical and electronic circuits.3)Representing computer networks and routing algorithms.4)Representing projects as AOE and AOV networks.5)Scheduling of interdependent tasks or activities and calculation of Critical Path. 6)Modeling of chemical compounds and gene sequences.7)The hypertext linked document structure can be represented using graphs.

AOV Network – A directed graph in which the vertices represent activities or tasks and the edges represent the precedence is called an AOV network.

Topological Sort Definition: The process of converting a set of precedences represented by an AOV network into a linear list in which no later activity precedes an earlier one is called Topological Sorting.

$L = \{a^n b^m c^n d^m \mid n, m \geq 1\}$

aabbccccdddd

Let $L = \{a^n b^n c^m d^m \mid n, m \geq 1\}$ $\begin{bmatrix} n=1 \\ m \geq 1 \end{bmatrix}$

find PDA that accept language L

$\delta(q_0, a, Z_0) = (q_0, aZ_0)$ } push a's
$\delta(q_0, a, a) = (q_0, aa)$

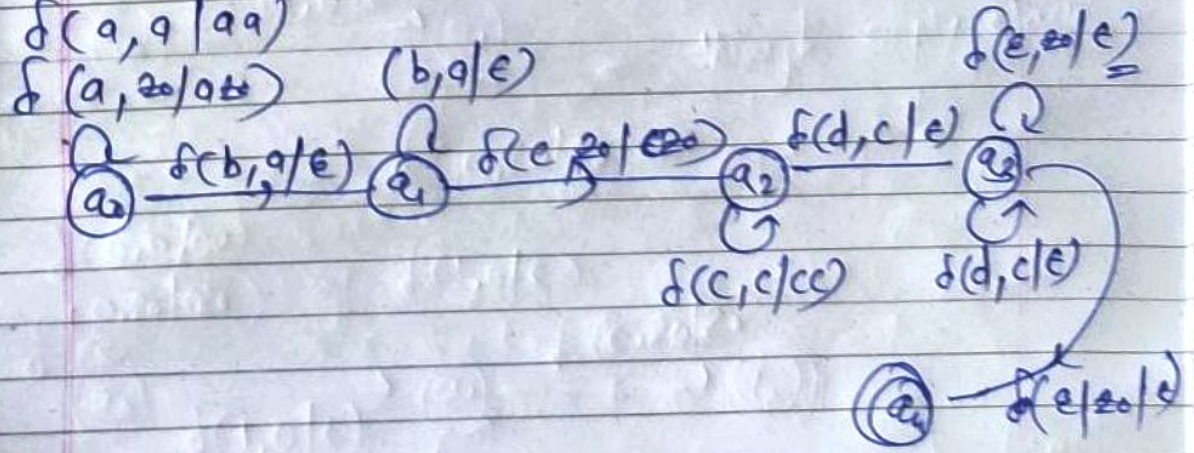
$\delta(q_0, b, a) = (q_1, \epsilon)$ } pop b's
$\delta(q_1, b, a) = (q_1, \epsilon)$

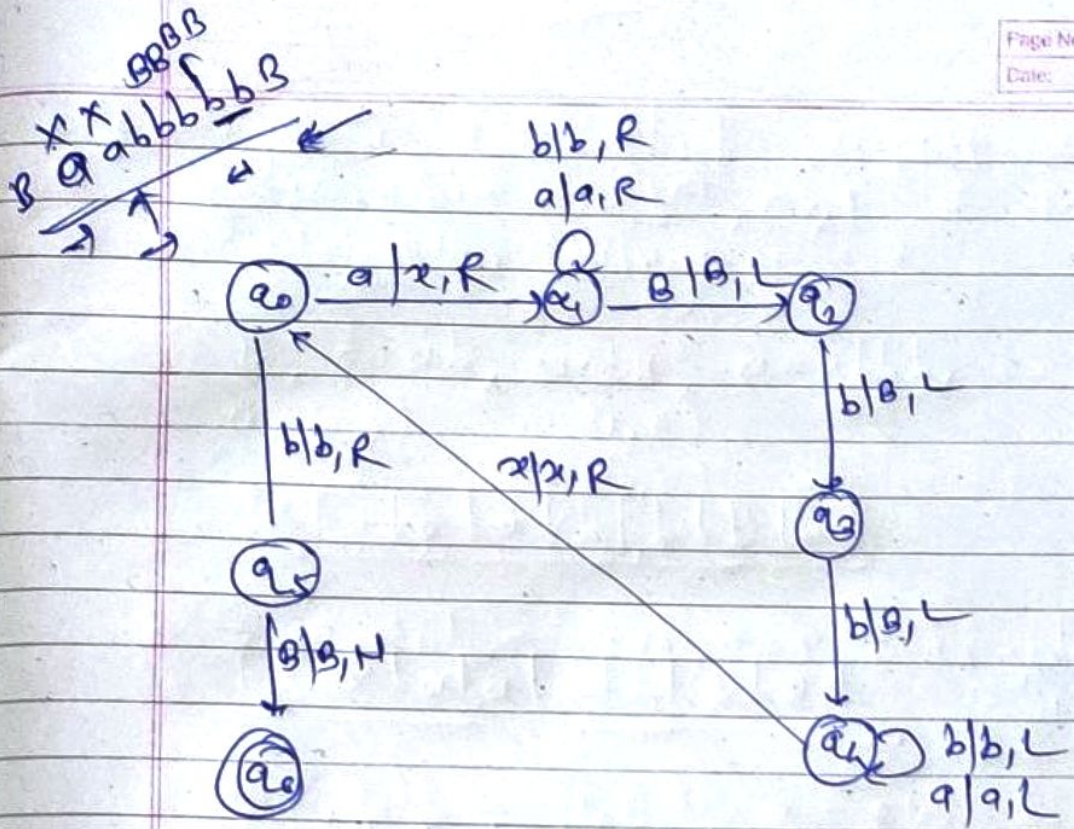
$\delta(q_1, c, Z_0) = (q_2, cZ_0)$ } push c's
$\delta(q_2, c, c) = (q_2, cc)$

$\delta(q_2, d, c) = (q_3, \epsilon)$ } pop d's
$\delta(q_3, d, c) = (q_3, \epsilon)$

$\delta(q_3, \epsilon, Z_0) = (q_3, \epsilon)$ acceptance by empty stack

$\delta(q_3, \epsilon, Z_0) = (q_4, \epsilon)$ acceptance by final state.





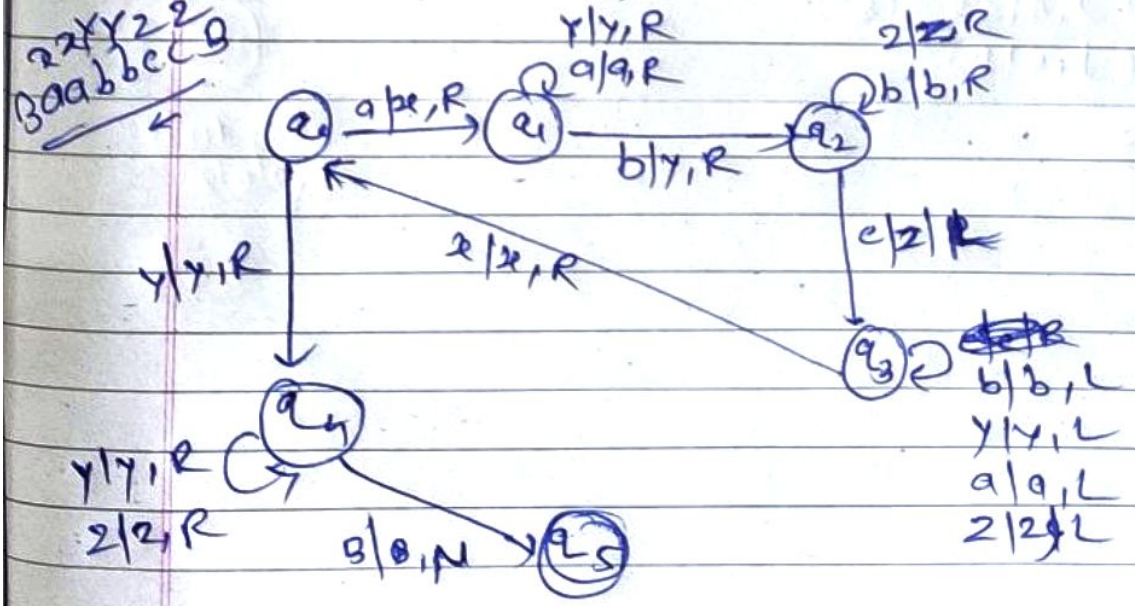
The TM, m is given by,

$m = (\{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}, \{a, b\}, \{a, b, x, B\}, \delta, q_0, B, \{q_6\})$

Q. Design TM which recognizes words of the form $a^n b^n c^n \mid n \geq 1$

n=1, n=2, n=3

abc, aabbcc, aaabbbccc, ...



Page No.:
Date: / /

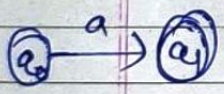
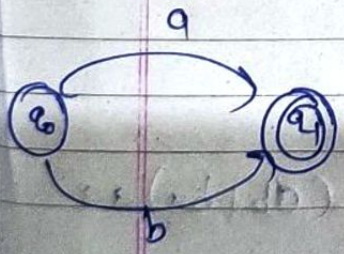
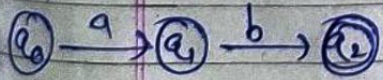
Regular Expression.

Regular Language:-

set of strings accepted by finite automata is known as regular language.

1. +, union operator
2. ·, concatenation operator
3. *, star or closure operator.

an expression written using set of operators (+, ·, *) & describing a regular language is known as Regular Expression

Automata	Language	R.E
	{ε}	ε
	{a}	a
	{a, b}	a + b
	{ab}	a·b | ab
	{ε, a, aa, aaa...}	a*

Page No.:
Date:

finite automata as output Device.

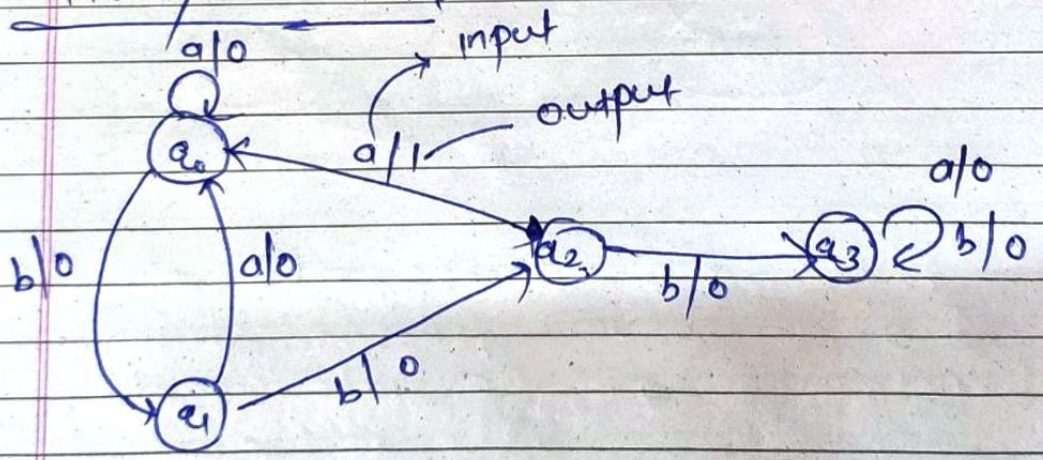
1) state transition function (δ)
2) output function (λ)

two types of automata with outputs:

1) Mealy machine:
output is associated with transition

2) Moore Machine:
output is associated with state.

Mealy machine:-



State transition func^n (δ) (STF)

δ	a	b
q_0	q_0	q_1
q_1	q_0	q_2
q_2	q_0	q_3
q_3	q_3	q_3

Prim's method -1)Starts by selecting a source vertex.2)Adds an edge having lowest weight and one vertex in the tree.3)An edge may be considered for selection multiple times.4)The partial solution is a tree.5)Binary or Fibonacci heaps are used for implementation.**Kruskal's method** -1)Starts by selecting the edge having lowest weight.2)Adds an edge having lowest weight and not forming a cycle.3)An edge is considered only once. It is accepted if it does not form a cycle, else rejected. 4)The partial solution is a forest.5)Disjoint sets are used for implementation.

Dynamic programming -This strategy is also used to solve optimization problems. Like divide-and-conquer method.Dynamic programming solves problems by combining the solutions of sub-problems.Applications of Dynamic programming -1)Matrix chain multiplication2(All pairs shortest path3)Longest common subsequence4)0/1 Knapsack.

Dijkstra's algorithm -1)Uses Greedy strategy.2)Single source shortest path algorithm finds the shortest path between a single vertex and all other vertices.3)Cannot be used if edges have negative weights.4)More space overhead.**Floyd Warshalls algorithm** -1)Uses Dynamic programming strategy.2)All pairs shortest path algorithm- finds the shortest path between all pairs of vertices.3)Can be used for negative edges.4)Less memory space required.

Hashing -Hashing is the process of converting a key into an index which is the location of the value in a Hash table.

Hash Table -The hash table is a sequential data structure that stores key-value pairs such that the key value determines the position where the value is stored. The basic operations on a hash table are Search, insert and delete.

Hash Function -A function f(X) that transforms a key X into a table index is called a hash function. It is a mathematical function which, when applied to a key, produces an integer (also called a hash code or hash address) which is used as an index for the key in the hash table. **Properties** -1)Efficiently computable: The time required for executing a hash function must be small,So that the hash address can be computed in a reasonable amount of time. 2)Deterministic: A hash procedure must be deterministic. This means that the same hash value must be generated for a given input value.3)Uniformity: A good hash function must map the keys as evenly as possible over its output Range.

Open addressing -1)Data structure used: Array.2)All keys are stored in the hash table.3)Number of keys cannot exceed the size of the hash table.4)Algorithm Complexity: Simple.5)Suitable for fiixed data size. **Chaining** -1)Data structure used: Array, Linked List.2)Keys are stored inside as well as outside the hash table.3)Number of keys can exceed the size of the hash table.4)Algorithm complexity: More complex.5)Suitable for Variable data size.

5. Turing Machine

Q. What is mean by TM?

 # Formal defination :-

Turing machine has 7 tuples

$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$

where Q is a set of finite states

Σ is a set of alphabets

Γ is a Input tape Alphabet

B is a Blank symbol

q_0 initial state

F is set of final state

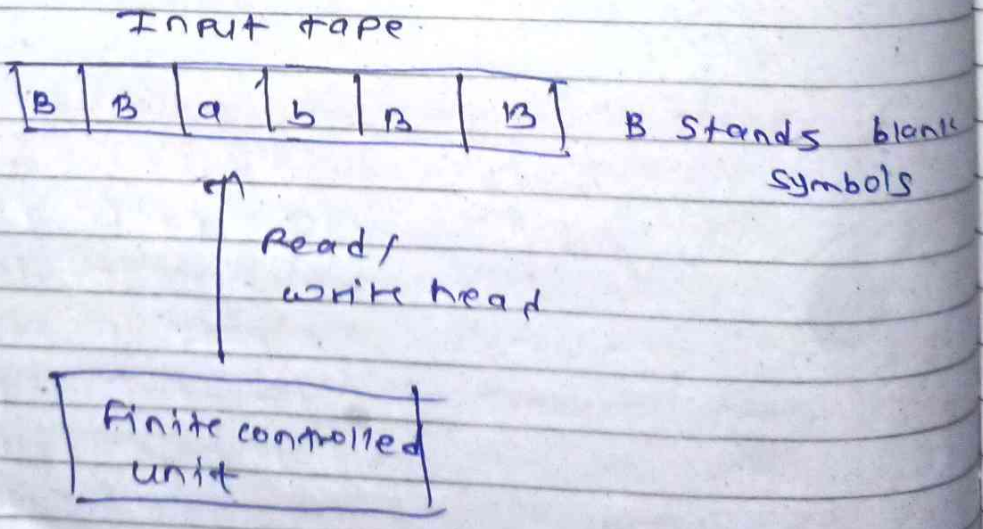
δ is transition function.

$Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$

$\delta(q_0, a) \rightarrow (q_2, X, R)$

$\delta(q_0, b) \rightarrow (q_1, Y, L)$

model.



Read/write head it is used to read/write data at a time one data.

Page No.:
Date:

* Design a TM which accepts all strings of the form $a^n b^n$, $n \geq 1$ & rejects all other strings

$L = \{ ab, aabb, aaabbb, \ldots \}$

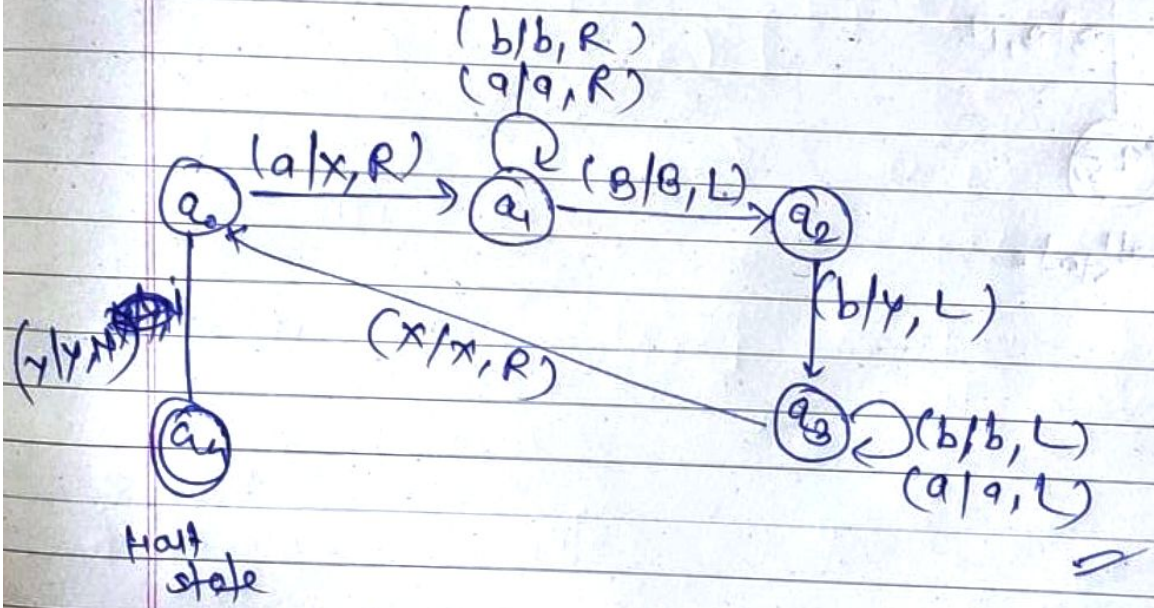
B	a	a	a	b	b	b	B

→

B	X	a	a	b	b	b	B

1st cycle ←

B	X	a	a	b	b	X	B



Theotical Computer Science

Page No.:
Date: / /

Finite Automata
Regular Languages
Context free grammer & Languages
PUSH DOWN Automata
Turing Machine

Mathematical Objects
1. sets
2. functions
3. Relations
4. Logic

→ set:
set is collection of things called elements or members

$N = \{1, 2, \ldots\}$

* Equality of sets

$A = B$

* The Empty set : NO elements in set

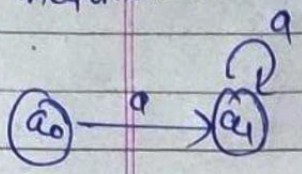
* subset:

$A = \{1, 2, 3, 4, 5, 6\}$
$B = \{3, 4, 5\}$

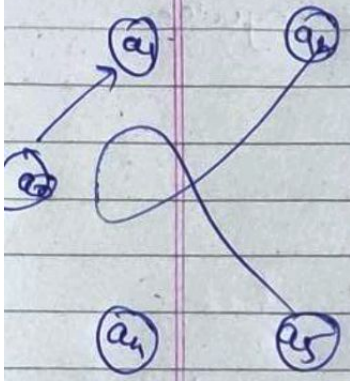
$B \subset A$.

Page No.:
Date:

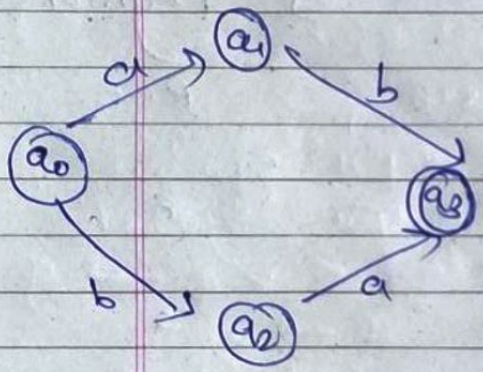
Automata Language R.E



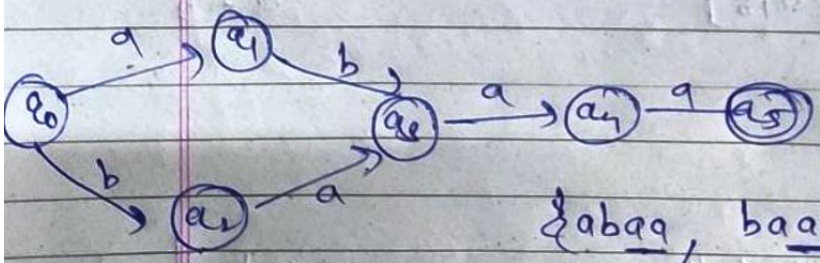
{a, aa, aaa, ...} aa* OR a+



{ab, ba}

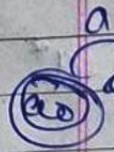


{ab, ba} ab + ba



(ab+ba)aa

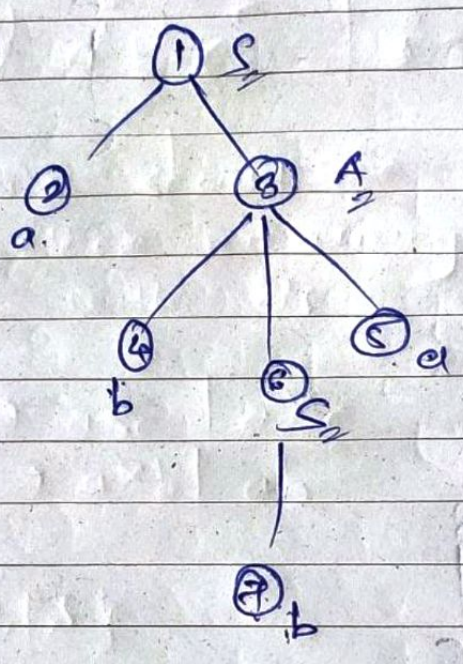
{abaa, baaa}



{ε, a, b, aa, ba, ab, bbb} (a+b)*

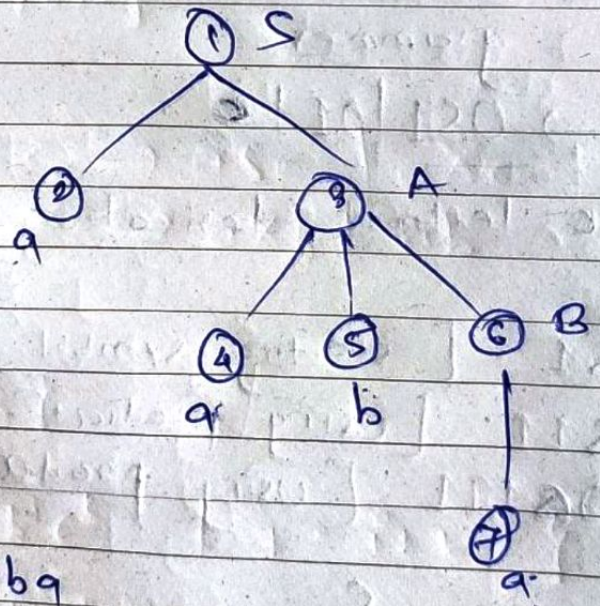
Page No.:
Date: \ \

Q2. what is the yield of derivation tree ?



ANS: abba

Q2



ANS: aaba

Page No.:
Date: \ \

- $L = \{ \epsilon, 0, 00, 000, \ldots \}$

  $R.E = 0^*$

- over $\{0,1\}$ ending with 1.

  ~~R.E =~~

  $L = \{1, 01, 11, 11 \ldots\}$

  $R.E = (0+1)^*.1$

- Starting with 0 & followed by any combination of 0,1. over $\{0,1\}$

  $R.E = 0.(0+1)^*$

- Set of strings over $\{a,b\}$ starting with 'a' & ending with 'b'

  $R.E = a.(a+b)^*.b$

- Design R.E for all strings starting & ending with 00 or 11 over $\{0,1\}$

  $L = \{00|11 \ldots 00|11, 0000, 0011, 1100, 1111, 00\ldots\}$

  $R.E = (00+11).(0+1)^*.(00+11)$

Page No.:
Date:

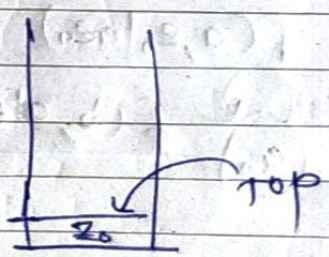
Push Down Automata

def

a pushdown automata M is defined as 7 tuples:

$$M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$$

Q = The set of states
Σ = Input alphabet
Γ = Stack symbols
δ = The transition function
q_0 = $q_0 \in Q$ is the initial state
f = $F \subseteq Q$, is the set of final states
z_0 = An initial stack symbol.



Empty stack

aabb

1. Read operⁿ with no i/p on stack.

$\delta(q_1, a, z_0) = (q_2, z_0)$

current state
i/p symbol
stack symbol
new stack symbol

Page No.:
Date: \ \

Output function:-

	a	b
q_0	0	0
q_1	0	0
q_2	1	0
q_3	0	0

formal defination of Mealy machine:-

Mealy machine is defined as,

$$M = \{Q, \Sigma, O, \delta, \lambda, q_0\}$$

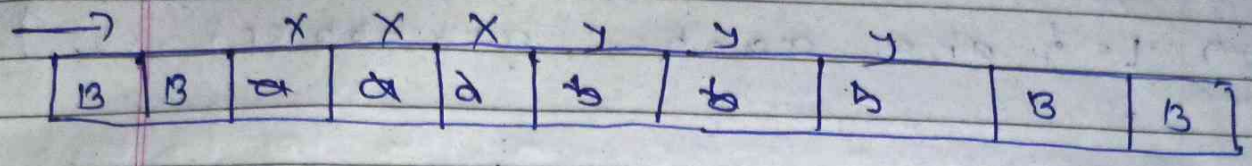
where,

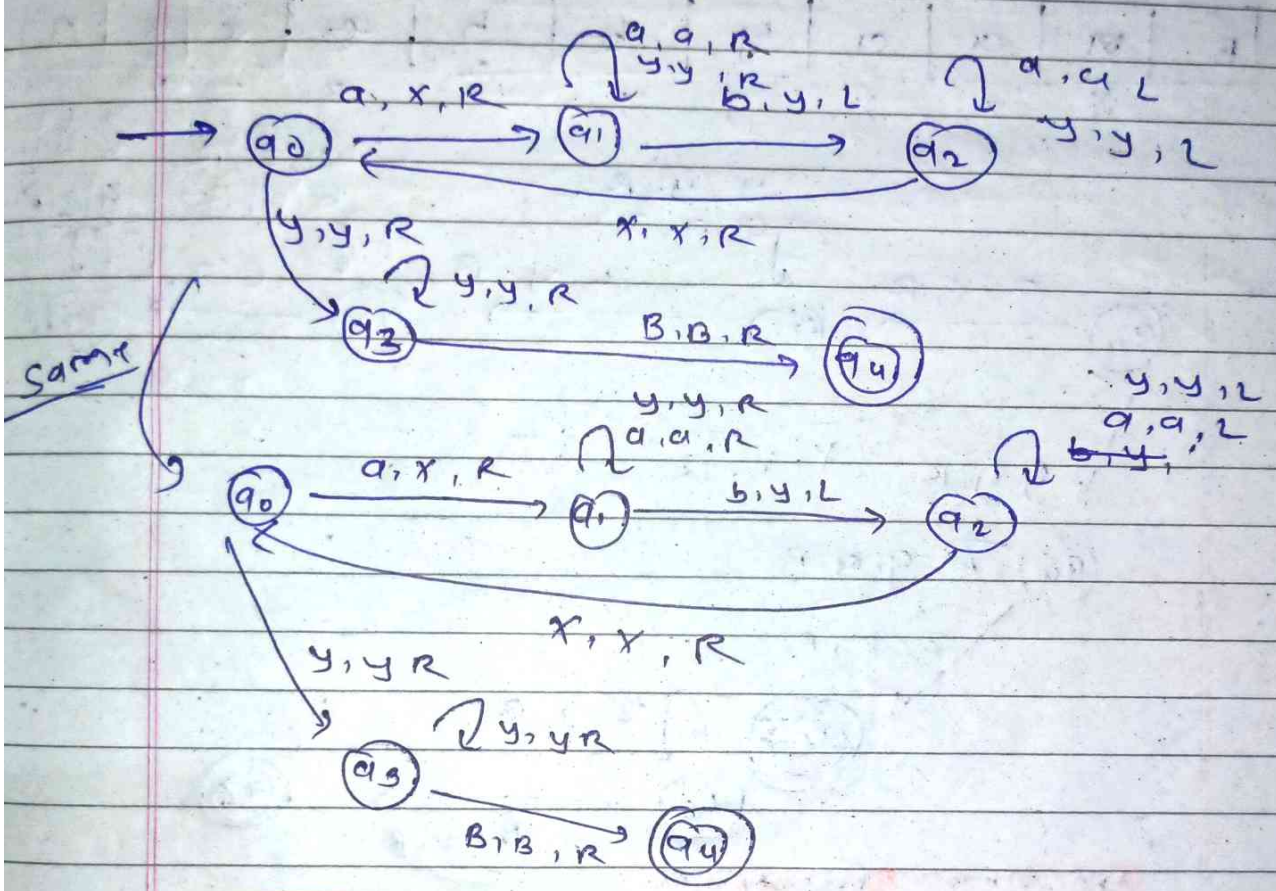
Q = a finite set of states
Σ = a finite set of input symbol
O = a finite set of output symbol
δ = a transition function $\Sigma \times Q \rightarrow Q$
λ = an output function $Q \rightarrow O$
q_0 = $q_0 \in Q$ is an initial state.

~~Push down~~

* For equal number of a's & b's both are equal numbers. construct Turing machine.

Q.) Design a Turing machine for $L = \{a^n b^n | n \geq 1\}$





Transition function

$(q_0, a) \longrightarrow (q_1, X, R)$

$(q_0, Y) \longrightarrow (q_3, Y, R)$

$(q_1, a) \longrightarrow (q_1, Y, R)$

$(q_1, b) \longrightarrow (q_2, Y, L)$

$(q_1, Y) \longrightarrow (q_1, Y, R)$

$(q_2, a) \longrightarrow (q_2, a, L)$

$(q_2, Y) \longrightarrow (q_2, Y, L)$

$(q_2, X) \longrightarrow (q_2, X, R)$

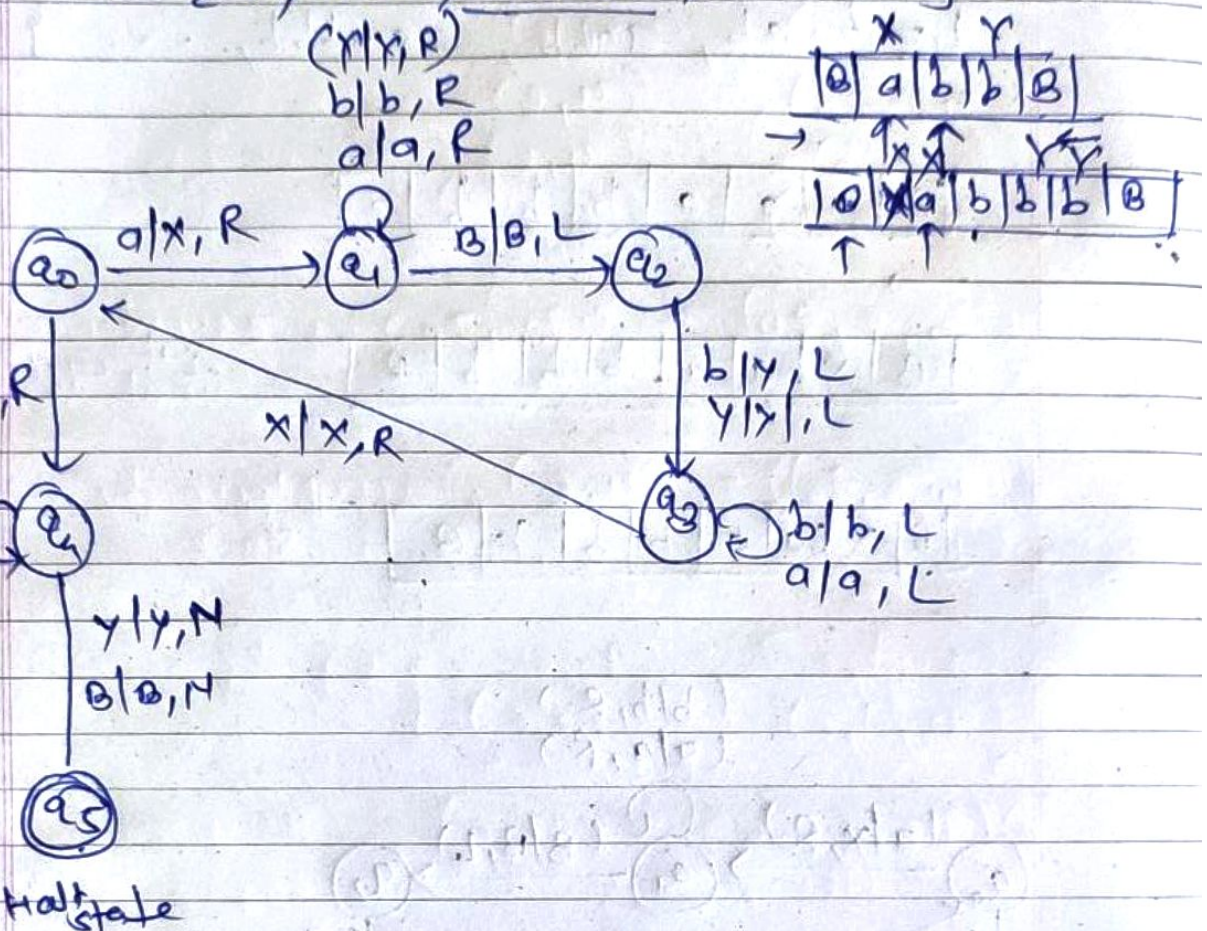
$(q_3, Y) \longrightarrow (q_3, Y, R)$

$(q_3, B) \longrightarrow (q_4, B, R)$

Page No.:
Date: \ \

Eg $L = \{ a^i b^j \mid i < j \}$

$L = \{ b, abb, aabbbb, \ldots \}$



Page No.:
Date: \ \

- $L = \{\epsilon, 0, 00, 000, \ldots\}$

  $R.E = 0^*$

- over $\{0,1\}$ ending with 1.

  ~~R.E =~~

  $L = \{1, 01, 11, 11 \ldots\}$

  $R.E = (0+1)^*.1$

- starting with 0 & followed by any combination of 0,1. over $\{0,1\}$

  $R.E = 0.(0+1)^*$

- set of strings over $\{a, b\}$ starting with 'a' & ending with 'b'

  $R.E = a.(a+b)^*.b$

Page No.:
Date: \ \

Languages:-

Finite Automata

state machine can be thought of as severly restricted model of a computer.

* **Alphabet**: It is finite non empty set of symbols. symbol Σ used for an alphabet

1. $\Sigma\{0, 1\}$, the binary alphabets
2. $\Sigma\{A, B, \ldots Z\}$, set of uppercase letters
3. $\Sigma\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ decimal alphabets
4. $\Sigma\{0, 1, 2\}$, ternary alphabet

* Strings (words).

It is finite sequence of symbols from alphabets

ex, 11011O, is a string from binary alphabet
5920 is a string from decimal alphabet
'STAMP' is a string from uppercase letters

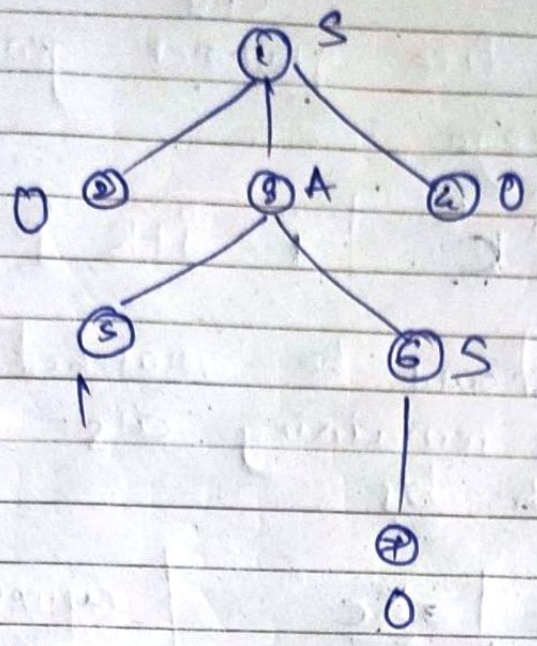
length of string:-

If $\omega = 01101$
then $|\omega| = 5$

length of an empty string is zero.
i.e $|\varepsilon| = 0$

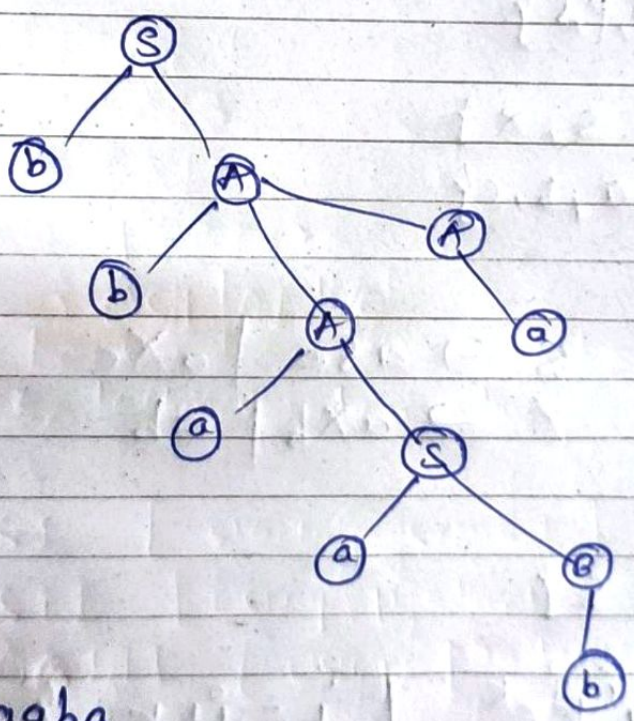
Page No.:
Date: \ \

Q2



ANS: 0100.

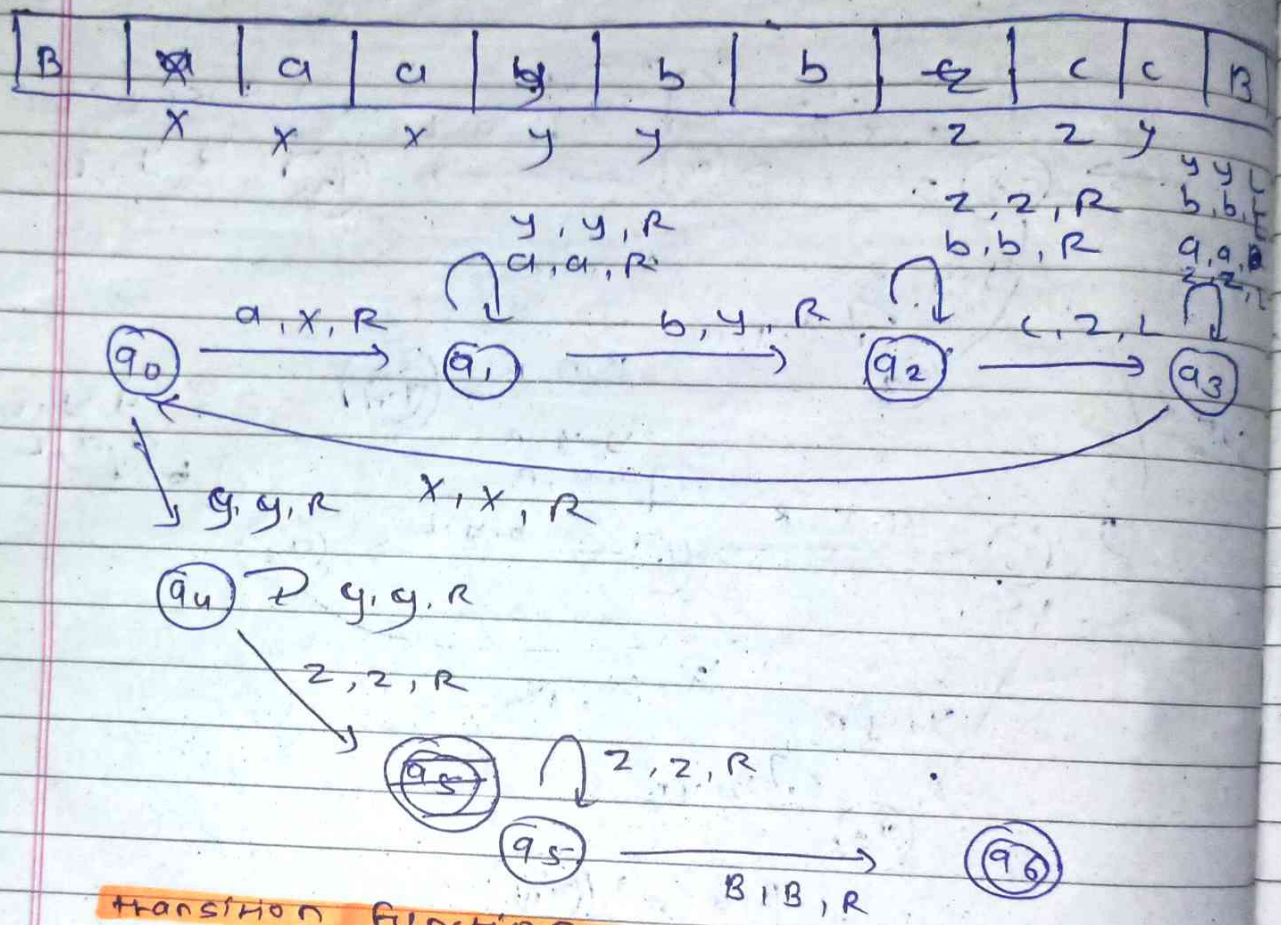
Q3



ANS: bbaaba

Q2) Design turing machine for $\{a^n b^n c^n \mid n>1, i y$

→ $L = \{abc, aabbcc, aaabbbccc, \ldots\}$



TRANSITION FUNCTION

$(q_0, a) = (q_1, X, R)$
$(q_0, y) = (q_4, y, R)$
$(q_1, a) = (q_1, a, R)$
$(q_1, y) = (q_1, y, R)$
$(q_1, b) = (q_1, y, R)$
$(q_2, b) = (q_2, b, R)$
$(q_2, c) = (q_3, z, L)$
$(q_2, z) = (q_2, z, R)$
$(q_3, a) = (q_3, a, L)$
$(q_3, b) = (q_3, b, L)$
$(q_3, y) = (q_3, y, L)$
$(q_3, z) = (q_3, z, L)$

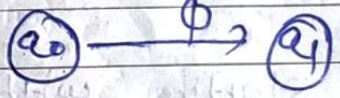
$ab^* + ab^* = R.E$

$ab^* + ab^* = ab^*$

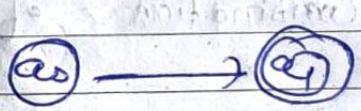
It represents set of strings over {a,b} where first symbol is 'a's & it is followed by 0 or more b's

ϵ

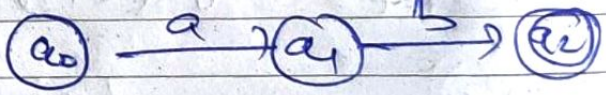
ϕ



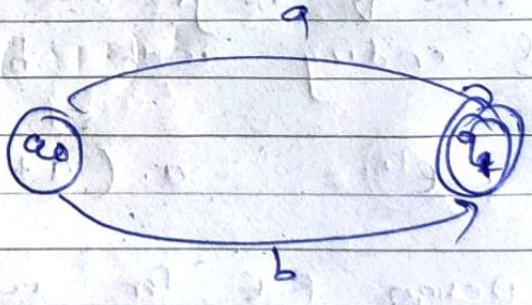
a



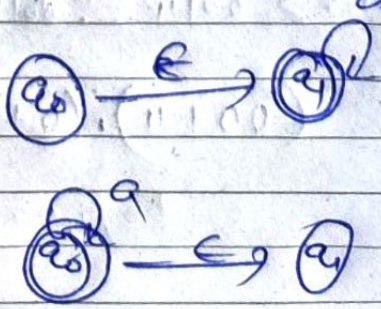
ab



a+b



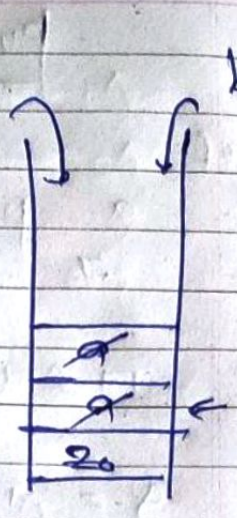
a^*



aabb

push pop

push a's pop b's



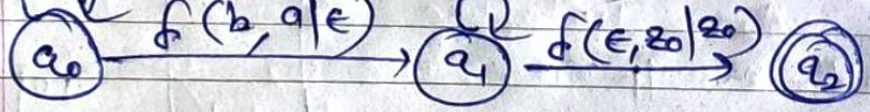
push $\delta(q_0, a, z_0) = (q_0, az_0)$
$\delta(q_0, a, a) = (q_0, aa)$

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

$\delta(q_1, \epsilon, z_0) = (q_1, \epsilon)$ ⇒ acceptance state.

aabb

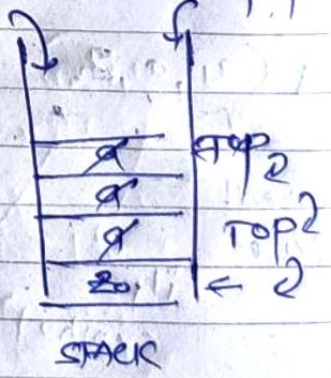
$\delta(q, a|aa)$
$\delta(q, z_0|az_0)$
$\delta(b, a|\epsilon)$



⇒

push pop
aaabbb ϵ ⇒

a push b pop



push $\delta(q_0, a, z_0) = (q_0, az_0)$
$\delta(q_0, a, a) = (q_0, aa)$

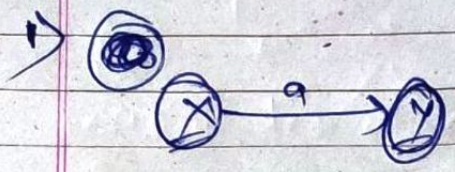
pop $\delta(q_0, b, a) = (q_1, \epsilon)$
$\delta(q_1, b, a) = (q_1, \epsilon)$

acceptance state $\delta(q_1, \epsilon, z_0) = (q_2, z_0)$ ⇒

Page No.:
Date: / /

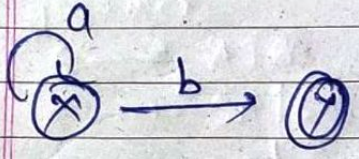
Context Free Grammer (CFG)

Equivalent Gram.



$X \rightarrow a$

N.T T



$X \rightarrow aX$

$X \rightarrow b$

aab

$X \rightarrow aX$

$X \rightarrow \epsilon$

$X \rightarrow aX$

aaX

aab

Ex $L = \{0^n 1^n \mid n \geq 1\}$

$L = \{01, 0011, 000111, \ldots\}$ 00001111

$X \rightarrow 01$ — (1)

$X \rightarrow 0X1$ — (2)

0011

$X \rightarrow 0X1$ ∵

$X \rightarrow 0011$

00001111

$\Rightarrow 0X1$ ∵

$00X11$ ∵ using

$000X111$ ∵

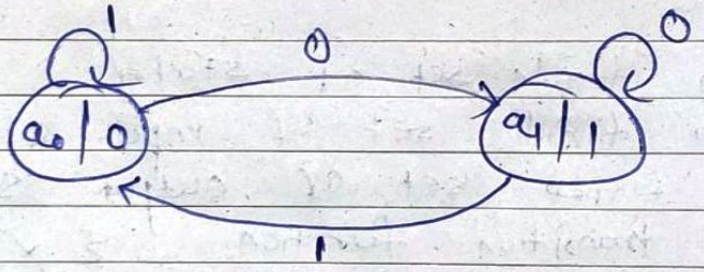
$0000X1111$ ∵ (3)

Page No.:
Date: \ \

Q.1 * Design Mealy & Moore machine for 1's complement of binary number.

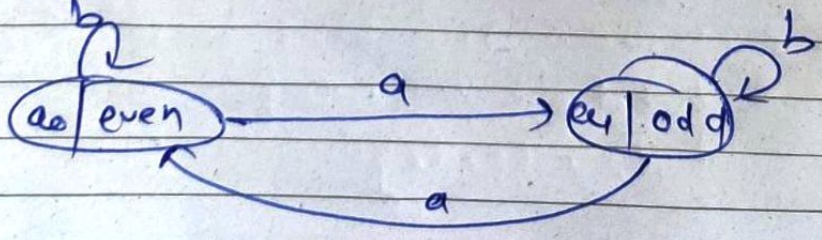
Mealy Machine:- (with transition)

Moore Machine:- (with state)



Q. 2 to take input from {a, b}, which outputs even or odd according to number of a's encountered is even or odd.

Moore:-



$$\left\{ \begin{array}{l} NT \rightarrow NT.NT \\ NT \rightarrow T \end{array} \right\}$$

Page No.:
Date:

Normal forms of CFG:-

1) Chomsky Normal form (CNF)
2) Greibach Normal form (GNF).

CNF (Chomsky Normal form):-

- a context free grammer (CFG) without ε production is said to be in CNF if every production is of the form

1. $A \rightarrow BC$, where $A, B, C \in V$
2. $A \rightarrow a$, where $A \in V$ & $a \in T$.

Q) find the CNF equivalent to

$$\left\{ \begin{array}{l} S \rightarrow aAbB \\ A \rightarrow aA \\ B \rightarrow bB \mid b \end{array} \right\}$$

⇒

$$\boxed{\begin{array}{l} NT \rightarrow NT.NT \\ NT \rightarrow T \end{array}}$$

New productions:-

$X \rightarrow a$

$Y \rightarrow b$

$S \rightarrow XA \mid YB$

$A \rightarrow XA$

$B \rightarrow YB \mid b$

$P \rightarrow XA$

$Q \rightarrow YB$

$(q_4, y) = (q_4, y, R)$

$(q_4, z) = (q_5, z, R)$

$(q_5, z) = (q_5, z, R)$

$(q_5, B) = (q_6, B, R)$

Page No.:
Date: \ \

Σ, exponential notation to express the set of all strings of certain length

Σ^k = set of strings of length k,

for ex,

if $\Sigma = \{0, 1\}$

if $\Sigma^1 = \{0, 1\}$

$\Sigma^2 = \{00, 01, 10, 11\}$

$\Sigma^3 = \{000, 001, 010, 100, 110, 101, 011, 111\}$

$\Sigma^0 = \{\epsilon\}$

Reversal of string ω is denoted by ω^R

if $\omega = xyz$

then $\omega^R = zyx$

$\omega = abb$ $\{a, b\}$

then $\omega^0 = \{\epsilon\}$

$\omega^1 = \{abb\}$

$\omega^2 = \{\omega, \omega\} = \{abbabb\}$

$\omega^3 = \{abbabbabb\}$

Page No.:
Date: \ \

Q* Find content free (CFG) grammer generating below language.

$L_1 = \{ a^i b^j c^k \mid i = j+k \}$

outer a's should be matched with c's then the remaining a's should be matched with b's

$S \to aSc \mid aXc$
$X \to aXb \mid ab$

| CAPITAL (N.T)
| small (T)

$G = \{V, T, P, S\}$

$V = \{ S, X \}$
$T = \{ a, b, c \}$

$P = \{$
$S \to aSc \mid aXc$
$X \to aXb \mid ab$
$\}$

$S = \{ S \}$

$L_2 = \{ a^i b^j c^k \mid j = i+k \}$

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

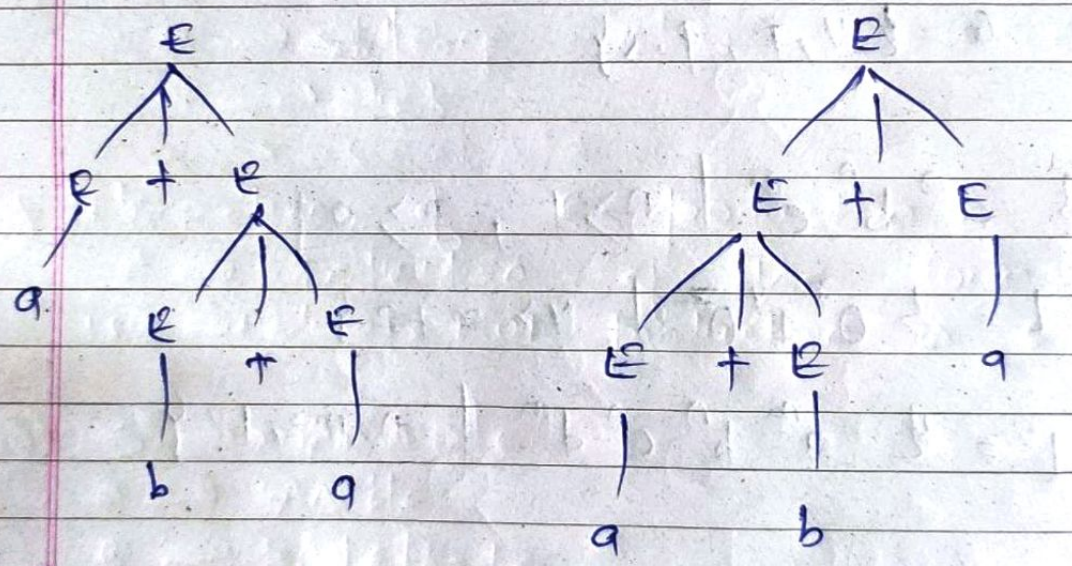
$S \to XY$
$X \to aXb \mid ab$
$Y \to bYc \mid bc$

* Ambiguous Grammar:-

A grammer is said to be ambiguous if the language generated by grammer contains some string that has two parse trees

$E \rightarrow E+E \mid a \mid b$

a string $a+b+a$ is generated using the above grammer.



=>

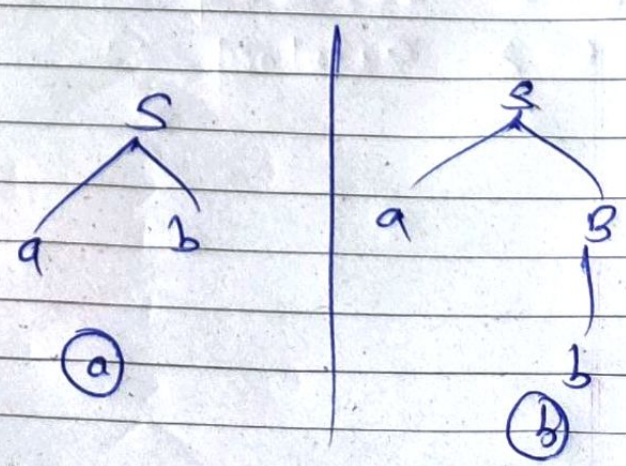
$S \rightarrow aB \mid ab$

$A \rightarrow aAB \mid a$

$B \rightarrow ABb \mid b$

String : ab

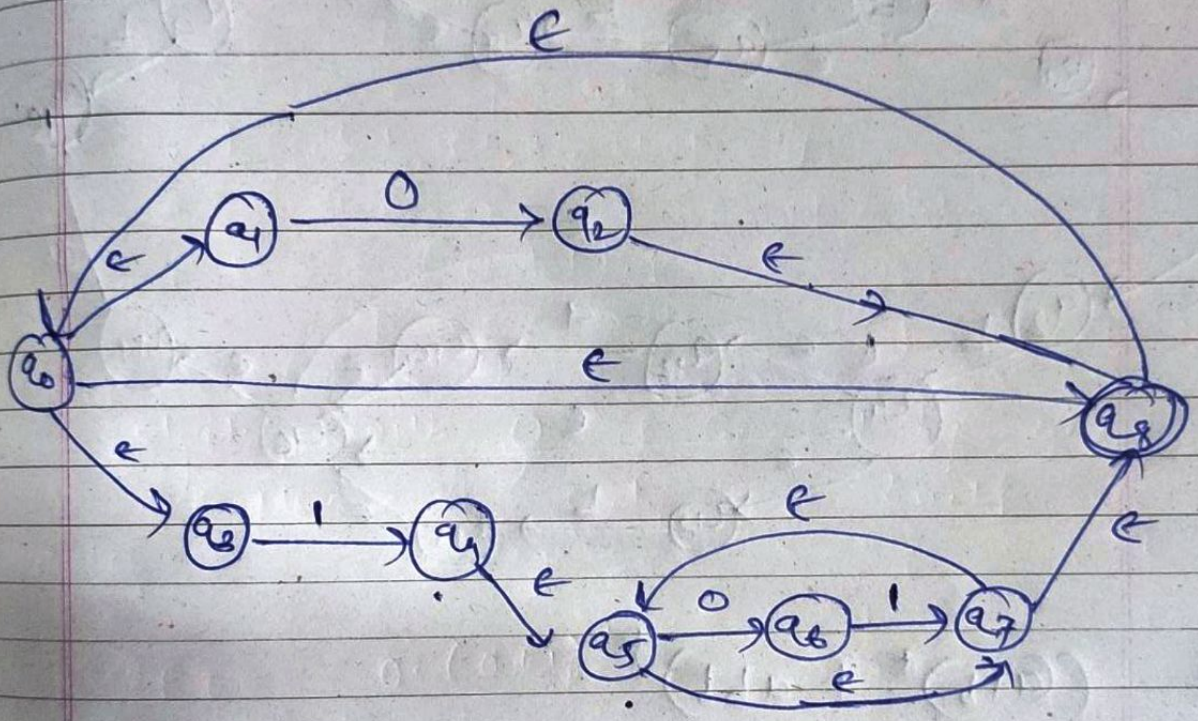
=>



Page No.:
Date: \ \

Q Construct a finite automata for the Regular expression

$$R = (0 + 1.(01)^*)^*$$



Page No.:
Date: \ \

a context free grammer G is quadruple (V, T, P, S).

V - is set of variables
T is set of terminals
P is set of productions
S is special variable called the starting symbol.

$S \rightarrow A1B$ — ①
$A \rightarrow 0A \mid \epsilon$ — ②
$B \rightarrow 0B \mid 1B \mid \epsilon$ — ③

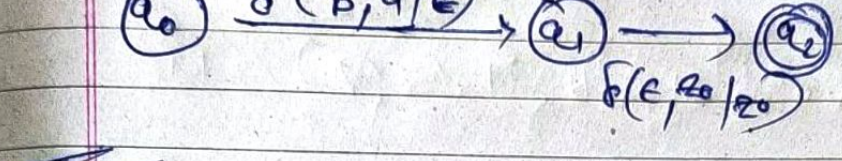
where, $G = (V, T, P, S)$
$V = \{S, A, B\}$.
$P = \{$ production, 1, 2, 3 $\}$
$S = \{S\}$
$T = \{0, 1\}$.

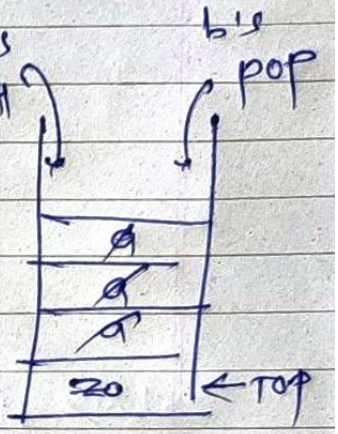
generate the string 00101

$S \rightarrow A1B$ ∵ starting production
$\rightarrow 0A1B$ ∵ using product^n $A \rightarrow 0A$
$\rightarrow 00A1B$ ∵ using product^n $A \rightarrow 0A$
$\rightarrow 001B$ ∵ using product^n $A \rightarrow \epsilon$
$\rightarrow 0010B$ ∵ using product^n $B \rightarrow 0B$
$\rightarrow 00101B$ ∵ using product^n $B \rightarrow 1B$
$\rightarrow 00101$ ∵ using product^n $B \rightarrow \epsilon$

aaabbbϵ

δ(a, a/aa)
δ(a, Z0/aZ0)
δ(b, a/ϵ)



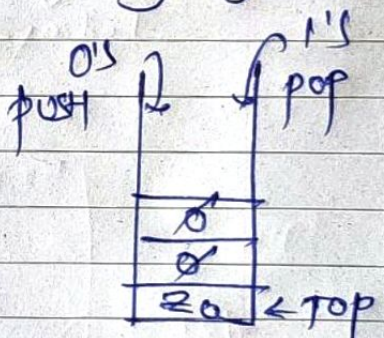


Q Give a PDA to accept the language

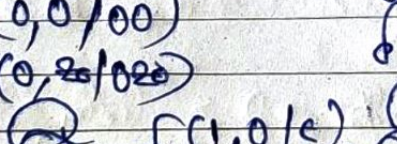
$L = \{ 0^n 1^m \mid n \le m \}$

1) Through empty stack
2) Through final state

{00111ϵ,

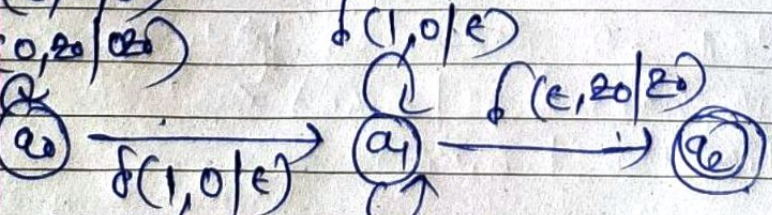


δ(0, 0/00)
δ(0, Z0/0Z0)
δ(1, 0/ϵ)
δ(1, Z0/Z0)
δ(ϵ, Z0/Z0)



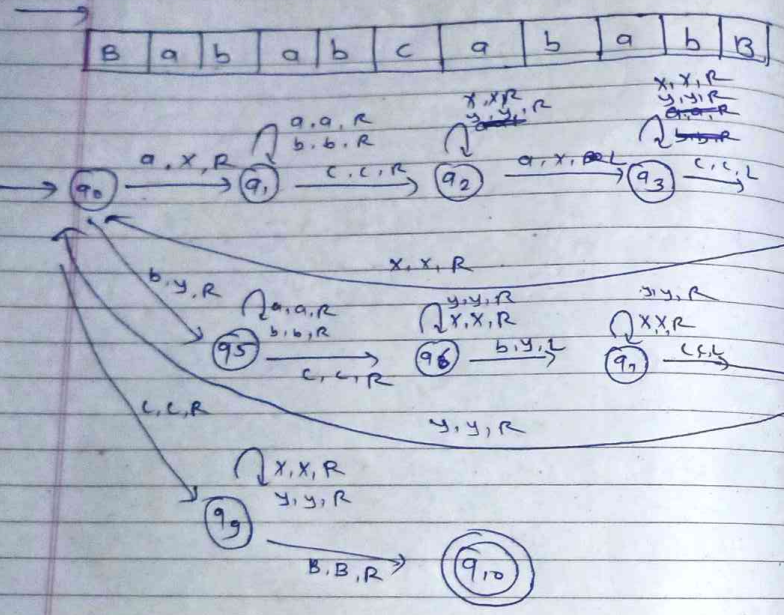
= through empty stack

δ(0, 0/00)
δ(0, Z0/0Z0)
δ(1, 0/ϵ)
δ(ϵ, Z0/Z0)
δ(1, Z0/Z0)



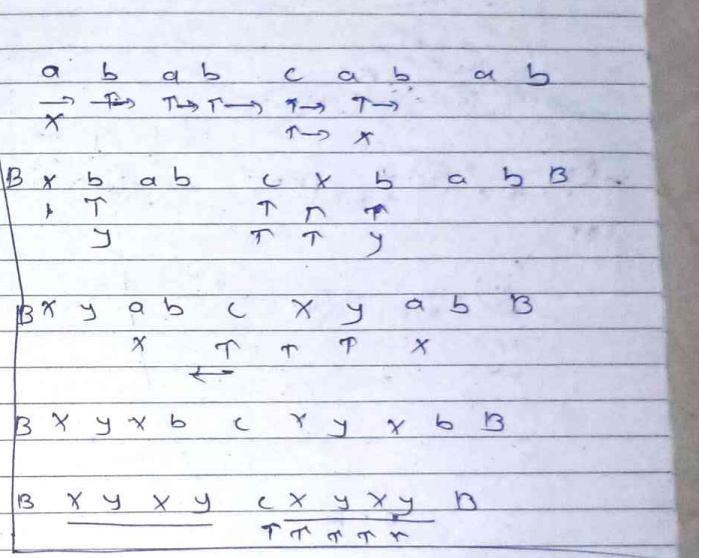
= through final state

Q3] construct turing machine for
$wcw \mid w \in \{a,b\}^*$



Transition Function

$(q_0, a) = (q_1, X, R)$	$(q_3, X) = (q_3, X, R)$
$(q_0, b) = (q_5, y, R)$	$(q_3, y) = (q_3, y, R)$
$(q_0, c) = (q_4, c, R)$	$(q_3, c) = (q_4, c, L)$
$(q_1, a) = (q_1, a, R)$	$(q_4, a) = (q_4, a, R)$
$(q_1, b) = (q_1, b, R)$	$(q_4, X) = (q_0, X, R)$
$(q_1, c) = (q_2, c, R)$	$(q_4, b) = (q_4, b, R)$
$(q_2, a) = (q_3, X, L)$	$(q_5, a) = (q_5, a, R)$
$(q_2, X) = (q_2, X, R)$	$(q_5, b) = (q_5, b, R)$
$(q_2, y) = (q_2, y, R)$	$(q_5, c) = (q_6, c, R)$



$(q_6, X) = (q_6, X, R)$	$(q_9, X, y) = (q_9, X, R)$
$(q_6, y) = (q_6, y, R)$	$(q_9, y) = (q_9, y, R)$
$(q_6, b) = (q_7, y, L)$	$(q_9, B) = (q_{10}, B, R)$
$(q_7, X) = (q_7, X, R)$	
$(q_7, y) = (q_7, y, R)$	
$(q_7, c) = (q_8, c, L)$	
$(q_8, a) = (q_8, a, R)$	
$(q_8, b) = (q_8, b, R)$	
$(q_8, y) = (q_8, b, R)$	

Page No.:
Date: / /

$NT \rightarrow NT.NT$
$NT \rightarrow T$

$S \rightarrow PQ$
$P \rightarrow XA$
$Q \rightarrow YB$
$A \rightarrow XA$
$B \rightarrow YB \mid b$
$X \rightarrow a$
$Y \rightarrow b$

Q $S \rightarrow aAbB \mid BbS$
$B \rightarrow aAbA \mid aAB \mid b$
$A \rightarrow aB \mid aBb \mid a$

Introducing new productions:

$X \rightarrow a$
$Y \rightarrow b$

after applying,

$S \rightarrow XAYB \mid BYS$
$B \rightarrow XAYB \mid XAB \mid b$
$A \rightarrow XB \mid XBY \mid a$

$P \rightarrow XA$
$Q \rightarrow YB$
$R \rightarrow BY$

$S \rightarrow PQ \mid RS$
$B \rightarrow PQ \mid PB \mid b$
$A \rightarrow XB \mid XR \mid a$
$P \rightarrow XA$
$Q \rightarrow YB$
$R \rightarrow BY$
$X \rightarrow a \mid Y \rightarrow b$

0110

Construct CFG which accepts set of palindrome over $\{0,1\}^*$

$S \to 0X1 \mid 0 \mid 1 \mid \epsilon$

$S \to 0S0$
$S \to 1S1$
$S \to 0 \mid 1 \mid \epsilon$

$S \to 0S0 \mid 1S1 \mid 0 \mid 1 \mid \epsilon$

$\Rightarrow G = \{V, T, P, S\}$

$L = \{ a^n b^n c^m \mid n \geq 1, m \geq 0 \}$

$L = \{ 0^x 1^y 0^z \mid y > x+2 \}$

$L = \{ a^n b^n c^n d^m \mid n, m \geq 1 \}$

Page No.:
Date: \ \

Languages:- a subset of strings over an alphabet Σ is a language.
any subset of Σ^* is a language

if Σ is an alphabet & $L \subseteq \Sigma^*$, then L is a language over Σ

ex. $L_1 = \{w \in \{0,1\}^* \mid w$ has an equal number of 0's & 1's$\}$

i.e. L_1 is a language over alphabet $\{0,1\}$ equal no. of 0's & 1's

$L_1 = \{\epsilon, 01, 10, 0011, 1100, 0101, 1010, \ldots\}$

ex. a language of all strings where the string represents a binary number. (0,1)

$L_2 = \{0, 1, 00, 01, 10, 11, 0001, \ldots\}$

ex The set of strings of 0's & 1's ending with 11

Page No.:
Date: \ \

* Simplification of CFG:-

1) Eliminating Useless symbols
2) Eliminating ε-productions
3) Eliminating Unit productions

* Useless Symbols:-

symbols or productions which are not useful for derivation of strings.

1) Non-generating symbols / production
2) Non-rechable symbols / production.

$S \rightarrow Aa \mid Bb \mid a \mid b$
$A \rightarrow Aa \mid a$
$B \rightarrow bB$

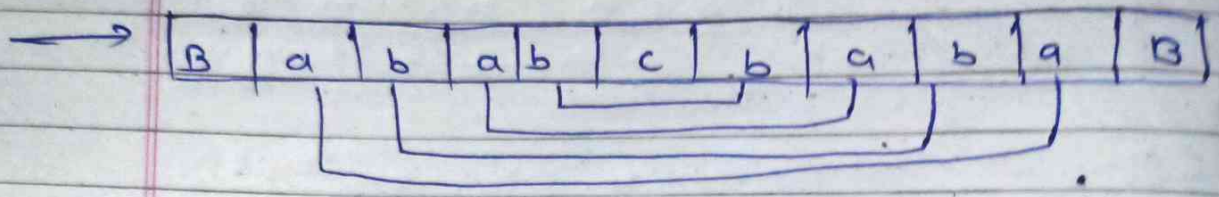
∴ B is non generating symbol.

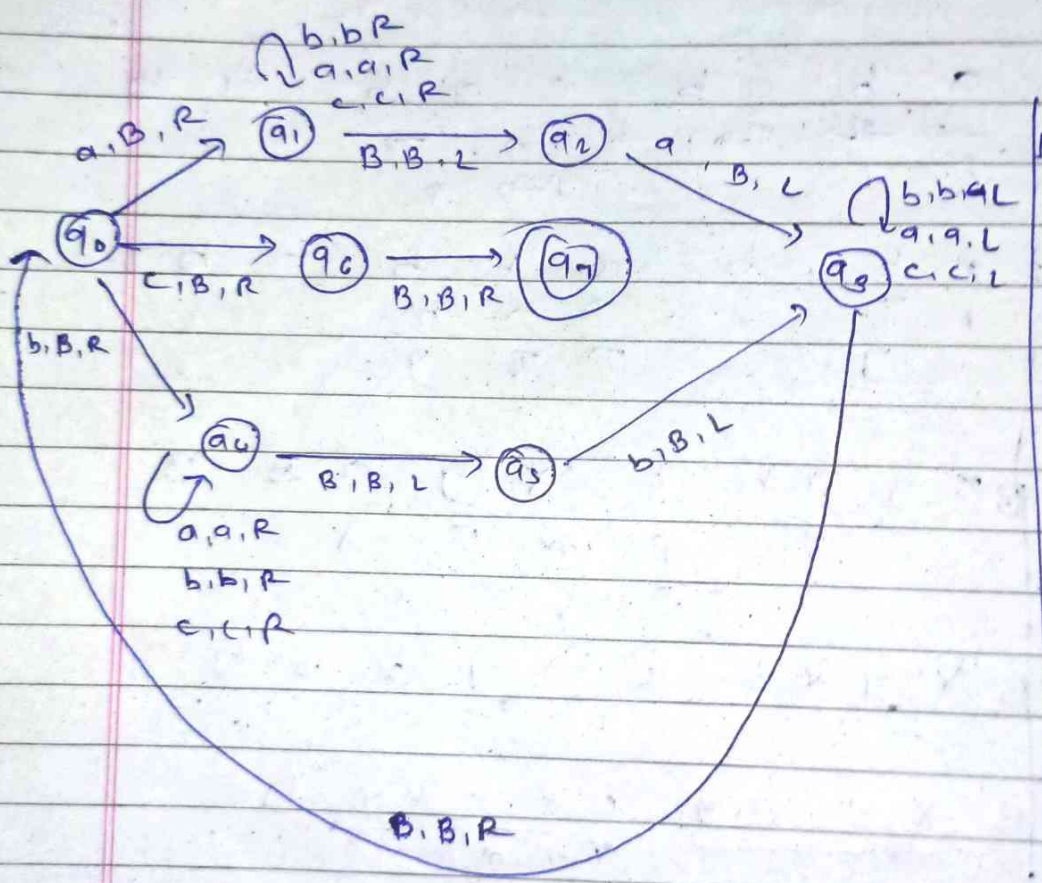
$B \rightarrow bB$ ∴ we can not generate string of terminal using $B \rightarrow bB$

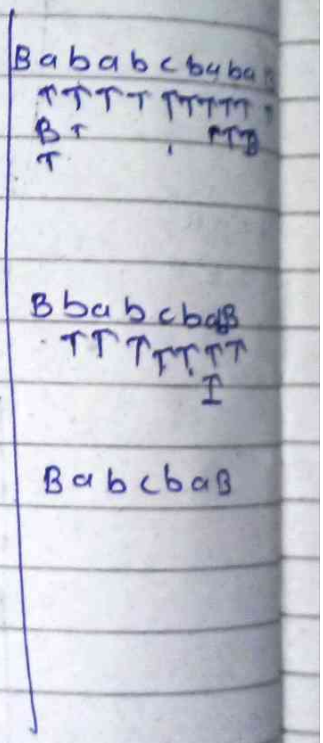
$S \rightarrow Aa \mid a \mid b$
$A \rightarrow Aa \mid a$

∴ deleting every production of useless symbol B

Q4) Design Turing machine for
$WcW^R \mid W \in \{a,b\}^*$
odd palindrome with symbol 'c'.

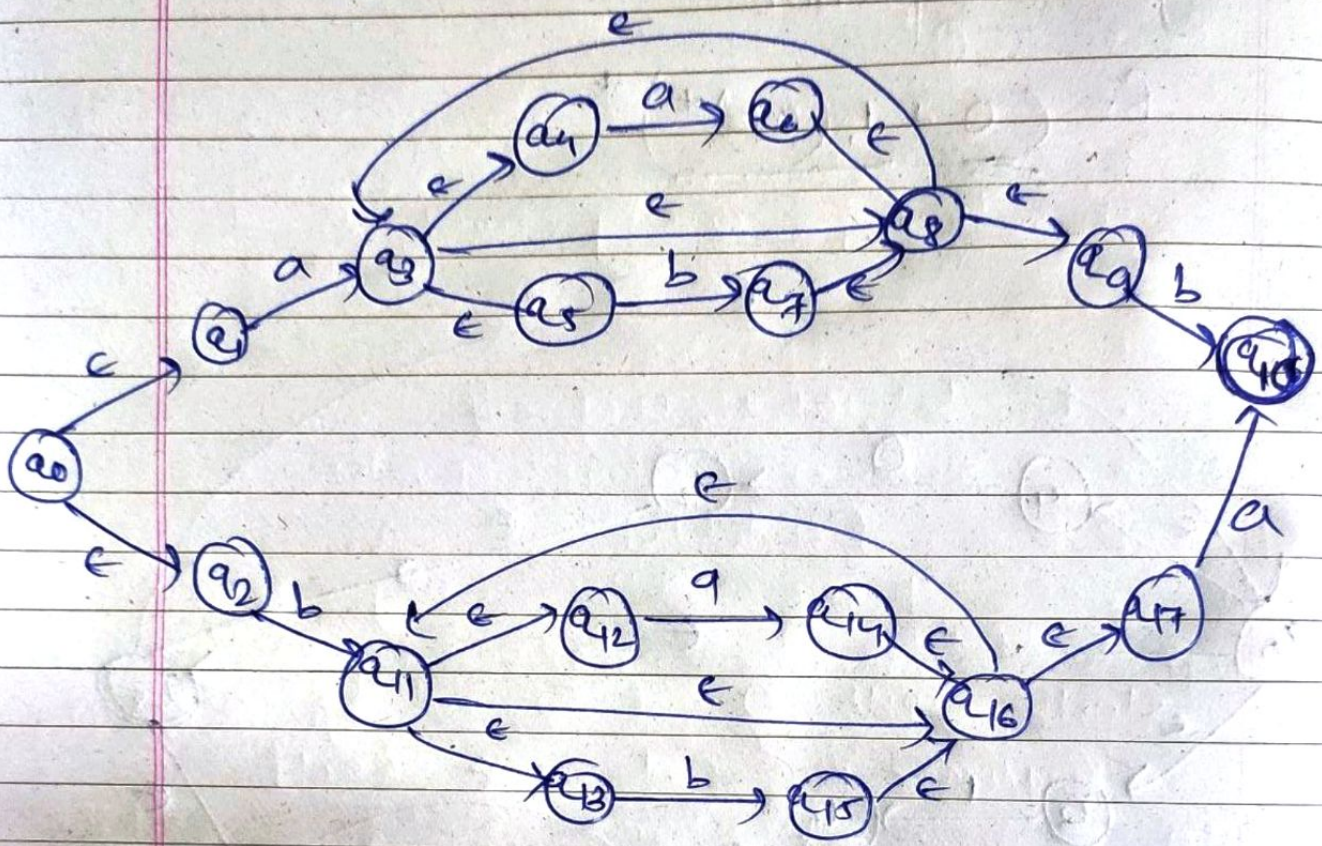






Page No.:
Date: / /

Q R.E = $\underline{a(a+b)^* b} + \underline{b(a+b)^* a}$

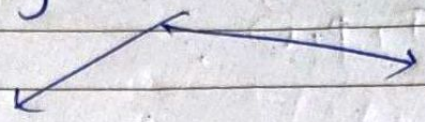


Q $((a+b)^* + abb)^*$

Q $a(ba)^* a + a(ba)^* bb$

Q $(01+10)^* + 0(01^*)^*$

String generation



Leftmost derivation Rightmost Derivation

for grammer

$S \rightarrow AIB$

$A \rightarrow 0A \mid \varepsilon$

$B \rightarrow 0B \mid 1B \mid \varepsilon$

Give the leftmost & Rightmost derivation of the string 1001

1) leftmost derivation of 1001 (In leftmost deriv leftmost variable is picked up for expansion).

$S \rightarrow A1B$ (Derivation always start from starting).

$\rightarrow 1B$ (using product^n $A \rightarrow \varepsilon$)

$\rightarrow 10B$ (using product^n $B \rightarrow 0B$)

$\rightarrow 100B$ (using product^n $B \rightarrow 0B$

$\rightarrow 1001B$ (using product^n $B \rightarrow 1B$

$\rightarrow 1001$ (using product^n $B \rightarrow \varepsilon$)

$NT \rightarrow NT.NT$
$NT \rightarrow T$

Page No.:
Date:

1) $S \rightarrow aSa \mid bSb$
$S \rightarrow a \mid b \mid aa \mid bb$

2) $S \rightarrow ABC$
$A \rightarrow a \mid b$
$B \rightarrow Bb \mid bb$
$C \rightarrow aC \mid cc \mid ba$

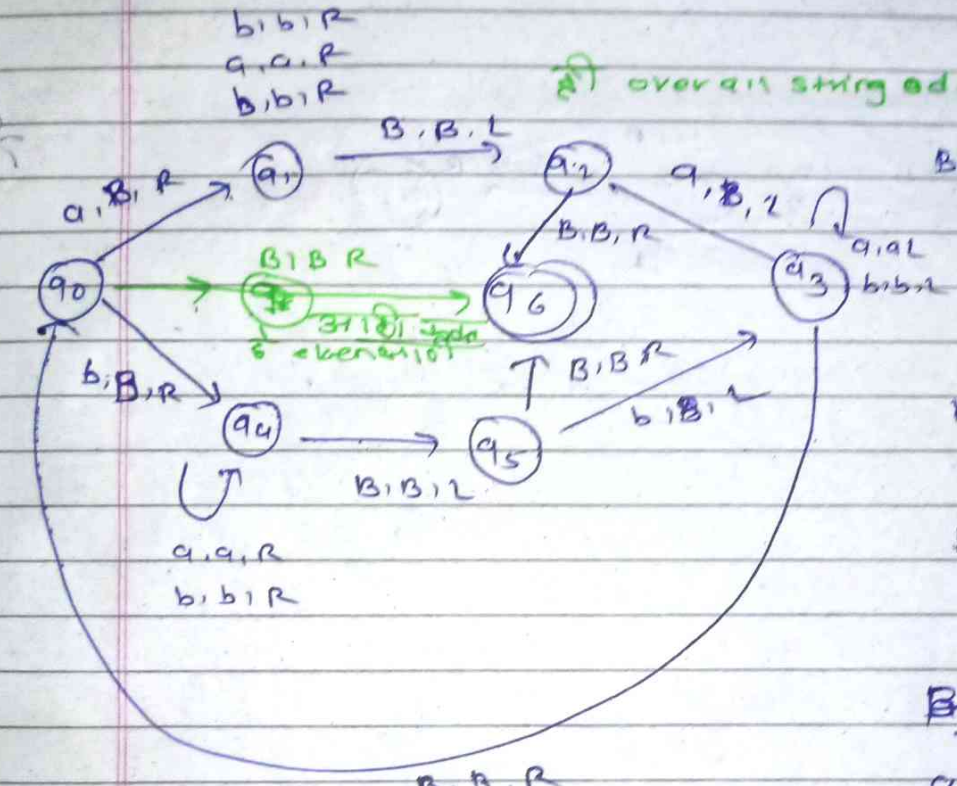
3) $S \rightarrow aB \mid aA$
$A \rightarrow aAB \mid a \mid b \mid \epsilon$
$B \rightarrow Abb \mid b$

Question हो odd या आई तब even & odd Both आती current आती चाहिए

Q5] For odd palindrome over {a, b}

waw^R
$|w|bw^R$ | $w \in \{a,b\}^*$

→ [B|a|b|a|b|a|B] ... [B|a|b|b|b|a|B]



हो overall string odd आती आती

B b a b a B B
T T T T T T

B b a b a B B
T T T T T T
T

B a b a B

B b B

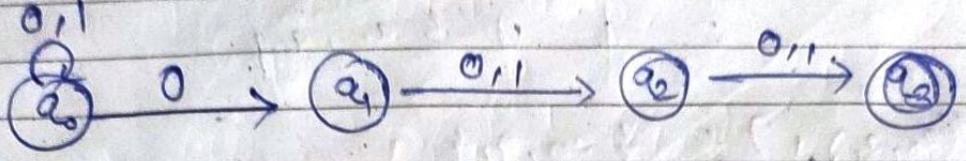
~~B B B~~ | ~~B a B~~

a b a
T T T
B

Page No.:
Date: \ . \

NFA

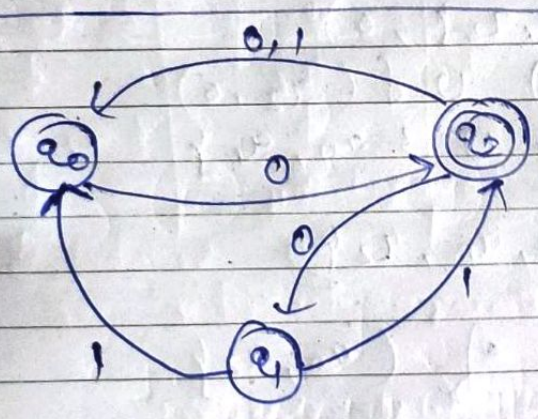
Q Over alphabet $\{0,1\}$ such that third symbol from right end is 0 $\{$



$L = \{011, 000, 100|011, \ldots\}$

NFA to DFA Conversion :-

Q



δ	0	1
q_0	$\{q_2\}$	ϕ
q_1	ϕ	$\{q_0, q_2\}$
q_2^*	$\{q_0, q_1\}$	$\{q_0\}$

$\delta(q_0, 0) = q_2$
$\delta(q_0, 1) = \phi$

$\delta(q_2, 0) = \{q_0, q_1\}$
$\delta(q_2, 1) = \{q_0\}$

$\delta(q_0, q_1) = \delta(\{q_0, q_1\}, 0)$
$= \delta((q_0, 0)) \cup \delta(q_1, 0)$
$= \{q_2\} \cup \phi$

$\delta(\{q_0, q_1\}, 0) = \{q_2\}$

Page No.:
Date: \ \

S → aAa
A → Sb | bCC
C → abb
E → aC

There is no path from S → E, Hence E is non-rechable

deleting every production containing non-rechable symbol E

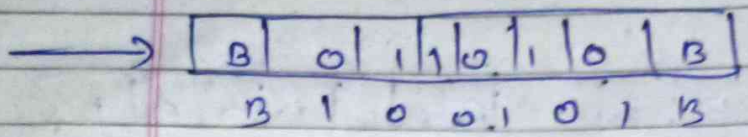
S → aAa
A → Sb | bCC
C → abb

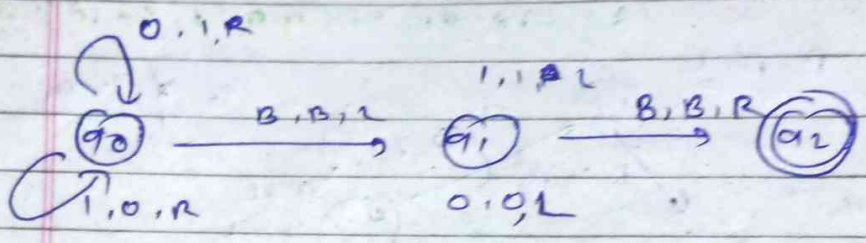
Elimination of ε-production :-
a production of the form A → ε is called a null production or ε-production

S → ABA
A → aA | ε
B → bB | ε

A is nullable ∵ production A → ε
B is nullable ∵ production B → ε
S is nullable ∵ production S → ABA

Q7] Design Turing machine For finding 1's complement of binary number.





Page No.:
Date: \ \

Right most derivation of 1001 (we will pick Rightmost variable for expansion)

$S \rightarrow A1B$ ← [derivat^n always start with starting symbol]

$\rightarrow A10B$ [Using product^n $B \rightarrow 0B$]

$\rightarrow A100B$ [using product^n $B \rightarrow 0B$]

$\rightarrow A1001B$ [using product^n $B \rightarrow 1B$]

$\rightarrow A1001$ [using product^n $B \rightarrow \varepsilon$]

$\rightarrow 1001$ [using product^n $A \rightarrow \varepsilon$]

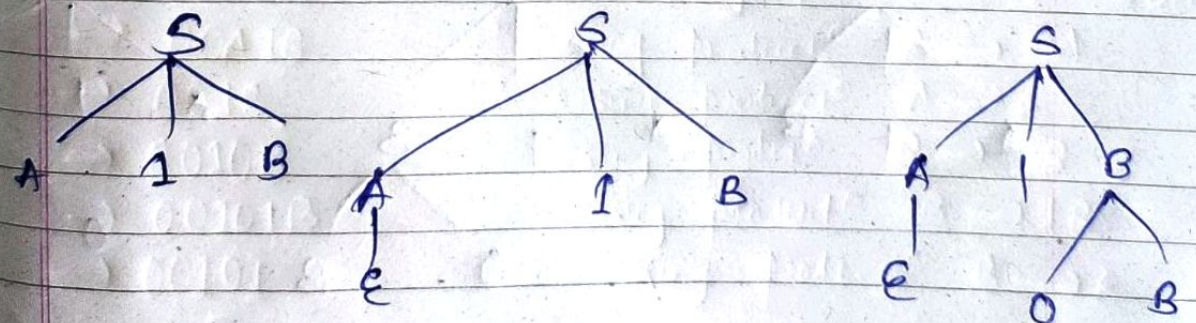
Parse Tree :-

$S \rightarrow A1B$

$A \rightarrow 0A \mid \varepsilon$

$B \rightarrow 0B \mid 1B \mid \varepsilon$

Give parse tree for leftmost & Rightmost derivation of the string 1001.



Page No.:
Date:

* Greibach Normal form (GNF):

A CFG $G = (V, T, P, S)$ is said to be in GNF if every production is of the form:

$$A \rightarrow a\alpha$$

where $a \in T$ & α is string of zero or more variables

$$NT \rightarrow T\,(NT/T)^*$$

⇒

* Remove Left Recursion:-

if $A \rightarrow A\alpha$
$A \rightarrow \beta$

$$A \rightarrow A\alpha \mid \beta$$

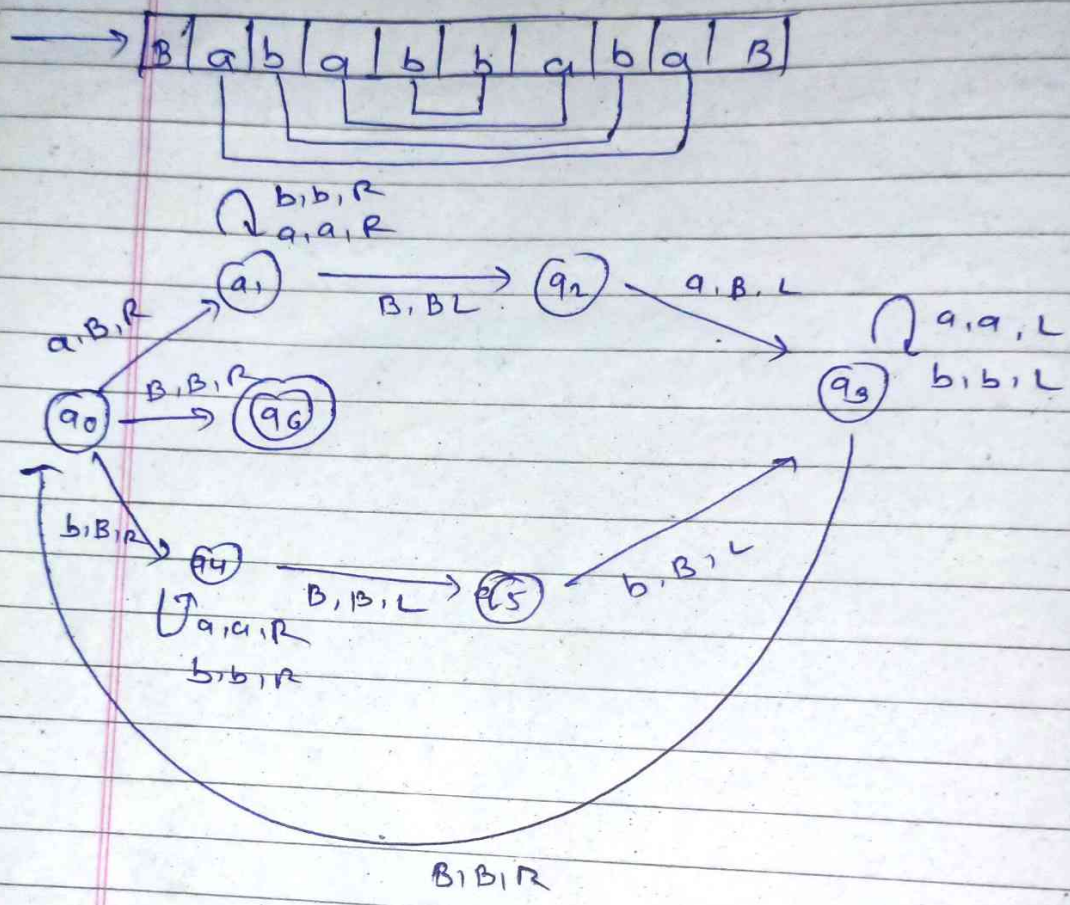
⇒

$$A \rightarrow \beta A' \mid \beta$$
$$A' \rightarrow \alpha A' \mid \alpha$$

} after removing the left recursion.

Q6] For Padindrom over $\{a,b\}^*$

$w w^R \mid w \in \{a,b\}^*$



B B b a b b a b a B

B b a b b a b B

B a b b a

B a b b B

B B

1- Successor of $\{q_0, q_1\}$

$= \delta(\{q_0, q_1\}, 1)$

$= \delta(q_0, 1) \cup \delta(q_1, 1)$

$= \phi \cup \{q_0, q_2\}$

$\delta((q_0, q_1), 1) = \{q_0, q_2\}$

new subset $\{q_0, q_2\}$, successor of subset $\{q_0, q_2\}$

0 - successor of $\{q_0, q_2\}$

$= \delta(\{q_0, q_2\}, 0)$

$= \delta(q_0, 0) \cup \delta(q_2, 0)$

$= \{q_2\} \cup \{q_0, q_1\}$

$\delta(\{q_0, q_2\}, 0) = \{q_0, q_1, q_2\}$

1 - successor of $(q_0, q_2\}$

$= \delta(\{q_0, q_2\}, 1\}$

$= \delta(q_0, 1) \cup \delta(q_2, 1)$

$= \phi \cup \{q_0\}$

$\delta(\{q_0, q_2\}, 1) = \{q_0\}$

Page No.:
Date: \ \

$S \to aS$
$S \to \varepsilon$ }

$S \to \varepsilon$, hence S is nullable symbol

$S \to a$ ∴

after removing ε production the grammer becomes

$S \to aS \mid a$

=

$S \to ABA$
$A \to \varepsilon$
$B \to \varepsilon$

A & B are nullable symbols of
$A \to \varepsilon$ & $B \to \varepsilon$

∴ $S \to ABA$ ∴ nullable.
∵ S is also nullable.

Nullable symbols { A, B, S }

$S \to ABA \mid BA \mid AA \mid AB \mid A \mid B$

Q) Define deterministic and non-deterministic Turing machine.

→ (i) Deterministic Turing machine

- A deterministic Turing machine defined as where for each possible state and tape symbol has exactly one possible state/action.

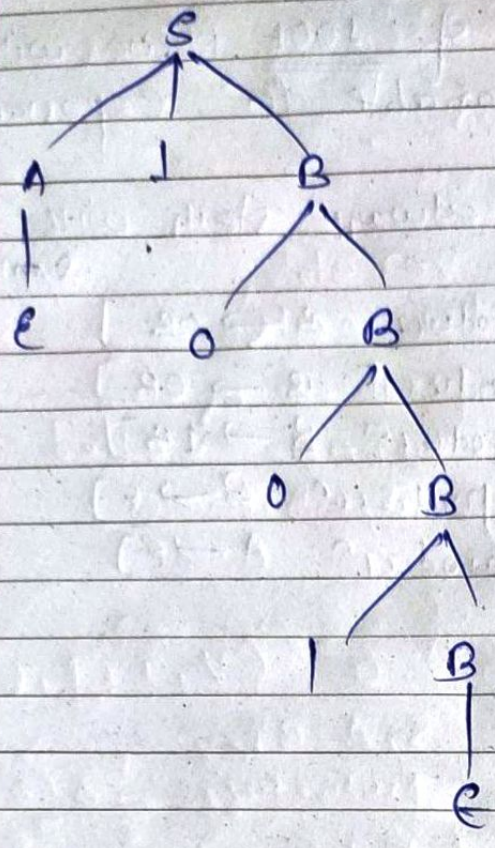
(ii) Non-deterministic Turing machine

- A non-deterministic turing machine defined as where, for each possible state and tape symbol has more than one action.

Q) Define LBA and its tuple

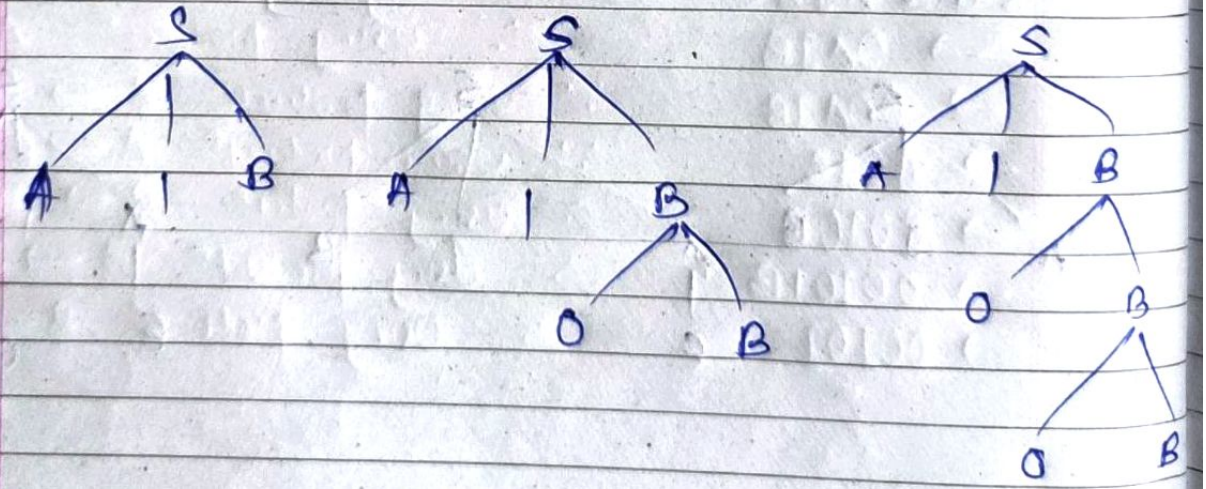
→ - LBA stands for linear bounded Automata.
- LBA is defined by a 7-tuples.
$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$

where,
Q: Finite set of states.
Σ: input alphabets.
Γ: Tape alphabets.
δ: transition function
q_0: initial or start state
q_{accept}: accept state
q_{reject}: reject state.



Leftmost derivation's parse tree.

= Parse tree for Rightmost Derivation:-



Page No.:
Date:

Q $A \rightarrow Aa \mid b$

$A \rightarrow A\alpha \mid \beta$

$A \rightarrow \beta A' \mid \beta$

$A' \rightarrow \alpha A' \mid \alpha$

~~$A \rightarrow bB \mid b$~~

$A \rightarrow bB \mid b$

$B \rightarrow aB \mid a$

Q $A \rightarrow Aa \mid b \mid c$

$\alpha \quad \beta_1 \quad \beta_2$

Removing left Recursion,

$A \rightarrow bp \mid cp \mid b \mid c$

$p \rightarrow ap \mid a$

Q $S \rightarrow S10 \mid 01 \mid 1$

$\alpha \quad \beta_1 \quad \beta_2$

$A \rightarrow A\alpha \mid \beta_1 \mid \beta_2 \mid \ldots$

$A \rightarrow \beta_1 A' \mid \beta$

$A' \rightarrow \alpha A' \mid \alpha$

after Removal of left Recursion,

$S \rightarrow 0R \mid 1R \mid 01$

$R \rightarrow 10R \mid 10$

new subset $\{q_0, q_1, q_2\}$ is generated successor of $\{q_0, q_1, q_2\}$

0-successor of $\{q_0, q_1, q_2\}$

$= \delta(\{q_0, q_1, q_2\}, 0)$

$= \delta(q_0, 0) \cup \delta(q_1, 0) \cup \delta(q_2, 0)$

$= \{q_2\} \cup \phi \cup \{q_0, q_1\}$

$\delta(\{q_0, q_1, q_2\}, 0) = \{q_0, q_1, q_2\}$

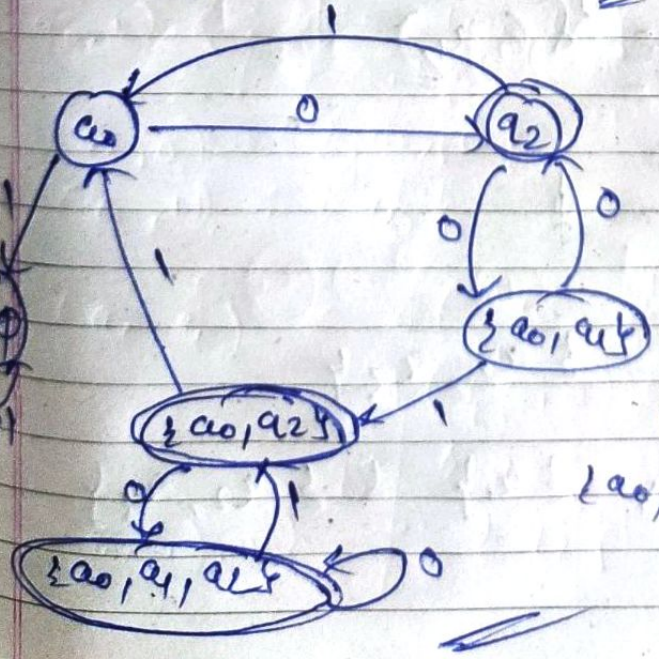
1-successor of $\{q_0, q_1, q_2\}$

$= \delta(\{q_0, q_1, q_2\}, 1)$

$= \delta(q_0, 1) \cup \delta(q_1, 1) \cup \delta(q_2, 1)$

$= \phi \cup \{q_0, q_2\} \cup \{q_0\}$

$\delta(\{q_0, q_1, q_2\}, 1) = \{q_0, q_2\}$



δ	0	1
q_0	$\{q_2\}$	ϕ
q_2^*	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_2\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_0\}$
$\{q_0, q_1, q_2\}^*$	$\{q_0, q_1, q_2\}$	$\{q_0, q_2\}$

$S \rightarrow aS \mid AB$
$A \rightarrow \varepsilon$
$B \rightarrow \varepsilon$

Nullable symbols $= \{A, B, S\}$

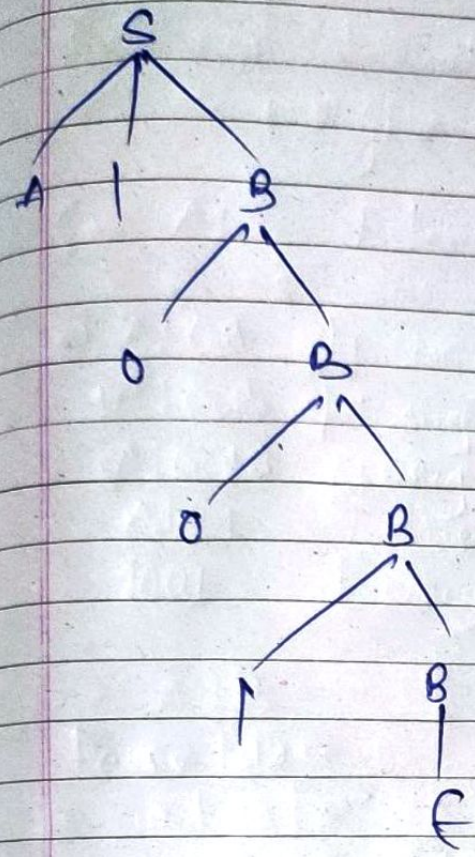
$S \rightarrow aS \mid AB \mid a \mid B \mid A$

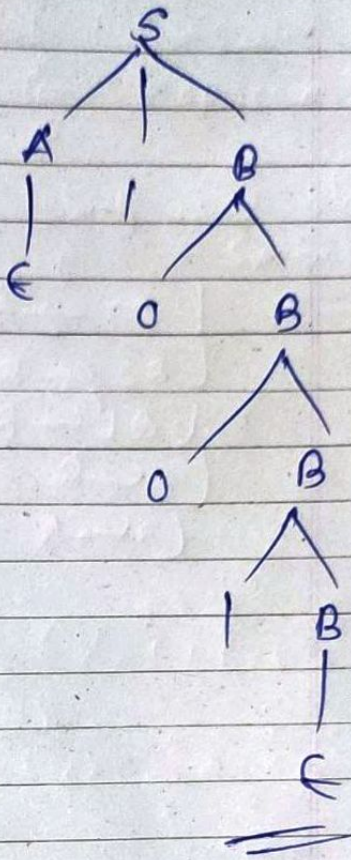
Remove ε-production from the grammar.

1) $S \rightarrow ABA$
$A \rightarrow aA \mid \varepsilon$
$B \rightarrow bB \mid \varepsilon$

2) $S \rightarrow AB$
$A \rightarrow aAA \mid \varepsilon$
$B \rightarrow bBB \mid \varepsilon$

3) $S \rightarrow aABbCD \mid \epsilon$
$A \rightarrow Amd \mid \varepsilon$
$B \rightarrow SAn \mid hn \mid \varepsilon$
$C \rightarrow Si \mid Cg$
$D \rightarrow aBD \mid \varepsilon$





for the grammer

$S \rightarrow 0S1 \mid 01$

Give a leftmost derivation of 000111

Page No.:
Date: \ \

* Elimination of unit production :-

$A \rightarrow B \rightarrow C \rightarrow D$

$A \rightarrow D$

- Simplify the following grammar :-

$S \rightarrow Ab$

$A \rightarrow a$

$B \rightarrow c \mid b$

$C \rightarrow D$

$D \rightarrow E$

$E \rightarrow a$

= There is ary ϵ production in the given grammer.

$B \rightarrow C \rightarrow D \rightarrow E$

after removing the unit production grammer becomes,

$S \rightarrow Ab$

$A \rightarrow a$

$B \rightarrow a \mid b$

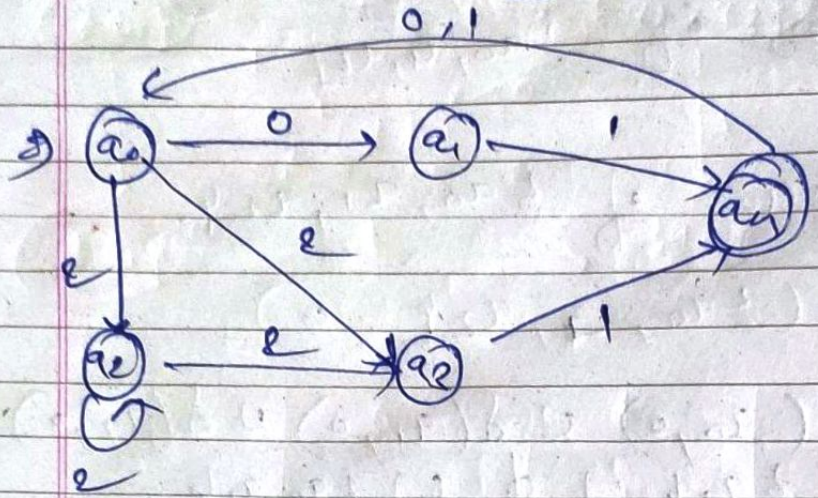
after removing useless symbol.

$S \rightarrow Ab$ }

$A \rightarrow a$ }

=

Construct DFA equivalent to NFA.



δ	0	1	ε
q_0	q_1	ϕ	$\{q_2, q_3\}$
q_1	ϕ	q_4	ϕ
q_2	ϕ	ϕ	$\{q_2, q_3\}$
q_3	ϕ	q_4	ϕ
q_4^*	q_0	q_0	ϕ

$\{q_0\}$ is the first subset

0-successor of q_0 is

$\delta(q_0, 0) = \delta(q_1)$

$=$

1-successor $=$

$\delta(q_0, 1) = \phi$

2-successor :

$\delta(q_0, \varepsilon) = \{q_2, q_3\}$

$=$

New subsets $\{q_1\}$ & $\{q_2, q_3\}$

$=$

$S \rightarrow aS \mid AB$
$A \rightarrow a \mid \varepsilon$
$B \rightarrow b \mid \varepsilon$
$D \rightarrow b$

after removing ε production :

$S \rightarrow aS \mid AB \mid A \mid B$
$A \rightarrow a$
$B \rightarrow b$
$D \rightarrow b$

after removing useless symbol $D \rightarrow b$

$S \rightarrow aS \mid AB \mid A \mid B$
$A \rightarrow a$
$B \rightarrow b$

1) $S \rightarrow A \mid bb$
$A \rightarrow B \mid b$
$B \rightarrow S \mid a$

2) $S \rightarrow XaX \mid bX$
$X \rightarrow X \circ X \mid XbX \mid \varepsilon$

3)

Page No.:
Date: \ \

0 - successor of $\{q_4\} = \phi$

1 - successor of $\{q_4\} = \{q_4\}$

2 - successor of $\{q_4\} = \phi$

0 - successor of $\{q_2, q_3\}$

$\delta(\{q_2, q_3\}, 0) = \delta(q_2, 0) \cup \delta(q_3, 0)$

$= \phi \cup \phi = \phi$

1 -

$\delta(\{q_2, q_3\}, 1) = \delta(q_2, 1) \cup \delta(q_3, 1)$

$= \phi \cup q_4 = \{q_4\}$

2 - successor of $\{q_2, q_4\}$

$\delta(\{q_2, q_4\}, 2) = \delta(q_2, 2) \cup \delta(q_4, 2)$

$= \{q_2, q_3\} \cup \phi$

$= \{q_2, q_3\}$

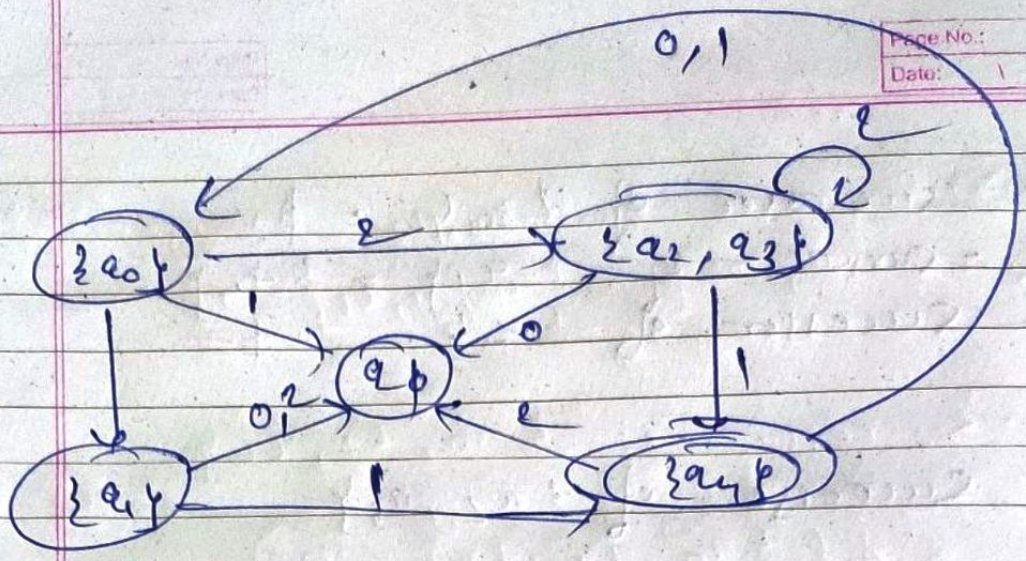
new subsets $\{q_4\}, \{q_2, q_3\}$

0 - successor of $\{q_4\}$

$= \delta(q_4, 0) = \{q_0\}$

1 - successor $\{q_4\} = \{q_0\}$

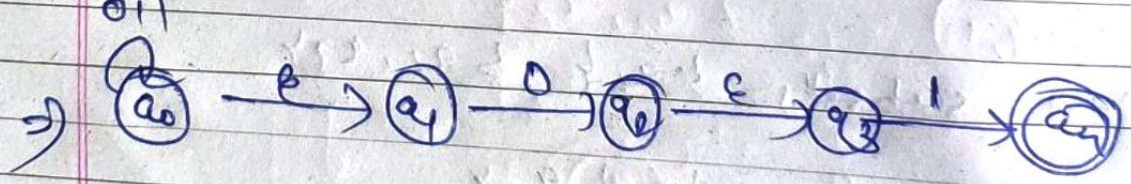
2 - successor $\{q_4, 2\} = \phi$

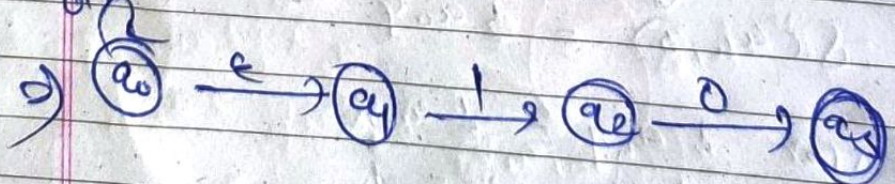


	0	1	ε
{q0}	{q1}	ϕ	{q2, q3}
{q1}	ϕ	{q4}	ϕ
{q2, q3}	ϕ	{q4}	{q2, q3}
{q4}*	{q0}	{q0}	ϕ

* NPA with ε - transitions :-

ε stands for null symbol, ε transition allows transition on ε (no input)

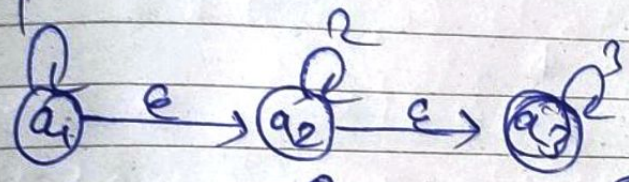
⇒ 

⇒ 

Page No.:
Date: \ \

formal Notation for ε-NFA

- Q = a finite set of states
- Σ = is an alphabet
- q_0 = $q_0 \in Q$ is the initial state
- F = $F \subseteq Q$ is a set of final/accepting state
- δ = a transition function $Q \times \{\Sigma \cup \{\epsilon\}\}$

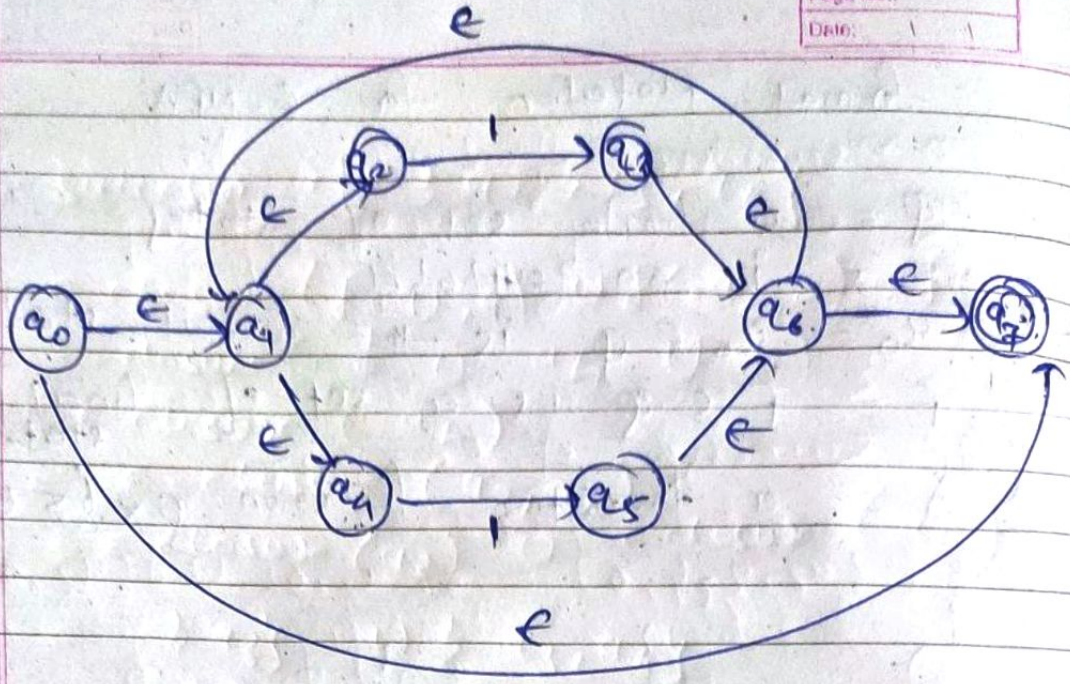


Q Σ q_0 F

$M = \{(q_1, q_2, q_3), (1,2,3), \delta, q_1, \{q_3\}\}$

δ =

	1	2	3	ε
q_1	$\{q_1\}$	ϕ	ϕ	$\{q_2\}$
q_2	ϕ	$\{q_2\}$	ϕ	$\{q_3\}$
q_3^*	ϕ	ϕ	$\{q_3\}$	ϕ



ε closure of $q_0 = \{q_0, q_1, q_2, q_4, q_7\}$

ε closure of $q_1 = \{q_1, q_2, q_4\}$

ε —— of $q_2 = \{q_2\}$

ε —— $q_4 = \{q_4\}$

ε closure of $q_3 = \{q_3, q_6, q_7\}$

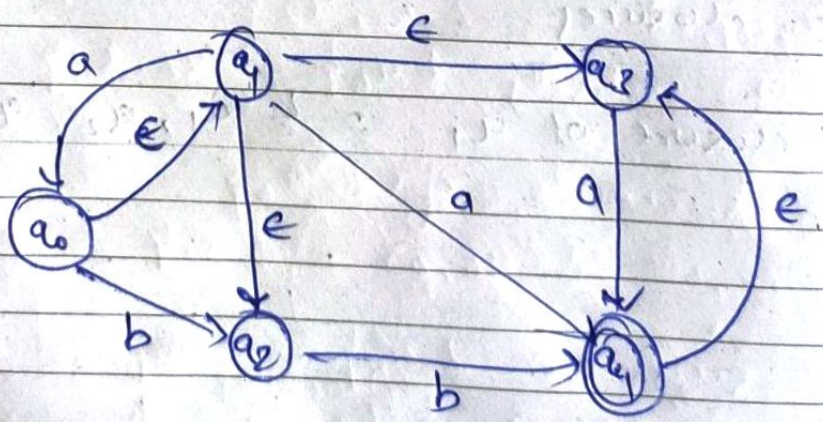
ε closure of $q_5 = \{q_5, q_6, q_7\}$

ε closure of $q_6 = \{q_1, q_2, q_4, q_6, q_7\}$

ε —— $q_7 = \{q_7\}$

ε-NFA to DFA :-

Q



Page No.:
Date: \ \

ε closures of state :- | $\Sigma = \{a, b, \epsilon\}$

State	ε-closure
q_0	$\{q_0, q_1, q_2, q_3\}$
q_1	$\{q_1, q_2, q_3\}$
q_2	$\{q_2\}$
q_3	$\{q_3\}$
q_4	$\{q_3, q_4\}$

a- successor of ε-closure $(q_0) = \{q_0, q_1, q_2, q_3\}$

$= \epsilon\text{-closure}(\delta(q_0, a) \cup \delta(q_1, a) \cup \delta(q_2, a), \cup \delta(q_3, a))$

$= \epsilon\text{-closure}(\phi \cup \{q_0, q_4\} \cup \phi \{q_4\})$

$= \epsilon\text{-closure}(q_0, q_4)$

$= \epsilon\text{-closure}\{q_0\} \cup \epsilon\text{-closure}\{q_4\}$

$= \{q_0, q_1, q_2, q_3\} \cup \{q_3, q_4\}$

$= \{q_0, q_1, q_2, q_3, q_4\}$

b - successor of ε-closure $(q_0) = \{q_0, q_1, q_3, q_3\}$

$= \epsilon\text{-closure}(\delta(q_0, b) \cup \delta(q_1, b) \cup \delta(q_2, b) \cup \delta(q_3, b))$

$= \epsilon\text{-closure}(\{q_2\} \cup \phi \cup \{q_4\} \cup \phi)$

$= \epsilon\text{-closure}(\{q_2, q_4\})$

$= \epsilon\text{-closure}(q_2) \cup \epsilon\text{-closure}(q_4)$

$= \{q_2\} \cup \{q_3, q_4\} =$

$= \{q_2, q_3, q_4\}$

Page No.:
Date: \ \

a - successor of $\{q_2, q_3, q_4\}$

$= \epsilon\text{-closure}(\delta(q_2, a) \cup \delta(q_3, a) \cup \delta(q_4, a))$

$= \epsilon\text{-closure}(\phi \cup q_4 \cup \phi)$

$= \epsilon\text{-closure}(q_4)$

$= \epsilon\text{-closure}(q_3, q_4)$ $\epsilon\text{-closure}(q_4)$

$= \epsilon\text{-closure}(q_2) \cup$

$= \{q_3\} \cup \{q_3, q_4\}$

$= \{q_3, q_4\}$