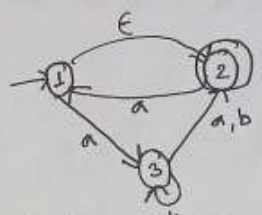
Name – MD. Rakib Hasan
ID – 2216080021
Course Code – CSI-317.
——— X ———

Answer to the Question No-01

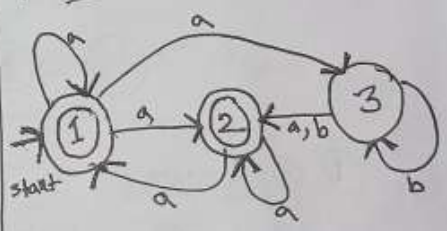
(b) Ans:



Transaction table for ϵ-NFA

ϵ-NF	a	b	ϵ
→ 1	{3}	ϕ	{2}
2	{1}	ϕ	ϕ
3	{2}	{2,3}	ϕ

⇒ NFA transaction diagram —



ϵ-closure for (1) → {1,2}
(2) → {2}
(3) → {3}

Transaction table for equivalent NFA.

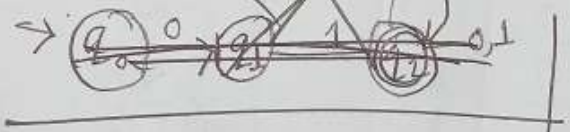
NFA	a	b
→ 1	{1,2,3}	ϕ
2	{1,2}	ϕ
3	{2}	{2,3}

1 © Ans: Non-terminal : Non-terminals are the Non-leaf Nodes in a parse tree. In the Expression grammer. E, T & F are non terminals. Some times Nonterminals are Enclosed Bawteen angle brackets to distinguish them from terminals. <digit> and <integer> are non terminals Symbols And. S, A. are non terminal symbols.

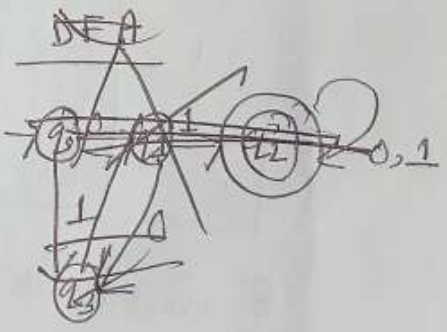
(1) (a) Ans: $L = \{w \mid w$ is a binary string End in $01\}$

NFA

$L = \{01, 010, 011, \ldots\}$

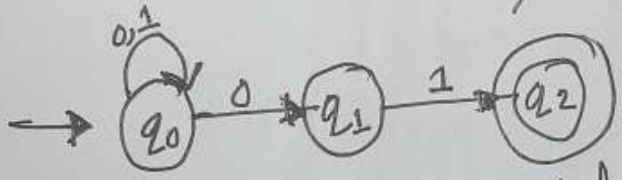


DFA



(I) (a) Answer:

$L = \{w \mid w$ ends in $01\}$

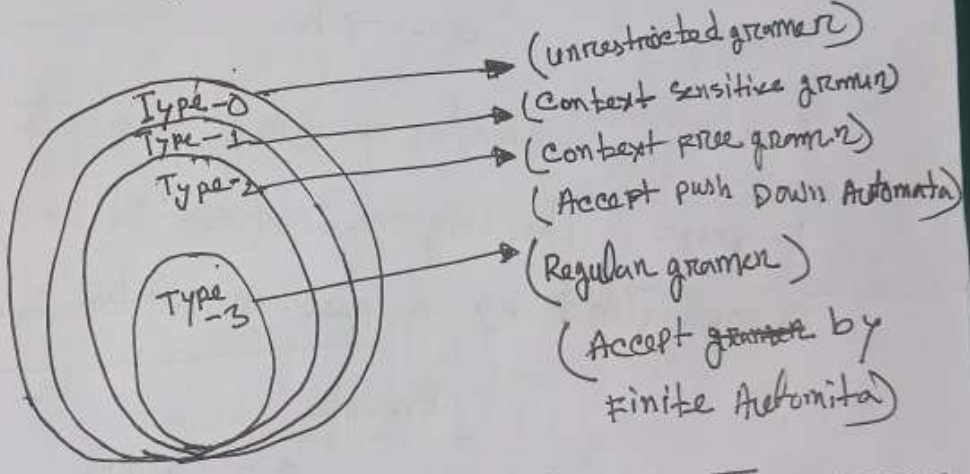


Find state.

Answer to the Question No - 03

(a) Answer: According to chomsky hierarchy, gramer is divided In to 4 types as Fallow.

(i) Type 0 is known as Unrestricted gramer.
(ii) Type 1 is known as Context - Sensitive gramer.
(iii) Type 2 is known as Context - Free gramer.
(iv) Type 3 Regular gramer.



① Type-0 : Type 0 gramers Include all formal gramer. Type 0 gramer Language are recognized by turing machin.

⇒ Example

Sab --> ba
A --> S.

variable S, A And Terminals a, b

$\alpha \rightarrow \beta$ where.
α is $(V+T)^* V (V+T)^*$
V : variable.
T : Terminals.
β is $(V+T)^*$.

Type 1: Context – sensitive grammar:

→ First of all Type 1 grammar should Type 0.

→ Grammar production in the form of

$$\alpha \rightarrow \beta$$

$$|\alpha| <= |\beta|$$

Example

S --> AB

AB --> abC

B --> b.

Type-2: Context-free grammar generate context-free Language. The Language generated by the grammar is Recognized by a pushdown automata. In Type-2.

Example

S --> AB

A --> a

B --> b

Type-3: Type 3 grammars generated regular languages. These Languages are Exactly all Language that can be Accepted by a finite-state Automation

Example:

S --> a.

3 (b) Answer: Context Free grammer:

=> Language generated by Context Free grammer is accepted by pushdown Automata.

=> It is a subset of type 0 And Type 1 grammer and superset of type 3 Grammer.

=> Also called phase structured grammer.

=> Only one parse tree -> Unambiguos.

=> More than one parse tree -> Ambiguous.

Regular grammer:

=> It is Accepted by Finite state Automata.

=> It is a Subset of typ 0, typ 1 And type 2 grammer.

=> This Language it generates is called regular Language.

=> Regular Language are closed under Oparation like Union-Intersection, Complement e.t.c,

=> They are the most restricted form of grammer.

3 © Answer: DFA for the regular Expression, $R = 1^*00^*1(0+1)^*$

=> DFA—

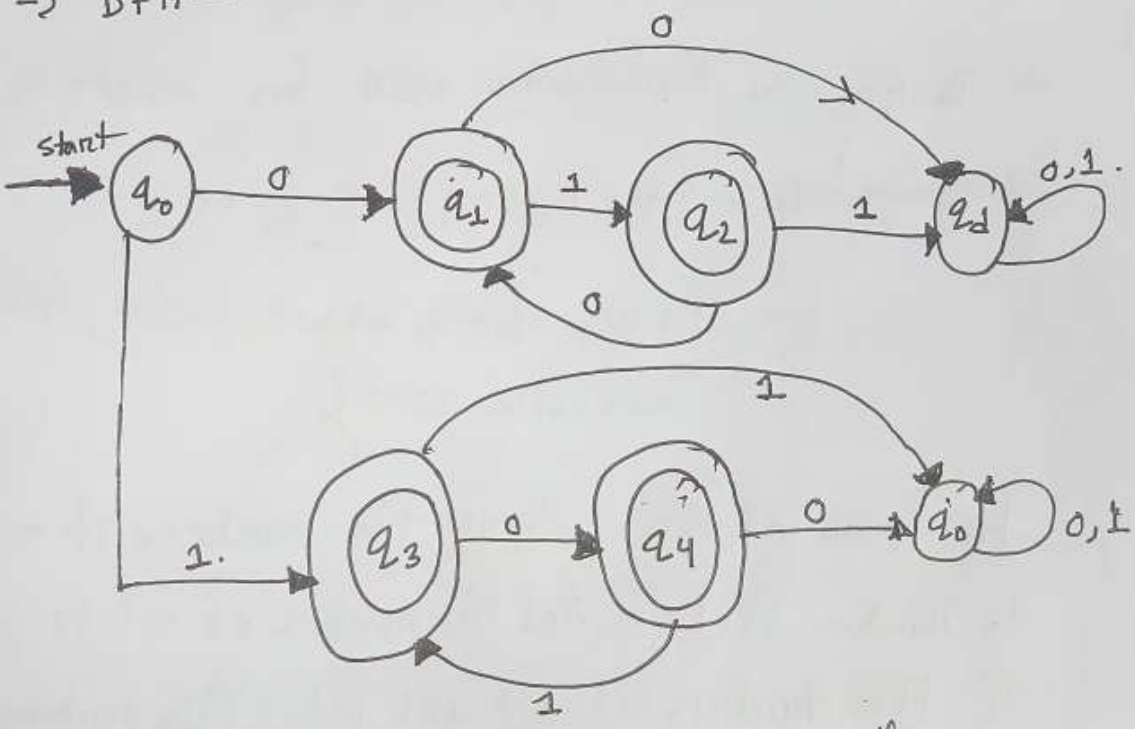


Fig- $R = 1^*00^*1(0+1)^*$

Answer to the qustion No - 4

(a) Npda for the Language $L = \{a^n b^n c^m | m, n \geq 1\}$

=> Design a non deterministic PDA for accepting the Language $L = \{a^n b^n c^m | n \geq 1\}$, ie.

$L = \{abc, abcc, abccc, aabbc, aaabbcc, aaaabbbb cccc\}$

In each of the string the number of a's is equal to Number of b's. And the number of c's is independent of the Number of a's and b's. This problm is quit Similar to the NPDA for accepting Longuage ~~$L = \{a^n b^n | n \geq 1\}$~~ $L = \{a^n b^n | n >= 1\}$ The only difference is here we add

=> Explanation: Here, we need to maintain the order of a's, b's c's. That is all the a's are coming first and then all the b's and then c's are coming. Thus, we need stack as along with the stat Diagram The Count of a's and b's is maintained by stack, we will take 2 stack Alphabets.

$\Gamma = \{a, z\}$.

P, t. O

Where are Γ = set of all stack Alphabet

z = stack start symbol.

PDA stack transition function —

$$\delta(q_0, a, z) \vdash (q_0, az)\ \delta(q_0, a, a) \vdash$$

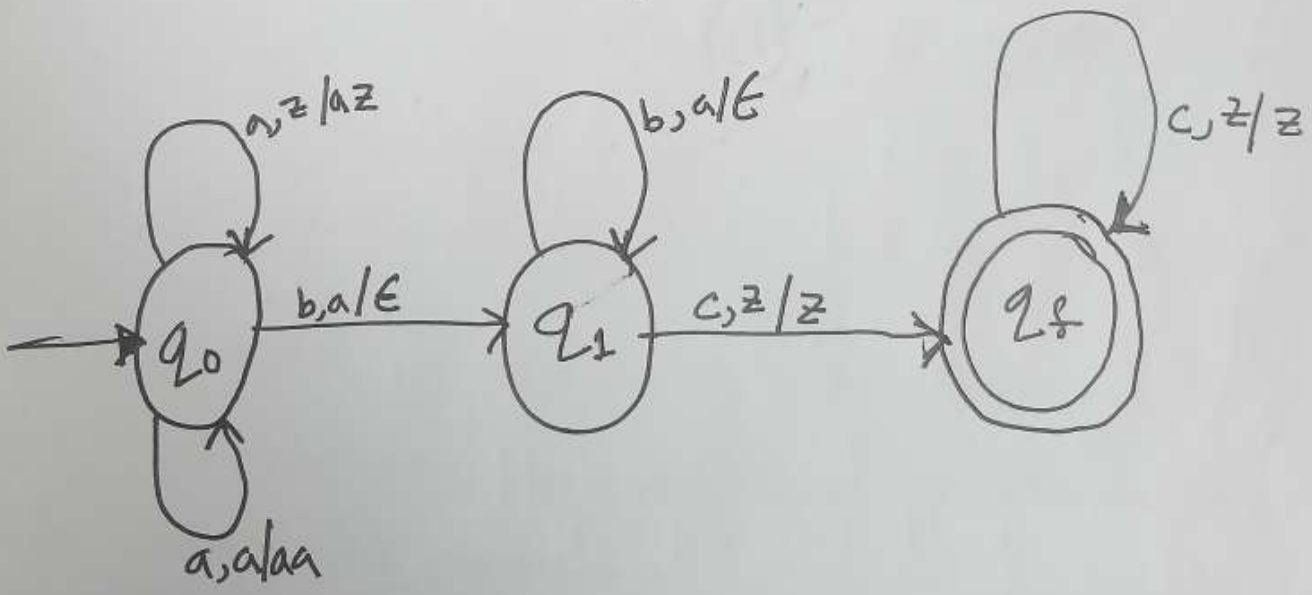
$$(q_0, aa)\ \delta(q_0, b, a) \vdash (q_1, a)\ \delta(q_1, b, a) \vdash$$

$$(q_1, a)\ \delta(q_1, c, a) \vdash (q_2, c, a) \vdash (q_2, \epsilon) \vdash (q_2, \epsilon, z)$$

$$\vdash (q_f, z)$$

Where, q_0 = Initial state q_f = Final state

ϵ = indicated pop Operation.



4 (b)

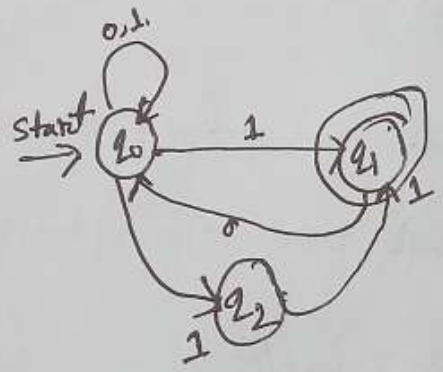
Answer: $L = \{w\}$ w is a bit string which Contains the Substring 11 $\}$

⇒ state Design:

→ q_0: start state (initially off), also means the most recent input wasn't Not a 1.

→ q_1: has never seen 11 but the most recent Input was a 1.

→ q_2: Has seen 11 at least once.



Answer to the Question No - 5

(a) Ans: Turing machine: A Turing Machine (TM) is a mathmatical Model wich ~~read~~ Consists of an infinite bngth tape divided into calls on which input isgiven. It consists of a head which reads the Input tape.
A state registen stores the state of the turing machine. After reading an Input symbol, it is reaplaced with Another symbal. Its ~~the~~ Internal state, ~~the input~~ is changed, and it moves ~~from~~ one call to the right. or left. If the TM reaches the final state, the Input string is Accepted, otherwise rejected.

2(c) Ans: ~~ca~~ Is Empty in stack? Stack.Empty() is use to check if a stack is Empty or not — This method requires No parameters. It return true if the stack is Empty and false if the stack is Not Empty.
std::stack::pop: Removes the element on top of the stack Effectively reducing its size by one. The element removed is the latest element. Interested in to the stack. whose value can be retrieved by calling member Stack::top.

⇒ ~~Example~~:

(5) (b) Answer: turing machin that recognizes,

$L = 0(0+1)^* 1$.

Ans: $\Sigma = 01$

$L = 01^* 1$.