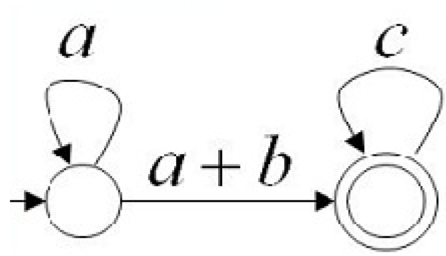
SHAMBHUNATH INSTITUTE OF ENGINEERING AND TECHNOLOGY

FIRST SESSIONAL EXAMINATION, EVEN SEMESTER, (2019-2020)

Theory of automata & formal language (KCS-402)

Section- A

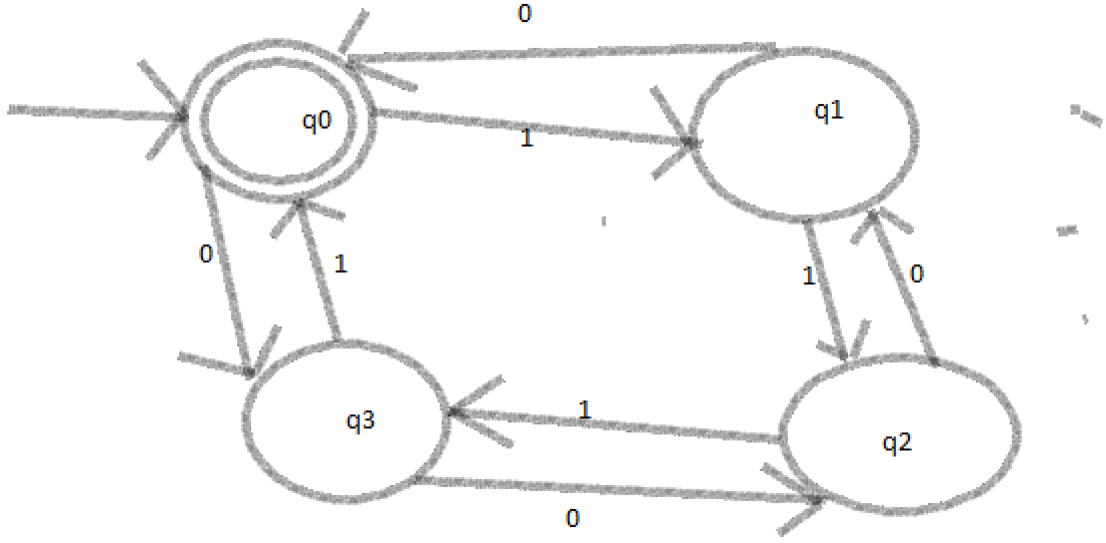
1.

1.a.	What do you mean by Λ- closure in FA? Solution: Λ -closure for a given state Q means a set of states which can be reached from the state Q With only (null) move includes the state Q itself.
1.b.	Design a regular expression that accepts all the strings for input alphabet {a,b}containing exactly 2 b's. Solution: **a*bba***
1.c.	What do you understand by generalized transition graph? Solution: It is just like Transition graph but directed edge connecting some pair of state and given by r.e. It is denoted by $\hat{\delta}$. $\hat{\delta}(q_i, re) = q_j$ 
1.d.	Give English description of the language of the following regular expression (0*.1*)*00(0+1)*. Solution: The set of all string which contains exactly two 0's.
1.e.	Given a DFA M. Suggest a procedure to draw DFA which accepts the complement of the language accepted by M. Solution: The complement of a DFA can be obtained by **making the non-final states as final states and vice-versa**.

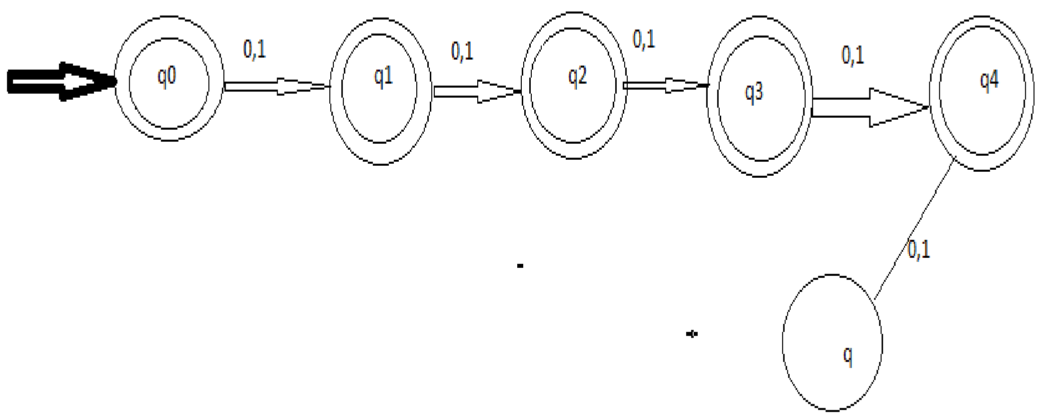
SECTION - B

2.

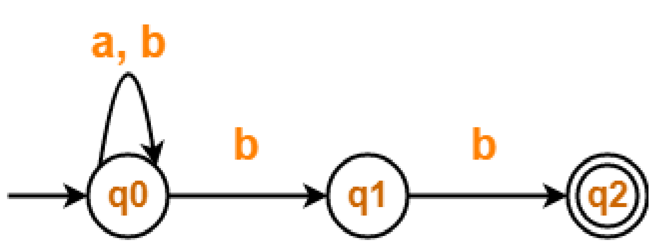
a.	Design the DFA of the following language over {0,1}: (i). All strings with Even no. of 0's and even no. of 1's. (Ii). All strings of length at most 4. Solution: (i)



(II)



b. Differentiate Between NFA & DFA. Convert the following NFA to equivalent DFA.



Solution: Difference between Deterministic Finite Automata and the Non deterministic Finite

Automata ((DFA Vs NFA):

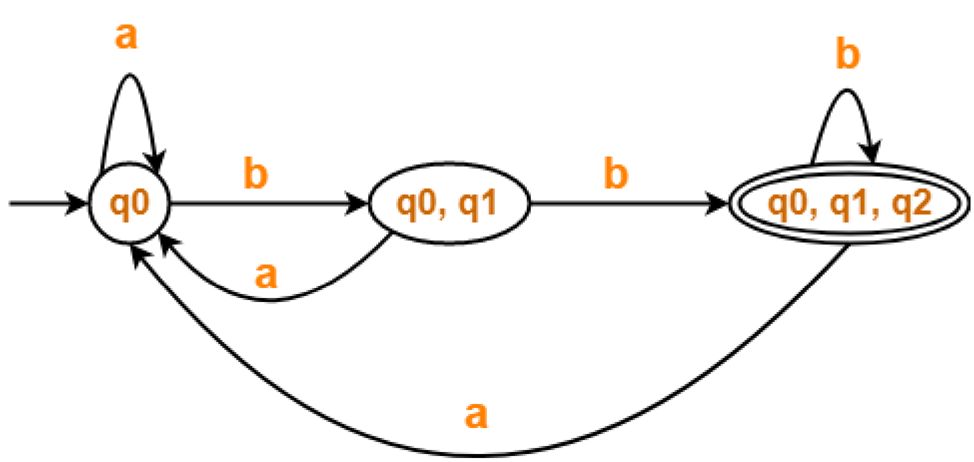
S. No.	DFA	NFA
1.	For Every symbol of the alphabet, there is only one state transition in DFA.	We do not need to specify how does the NFA react according to some symbol.
2.	DFA cannot use Empty String transition.	NFA can use Empty String transition.
3.	DFA can be understood as one machine.	NFA can be understood as multiple little machines computing at the same time.
4.	DFA will reject the string if it end at other than accepting state.	If all of the branches of NFA dies or rejects the string, we can say that NFA reject the string.
5.	Backtracking is allowed in DFA.	Backtracking is not always allowed in NFA.
6.	DFA can be understood as one machine.	NFA can be understood as multiple little machines computing at the same time.
7.	DFA will reject the string if it end at other than accepting or final state.	If all of the branches of NFA dies or rejects the string, we can say that NFA reject the string.
8.	DFA is more difficult to construct.	NFA is easier to construct.

NFA to equivalent DFA:

State / Alphabet	a	b
→q0	q0	{q0, q1}
{q0, q1}	q0	{q0, q1, q2}

{q0, q1, q2}	q0	{q0, q1, q2}

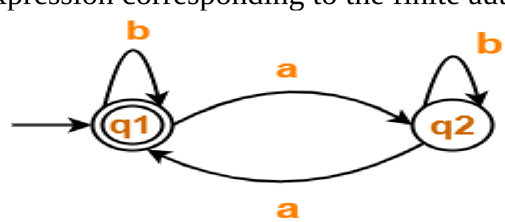
State / Alphabet	**a**	**b**
→q0	q0	{q0, q1}
{q0, q1}	q0	*{q0, q1, q2}
***{q0, q1, q2}**	q0	*{q0, q1, q2}



Deterministic Finite Automata (DFA)

c.

Find the regular expression corresponding to the finite automata given below:



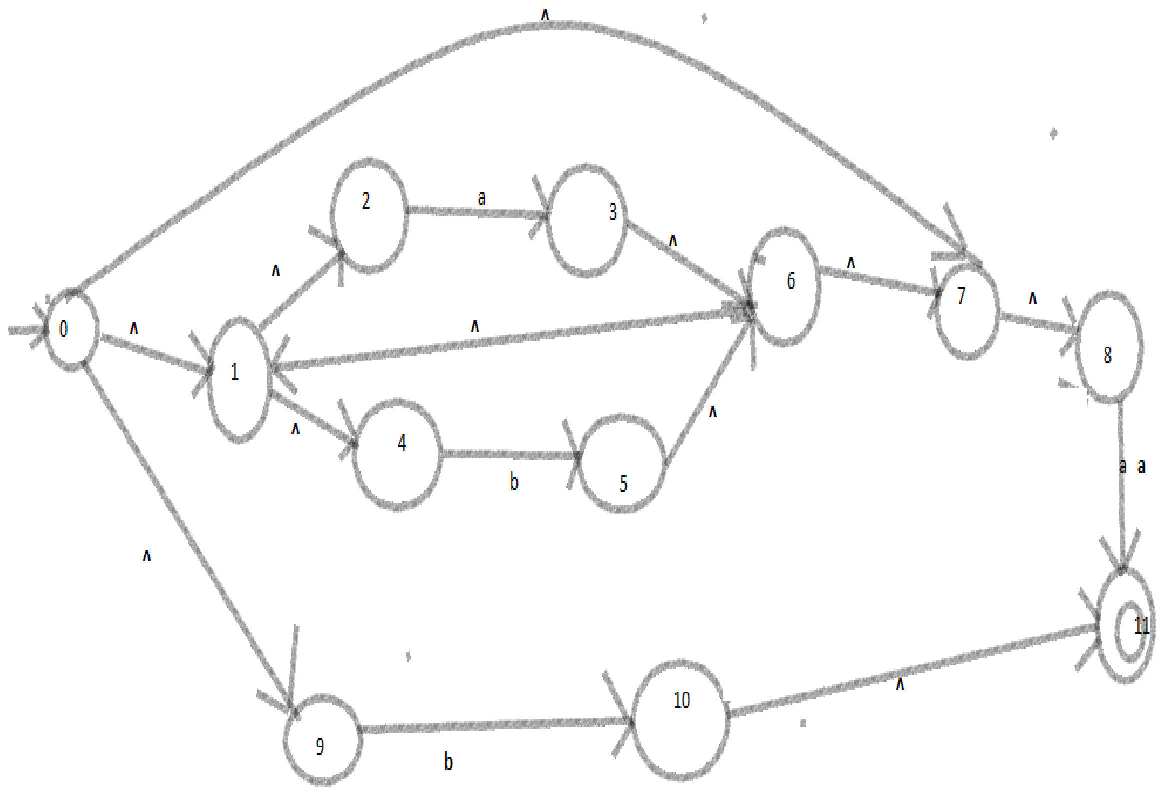
Solution: Form a equation for each state-

q1 = q1.b + q2.a + ^ ……(1)

q2 = q1.a + q2.b ……(2)

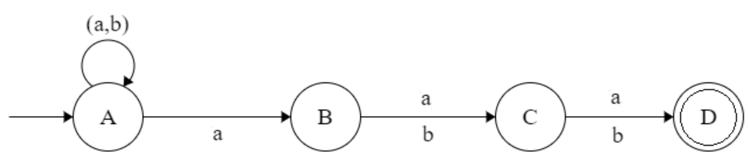
Bring final state in the form R = Q + RP.

	Using Arden's Theorem in (2), we get- q2 = q1.a.b* …….(3) Using (3) in (1), we get- q1 = ∈ + q1.b + q1.a.b*.a q1 = ∈ + q1.(b + a.b*.a) …….(4) Using Arden's Theorem in (4), we get- q1 = ∈.(b + a.b*.a)* q1 = (b + a.b*.a)* Thus, Regular Expression for the given DFA = **(b + a.b*.a)***
d.	Design finite automaton of the following regular expression: Solution: **(a+b)*a+b**

SECTION - C

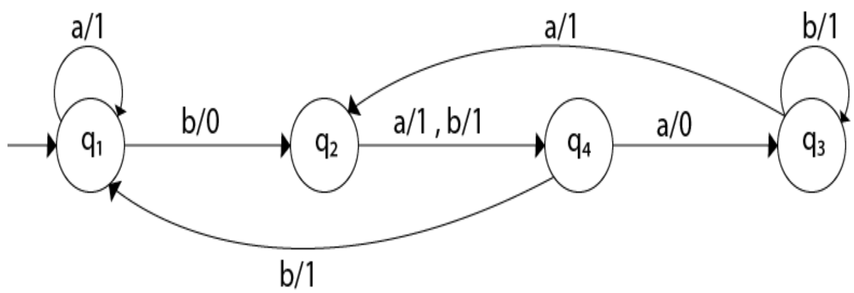
3.

a.	Design a NFA for the language L which accepts all the string in which the third symbol from right side is always 'a' over input {a,b}. Also write the regular expression for this language. Solution: The NFA of the language containing all the strings in which 3rd symbol from the RHS is "a" is:  Required NFA Regular Expression: **(a+b)* a (a+b) (a+b)**
b.	Let ∑={a,b}. For each of the following languages over ∑, find a regular expression representing it: (i) All string that exactly contain one 'a'. (ii) All string beginning with 'ab'. (iii) All string that contain either the sub-string 'aaa' or 'bbb'. Solution:

	(i) **r .e.=b*ab*** **(ii)** **r .e.=ab(a+b)*** **(iii)** **r .e. =(a+b)*(aaa+bbb) (a+b)***

4.

	Differentiate Mealy and Moore machine with example. Convert the given Mealy machine as shown in fig. into Moore Machine.



Solution: Mealy Machine vs. Moore Machine

The following table highlights the points that differentiate a Mealy Machine from a Moore Machine.

a.

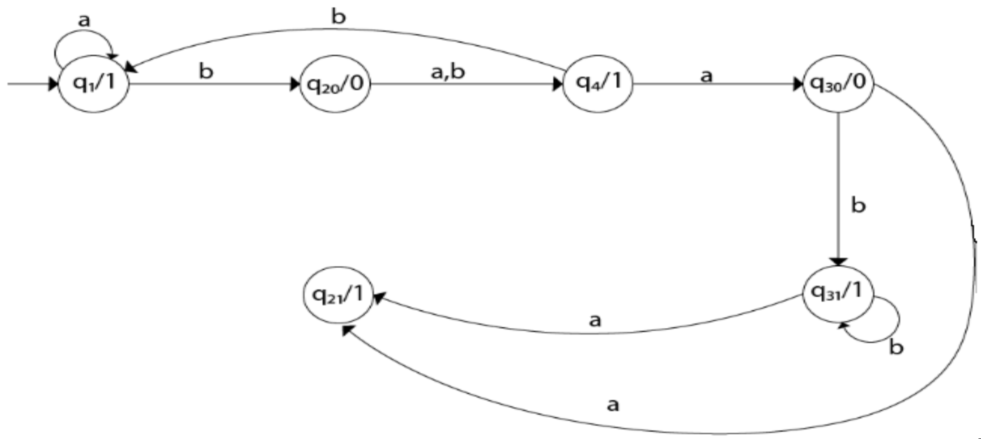
Mealy Machine	Moore Machine
Output depends both upon the present state and the present input	Output depends only upon the present state.
Generally, it has fewer states than Moore Machine.	Generally, it has more states than Mealy Machine.
The value of the output function is a function of the transitions and the changes, when the input logic on the present state is done.	The value of the output function is a function of the current state and the changes at the clock edges, whenever state changes occur.
Mealy machines react faster to inputs. They generally react in the same clock cycle.	In Moore machines, more logic is required to decode the outputs resulting in more circuit delays. They generally react one clock cycle later.

Transition table for above Mealy machine is as follows:

Present State	Next State			
	a		b	
	State	O/P	State	O/P
q_1	q_1	1	q_2	0
q_2	q_4	1	q_4	1
q_3	q_2	1	q_3	1
q_4	q_3	0	q_1	1

- For state q1, there is only one incident edge with output 0. So, we don't need to split this state in Moore machine.
- For state q2, there is 2 incident edge with output 0 and 1. So, we will split this state into two states q20(state with output 0) and q21(with output 1).
- For state q3, there is 2 incident edge with output 0 and 1. So, we will split this state into two states q30(state with output 0) and q31(state with output 1).
- For state q4, there is only one incident edge with output 0. So, we don't need to split this state in Moore machine.

Present State	Next State		Output
	a=0	a=1	
q_1	q_1	q_2	1
q_{20}	q_4	q_4	0
q_{21}	∅	∅	1
q_{30}	q_{21}	q_{31}	0
q_{31}	q_{21}	q_{31}	1
q_4	q_3	q_4	1



b.

State pumping lemma for regular set. Show that the set $L = \{ a^p | p \text{ is prime number} \}$ is not regular.

Solution: **Formal statement Pumping Lemma for Regular language:** " Let L be a regular set, then there exists a positive integer constant m such that , if ω is any word in L such that the length of ω is at least m that is $|\omega| \geq m$ and we can rewrite ω=xyz in such a way that

I) $|xy| \leq m$

II) $|y| \geq 1$, $y \neq \wedge, (y \text{ is pumped})$

III) $For\ all\ i \geq 0,$ $xy^i z \in L.$

Application of Pumping Lemma: It is used to check whether a given language is regular language or not.

Step:

I) Assume that given language L is regular language. Let m be the constant of pumping lemma as $m = |w| + 1$.

II) Take a language string ω from L such that $|\omega| \geq m$ & write ω =xyz. It means ω can be broken into three parts as x, y and z.

III) Find a suitable $i \geq 0$, $xy^i z \in L$.

Note:

In this lemma proof is done by contradiction by pigeonhole principle.

Pumping lemma should not used to proof that a given language to be regular.

Which string ω to select is very important and it is the key to solution using pumping lemma so always take **larger string**.

Suppose the statement is true, and this langauge is regular. Then there exists a FSA(finite state automaton) that recognizes this language, which we call M.

The pumping lemma says that there exists a natural number p such that for every string s in L(M) of length at least p, there is a decompositon of s=xyz such that: |y| > 0 |xy| <= p

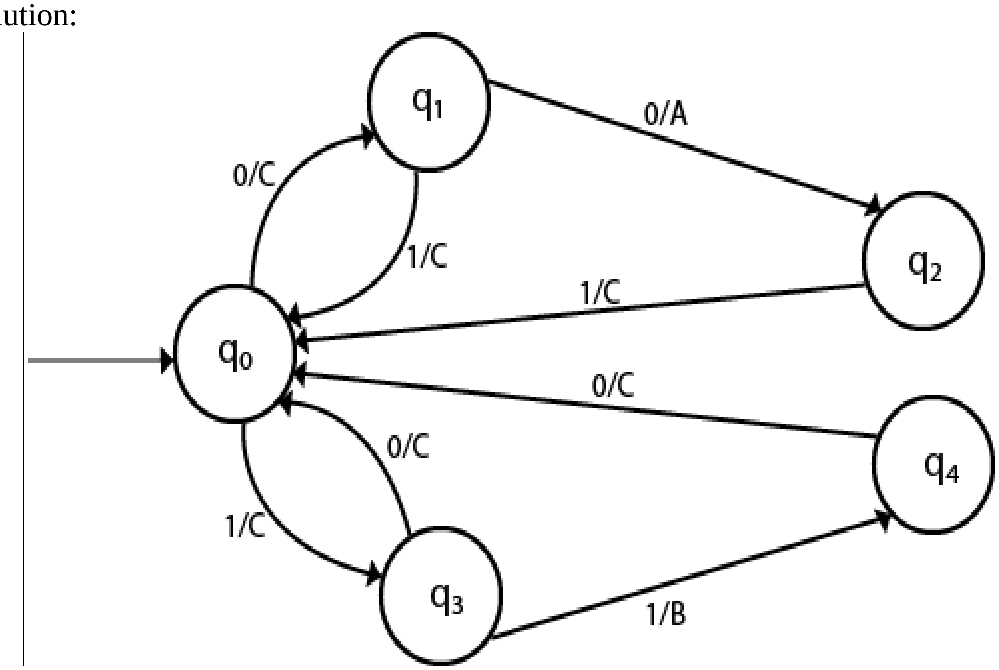
Now, we can assume that there is a string w in L(M) such that |w|=k is the first prime number greater than p since there are infinitely many prime numbers.

Because w is in L(M) and |w| > p, w can be decomposed as w=xyz that satisfies the above conditions. Now consider the string .

By the condition 3 above, v is in L(M). Thus, the length of v must be a prime number. But . Clearly, k | k(1+|y|) and k > 1. Hence |v| is not prime. This contradiction implies that the supposition is false, and the given langauge is not regular.

5.

a. Design a mealy machine that scans sequence of inputs of 0 and 1 and generates output ‘A’ if the input string terminates in 00, output ‘B’ if the string terminates in 11, and output ‘C’ otherwise.
Solution:



b. Construct the minimum state automata equivalent to DFA described by the fig.

Present state	Next State	
	Input 0	Input 1
$\rightarrow q_0$	q_1	q_2
q_1	q_3	q_4
q_2	q_5	q_6
q_3	q_3	q_4
q_4	q_5	q_6
$* q_5$	q_3	q_4

q_6	q_5	q_6

Solution: Q=$\{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}$
π_0 is collection of final and nonfinal.

$\pi_0 = (\{q_5\}, \{q_0, q_1, q_2, q_3, q_4, q_6\})$
On providing input 0 breaks into $set\{q_0, q_1, q_3\}\ and\ \{q_2, q_4, q_6\}$.
After this provide 1 to both set$\{q_0, q_1, q_3\}\ and\ \{q_2, q_4, q_6\}$ belongs to same block. So further not break.

$$\pi_1 = (\{q_5\}, \{q_0, q_1, q_3\}, \{q_2, q_4, q_6\})$$

On providing 0 and 1 it does not break so

$$\pi_2 = (\{q_5\}, \{q_0, q_1, q_3\}, \{q_2, q_4, q_6\})$$

Now

$$\pi_2 = \pi_1.$$

Present state	Next State	
	Input 0	Input 1
$\rightarrow \{q_0, q_1, q_3\}$	$\{q_0, q_1, q_3\}$	$\{q_2, q_4, q_6\}$
$\{q_2, q_4, q_6\}$	$\{q_5\}$	$\{q_2, q_4, q_6\}$
$* \{q_5\}$	$\{q_0, q_1, q_3\}$	$\{q_2, q_4, q_6\}$