Theory of Automata Practice Questions for Mid 2

Kleene Properties – Practice Questions

1. Define the Kleene star and Kleene plus operations. What is the difference between L^* and L^+?
2. Prove or disprove: $(L^*)^* = L^*$
3. Let $L = \{a\}$. Is $(L^+) = L^* - \{\varepsilon\}$? Explain.
4. If $L = \{\varepsilon\}$, what is L^* and L^+?
5. Given $L = \varnothing$ (empty set), what are L^* and L^+? Justify your answer.
6. Prove that $(L_1 \cup L_2)^* = (L_1^* \cup L_2^*)^*$
   Is this always true? Give a counterexample if not.
7. If L is regular, prove that L^* is also regular using closure properties.
8. Given a regular expression $R = (a \cup b)^*$, describe the language it represents and draw the corresponding DFA.
9. Prove that for any language L, $L^+ = L \cdot L^*$
10. Show that $(L^*) \cdot (L^*) = L^*$. Use set-theoretic reasoning or examples.

Kleene Theorem 3 Parts - Practice

Part 1: FA → RE (Finite Automaton to Regular Expression)

1. Convert the given DFA into a regular expression using state elimination.
2. Given an NFA with 3 states, derive a regular expression for the language it accepts.
3. Given a DFA that accepts binary strings ending in 01, convert it into an RE.
4. Construct a regular expression from an NFA that accepts all strings containing at least one a.
5. Eliminate states step by step from the given automaton and derive the equivalent RE.
6. Given an FA with ε-transitions, convert it to an RE using generalized transition graph (GTG).
7. Design a DFA for $L = \{ w \mid w \text{ contains exactly one } 1 \}$, then find its RE.
8. Convert a DFA accepting even length strings over $\{a, b\}$ to a regular expression.

9. Given a DFA for L = { w | w contains substring "ab" }, derive its RE.
10. Convert the minimal DFA for (a ∪ b)*abb to a regular expression.

Part 2: RE → TG (Regular Expression to Transition Graph)

11. Construct a transition graph (TG) from the RE: a*b
12. Create a TG for the RE: (ab ∪ ba)*
13. Build a TG for the RE: a(b ∪ c)*d
14. Design a TG for the RE: (a ∪ b)(a ∪ b)
15. Convert the RE: (a ∪ b)*abb into a TG with labeled transitions.
16. Show the transition graph corresponding to RE: $a^+b^*c^+$
17. Draw a TG for RE: (a ∪ ε)b*
18. Design a TG for RE: ((a ∪ b)(c ∪ d))*
19. Convert RE: a(bc)* ∪ b(ac)* to a TG.
20. Construct a transition graph for the RE: a* ∪ ba*b

Part 3: TG → FA (Transition Graph to Finite Automaton)

21. Convert the given TG to an equivalent NFA.
22. Given a TG with multiple transitions between states, simplify it into an FA.
23. Construct an NFA from a TG where transitions are labeled with REs.
24. Convert a TG with loops and unions into a deterministic FA.
25. Build a DFA from a TG accepting the language of binary strings ending with 00.
26. Show the FA for a TG where transitions are labeled with ε, a, and b.
27. Design an FA equivalent to a TG for RE: (ab)* ∪ (ba)*
28. Convert a transition graph with 4 states and labeled transitions into an equivalent NFA without ε-moves.
29. Build an FA from the TG of a regular expression containing concatenation and union operations.
30. Given a TG constructed from an RE, simplify and construct a minimal FA.

Pumping Lemma Questions

1. $L_1 = \{ a^n b^n \mid n \geq 0 \}$
   Is this language regular? Prove using Pumping Lemma.
2. *L = { w ∈ {a, b} | w contains an equal number of a's and b's }**
   Is this language regular? Use Pumping Lemma to justify.
3. $L_3 = \{ a^m b^n c^p \mid m, n, p \geq 0 \text{ and } m = n \text{ or } n = p \}$
   Is this language regular?
4. *L = { w ∈ {a, b} | w contains the substring "aba" }**
   Is this regular? Prove it using automata or Pumping Lemma.
5. $L_5 = \{ a^n b^m \mid n \neq m \}$
   Use Pumping Lemma to test if this language is regular.
6. *L = { w ∈ {a, b} | w is a palindrome }**
   Prove whether the language is regular or not using Pumping Lemma.
7. $L_7 = \{ a^n b^n c^n \mid n \geq 1 \}$
   Is this language regular? Prove or disprove using the Pumping Lemma.
8. $L_8 = \{ a^n b^m \mid n, m \geq 0 \}$
   Use Pumping Lemma to verify whether the language is regular.
9. *L = { w ∈ {0,1} | w has more 1s than 0s }**
   Is this language regular? Justify your answer.
10. $L_{10} = \{ a^n b^n c^m \mid n, m \geq 0 \}$
    Use Pumping Lemma to determine if the language is regular.

Mealy Machine Construction – Questions, Construct and then convert it into Moore Machine

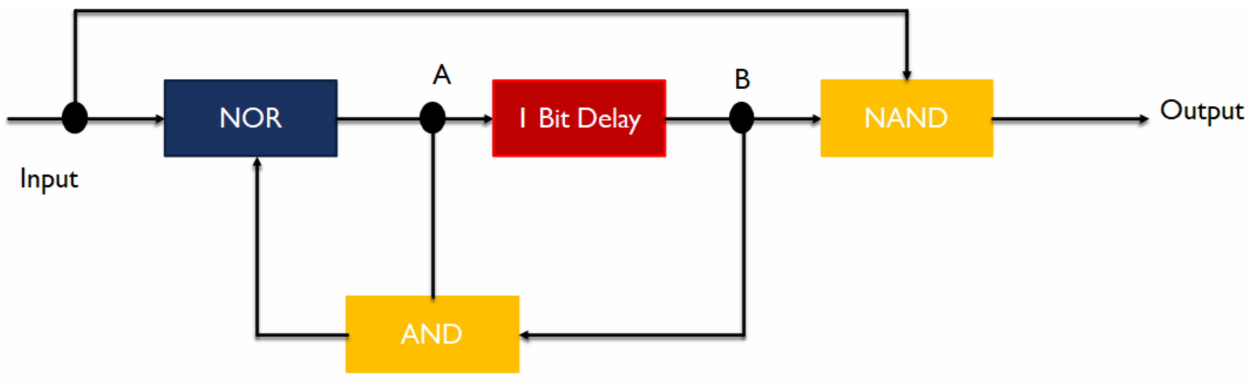
1. Construct a Mealy Machine that outputs `1` when the input symbol is `a`, and `0` when the input symbol is `b`.
2. Design a Mealy Machine that outputs `1` if the current input is the same as the previous input, otherwise outputs `0`. Assume the first input always outputs `0`.
3. Build a Mealy Machine that detects the substring "ab" and outputs `1` upon detection, `0` otherwise. Over the alphabet {a, b}.
4. Construct a Mealy Machine that, for a binary input string, outputs the complement of each bit (i.e., `0` becomes `1` and `1` becomes `0`).
5. Design a Mealy Machine that counts the number of `1`s modulo 2 in a binary string and outputs `0` or `1` based on whether the count is even or odd.

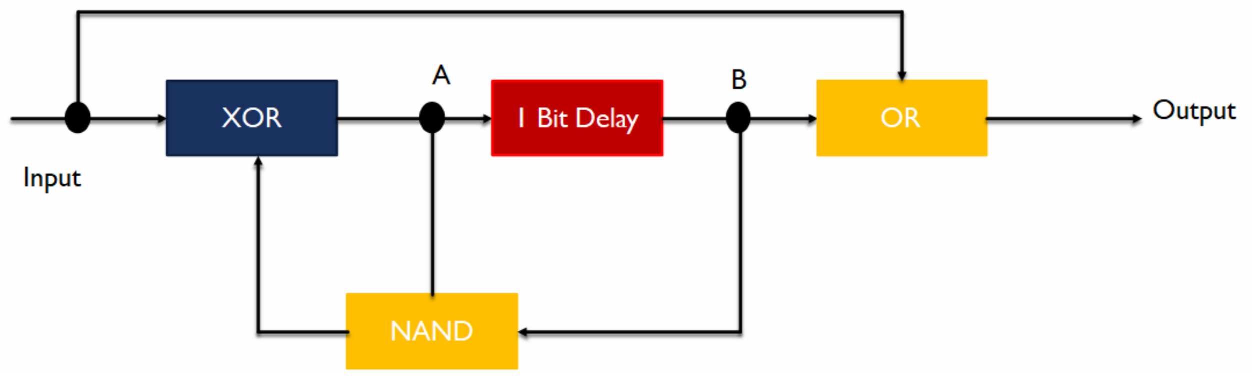
6. Build a Mealy Machine that outputs `1` if the input symbol is different from the previous one, otherwise outputs `0`.
7. Create a Mealy Machine that accepts strings over {a, b} and outputs `1` only when an `a` is immediately followed by a `b`. Otherwise, output `0`.
8. Construct a Mealy Machine that outputs `1` every third symbol regardless of input, and `0` otherwise.
9. Design a Mealy Machine over input {a, b} that outputs the total number of a's seen so far modulo 3.
10. Construct a Mealy Machine that outputs `1` if the total number of `a`s seen so far is even, and `0` if odd.

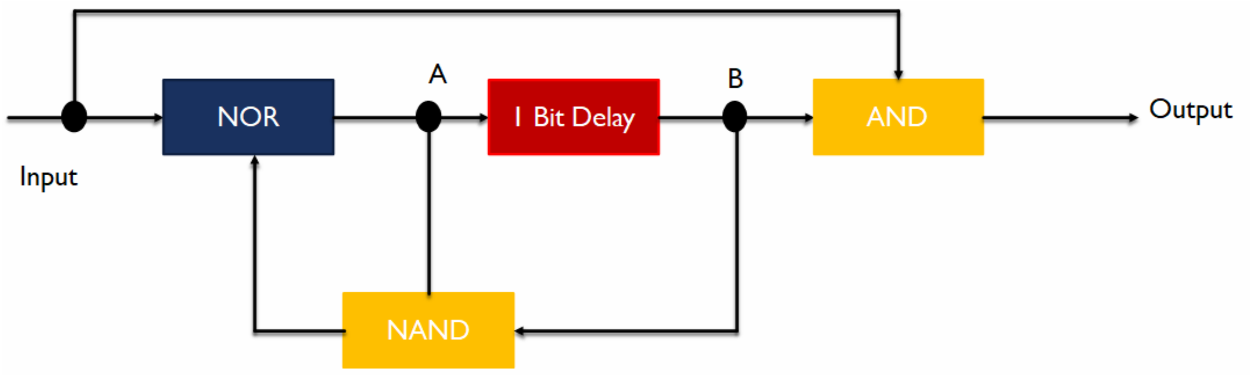
Moore Machine – Construct them and then convert it into Mealy Machine

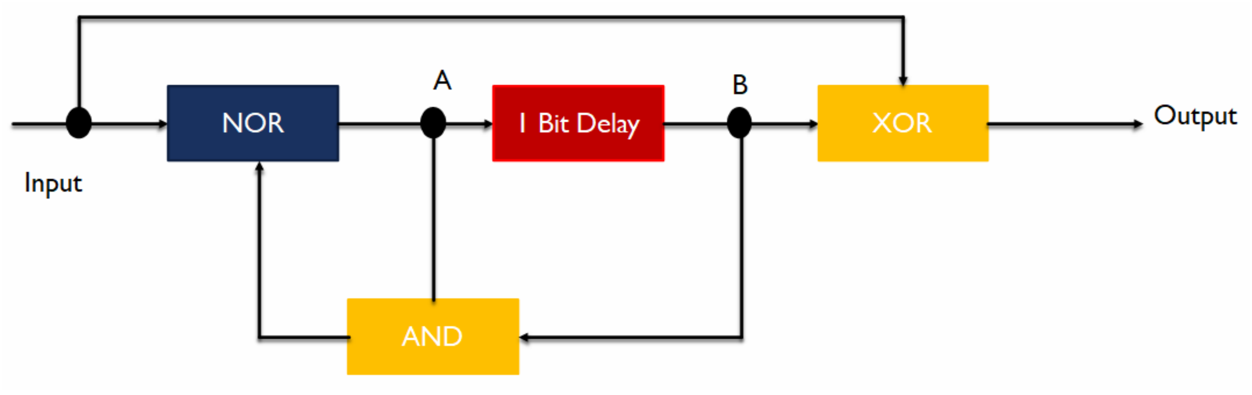
1. Construct a Moore Machine that outputs `0` on the initial state and outputs `1` once it reads an `a`, regardless of any further input.
2. Design a Moore Machine that outputs `0` until it reads the substring "ab", after which it continuously outputs `1`.
3. Create a Moore Machine over input {0, 1} that outputs `1` if the last two inputs were `1 1`, and `0` otherwise.
4. Build a Moore Machine that outputs the number of `a`s seen so far modulo 2 (i.e., output is `0` if even number of `a`s seen, `1` if odd).
5. Construct a Moore Machine to detect the sequence `101` in a binary input string and outputs `1` in the state after recognizing it; otherwise, output `0`.
6. Design a Moore Machine over input {a, b} that outputs `1` if the current state corresponds to having seen more `a`s than `b`s, otherwise `0`.
7. Build a Moore Machine where the output is `1` for every third input symbol, regardless of the actual input symbol.
8. Construct a Moore Machine that outputs `1` if the input string ends with "ba", and `0` otherwise.
9. Create a Moore Machine over input {a, b} such that it outputs `0` until it sees the pattern "aaa", after which it always outputs `1`.
10. Design a Moore Machine that outputs the parity (even or odd) of the total number of 1's seen in a binary input stream. Output `0` for even, `1` for odd.

Transducers – Solve the following and convert them into Mealy Machines then convert them into corresponding Moore Machines

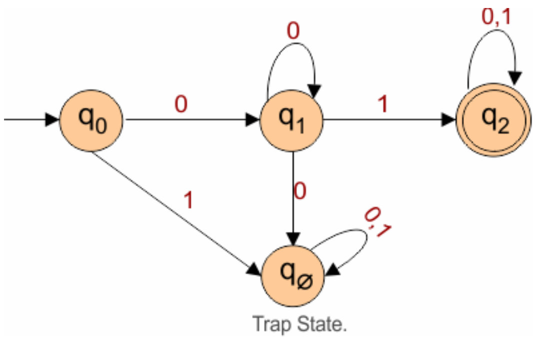


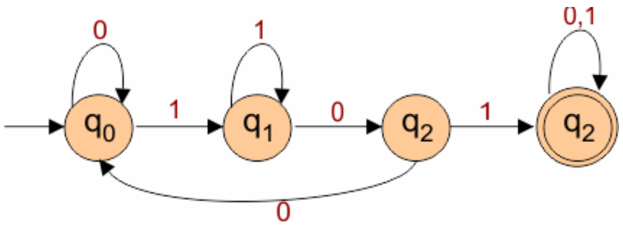


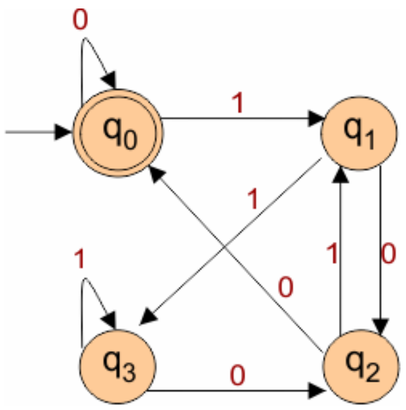


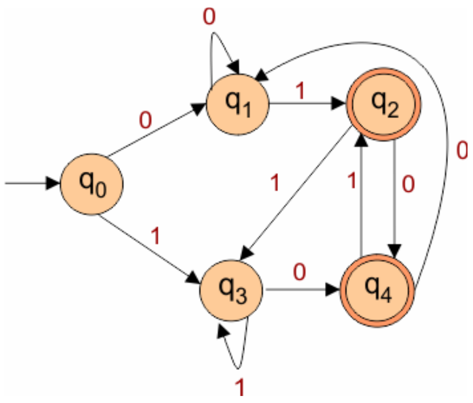


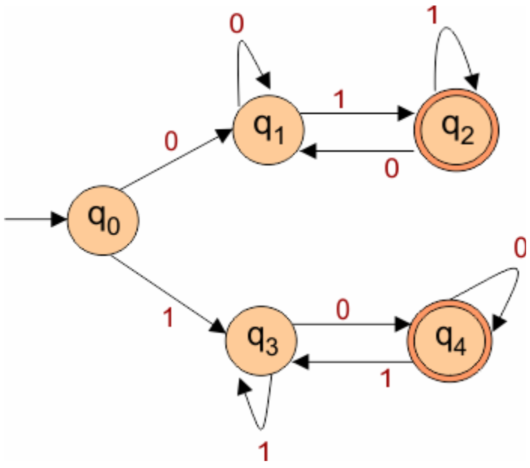
DFA Minimization using either My-Hill Nerode Theorem or by partitioning Method

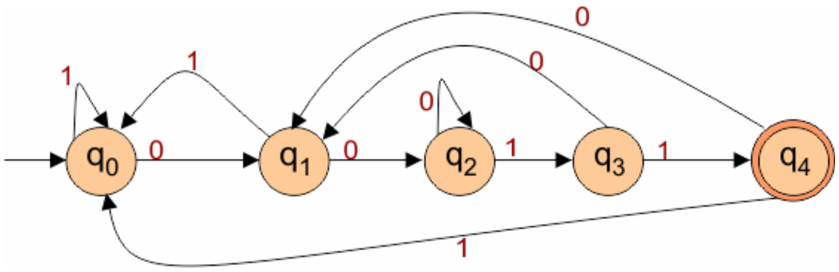






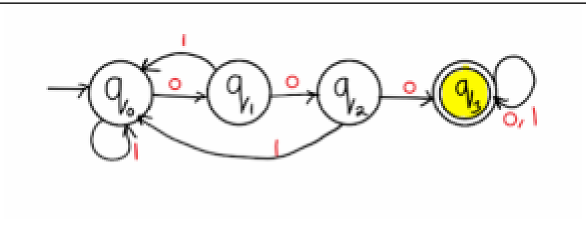
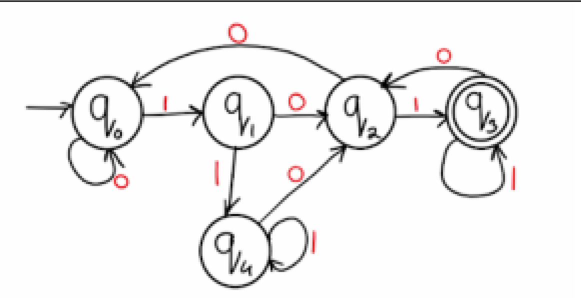
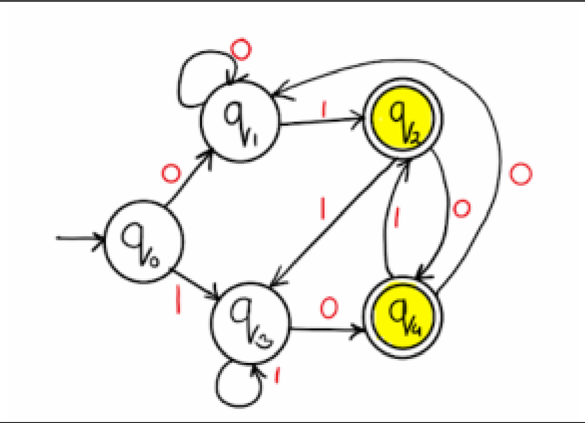
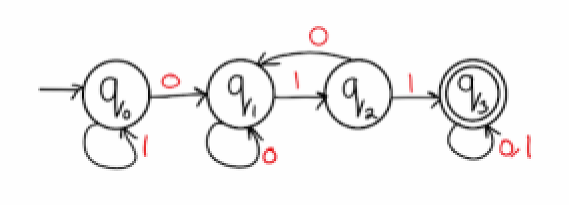
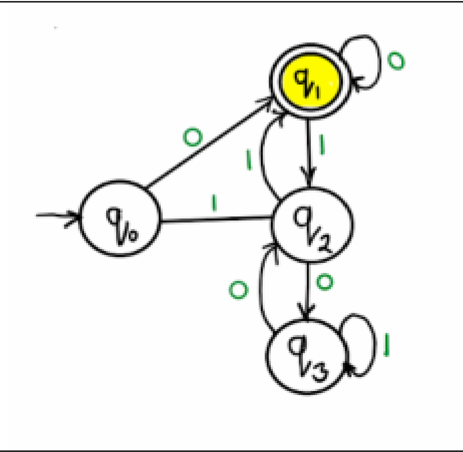
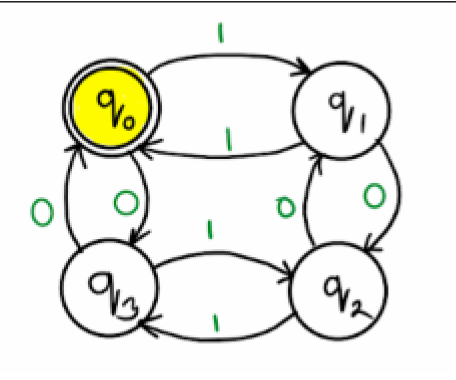
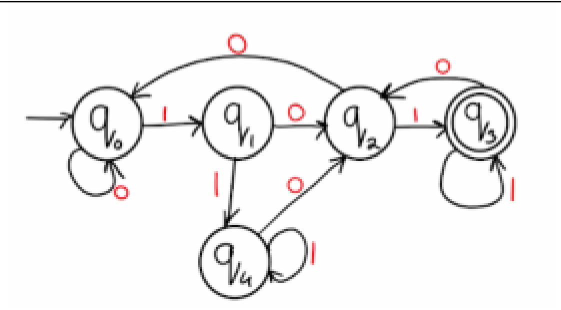
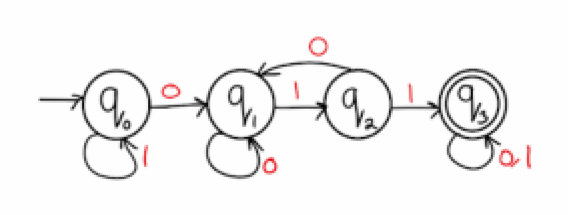


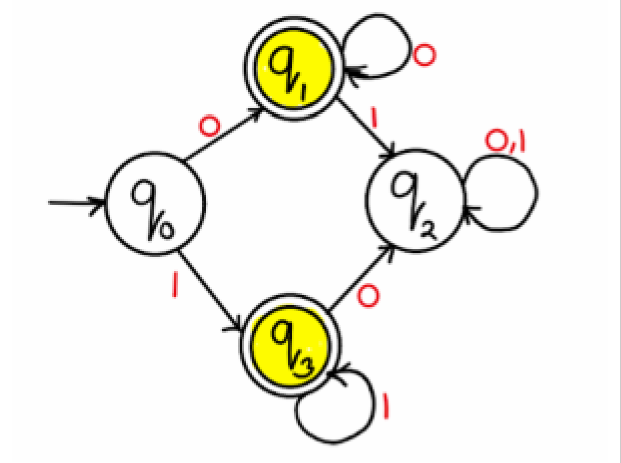
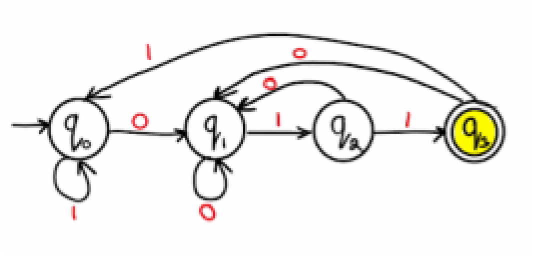




Perform Union, Intersection, Compliment, Product and Concatination for the following Machines (M1 and M2)

1. Now first analyze the Machine 1 and write the behavior with 5 accepted strings. Do same for Machine 2.
2. Now test them for the Union, Intersection, Compliment, Product and Concat Machines.
3. Prove your solution

Machine 1	Machine 2
	
	
	
	

Extra Practice – From RE to FA , then DFA then Minimization

Construct the DFA from the following and then perform DFA minimization using any method

1. DFA for (a+b)* (a+b)a .
2. DFA for (bb)*(aa)* .
3. DFA for b+a(a+b)*+a.
4. DFA for (a+b)*b+(bb)*a.
5. DFA for bb+a(a+b)*+aa.
6. DFA for a(a+b)*+bb(a)* .
7. DFA for a(a+b)b*+bb(a)*.
8. DFA for b(aa)*a+a(bb)*b.
9. DFA for a+a(aa+b)*(aa)b.
10. DFA for a+a(aa+b)*+(aa)b.
11. DFA for (a+b)b(a+b)*+(aa)*b.
12. FA for strings starting with a and ending with a.
13. FA for the language of all those strings starting with a.
14. FA for the language of all those strings containing aa as a substring.
15. DFA for the language of all those strings starting and ending with the same letters.
16. DFA for the language of all those strings starting and ending with different letters.

17. DFA for the language of all those strings having double 0 or double 1.
18. DFA for the language of all those strings starting and ending with b.
19. DFA for ending with b.
20. DFA for the string of even A's and even b's.

Grammar

Write the productions for the given grammar and generate the Total Language tree for each case. Write RE for each case (if possible) based on the productions.

Sno	Production Rules
1	$S \rightarrow X$ $S \rightarrow Y$ $X \rightarrow \lambda$ $Y \rightarrow aY$ $Y \rightarrow bY$ $Y \rightarrow a$ $Y \rightarrow b$
2	$S \rightarrow aS$ $S \rightarrow bS$ $S \rightarrow a \mid b \mid \lambda$
3	$S \rightarrow AA$ $A \rightarrow AAA \mid bA \mid Ab \mid a$
4	$S \rightarrow SS$ $S \rightarrow XXX$ $X \rightarrow aX \mid Xa \mid b$
5	$S \rightarrow XaXaX$ $X \rightarrow aX \mid bX \mid \lambda$