Automata Theory Assignment 1

Submission by : 2023113019

Ans 1:

For each state, the number of states it can transition to = 5[including itself]. For each symbol, and an extra epsilon, we can have a total of 4 different symbols which can transition to each of these 5 states.

Thus, for each state, there are 4*5 = 20 transitions that can be defined.

Since number of states = 5, total transitions in δ possible = 20*5 = **100.**

Ans 2:

Consider p $\in \Sigma \setminus \{\%\}$

- x/y denotes that x is read from the input and y is written to the output.
-

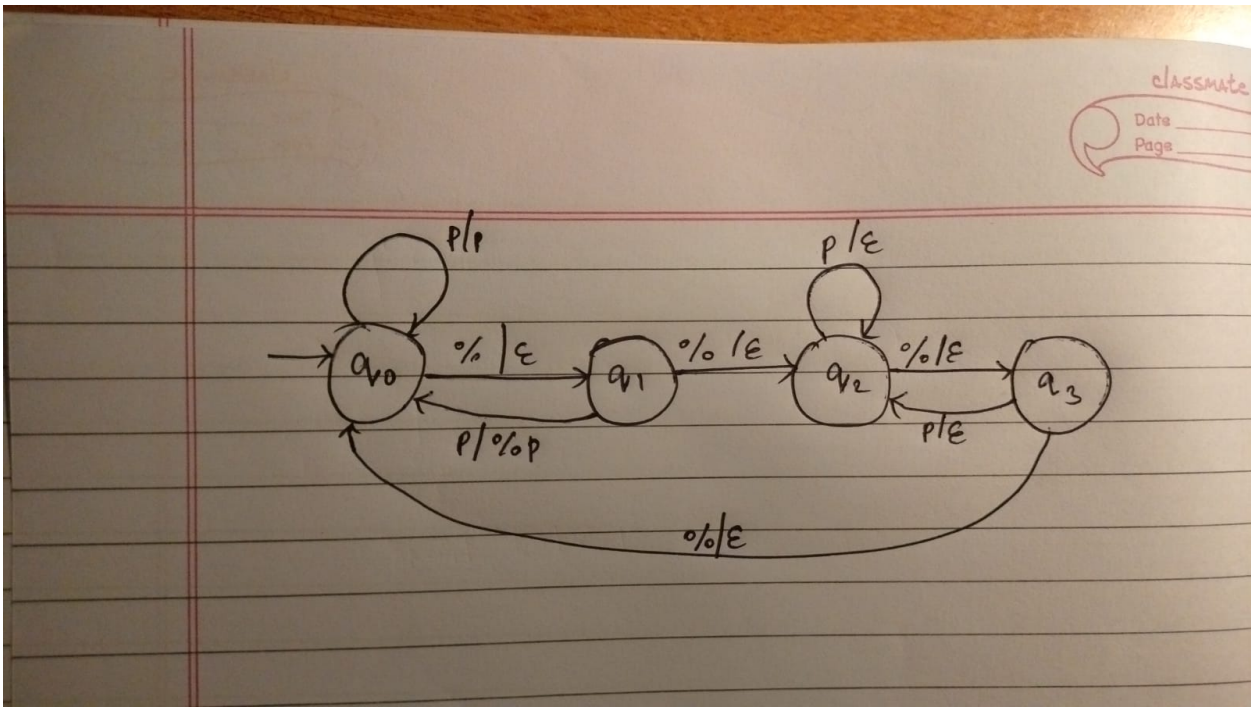
ϵ is taken to be an empty string.

-

Γ is taken to be $\Sigma \cup \{\sigma1\sigma2 \mid \sigma1, \sigma2 \in \Sigma\} \cup \epsilon$. The reason for this definition of Γ is because at times we do need to have more than one character printed to the output, such as “%s” on just one account. We could have put $\Gamma = \Sigma^*$ but that would have included a lot more information than what we require.

Assuming the program does not end with a single “%” {to include it would involve non determinism in the output}

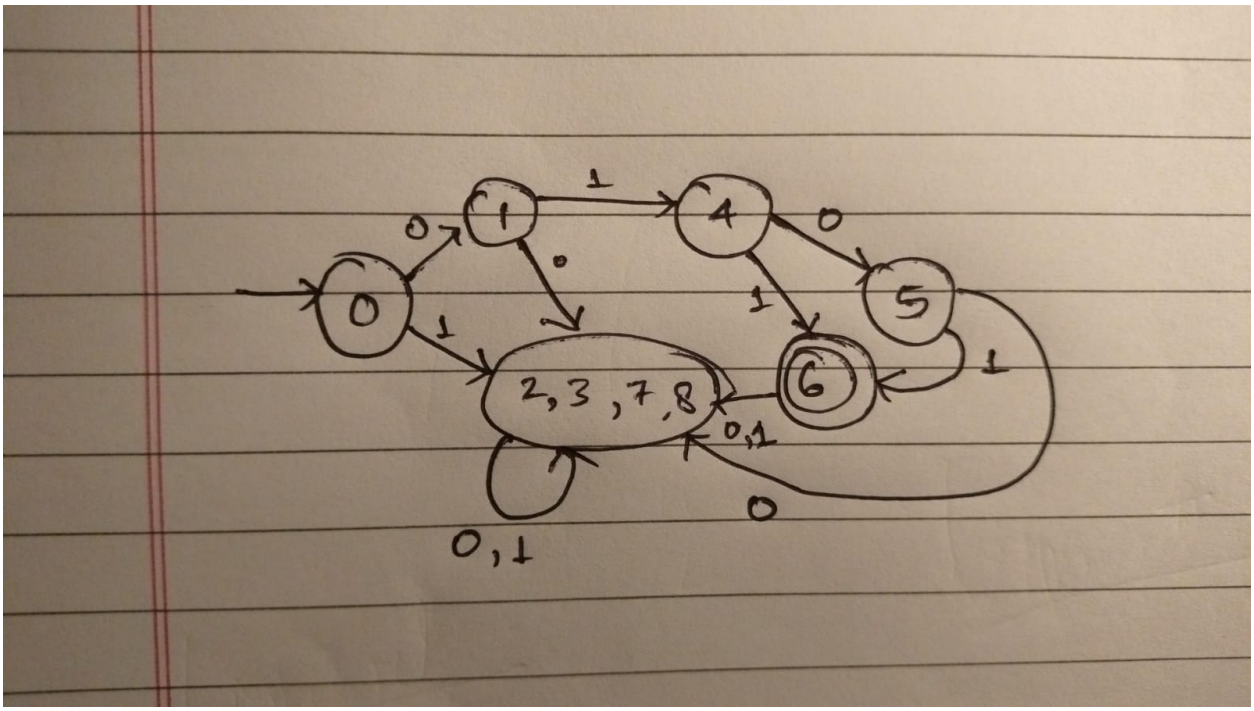
For δ function, we can use the following diagram



Thus, the FST $M = \{q_0, q_1, q_2, q_3\}, \Sigma, \Gamma, \delta, q_0\}$

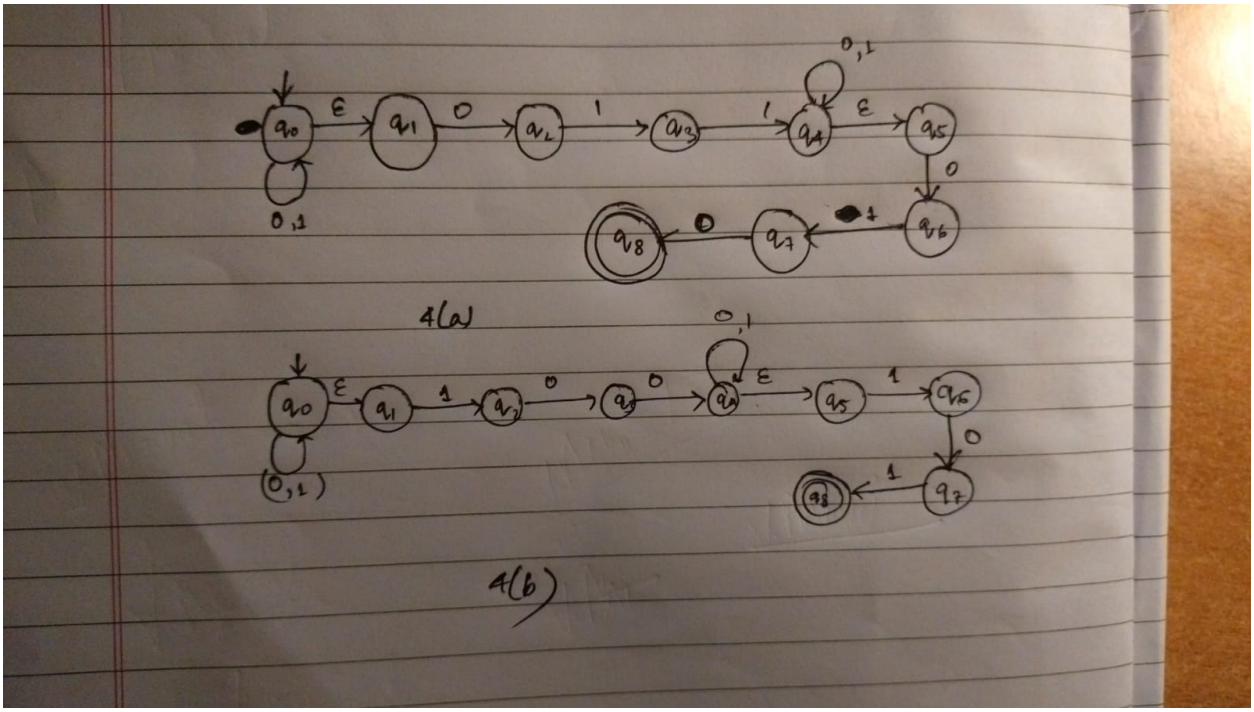
Ans 3:

For doing this, many algorithms can be used, like the Myhill Nerode method, however, in this particular question, pure observation is sufficient to minimize the DFA. We use the fact that 8 and 7 are non-final states, that have no outdegree(always give false when it reaches there). We also use the fact that 2 and 3 once reached, stay there or go to 8, again getting stuck in a non-final state. Hence, as far as the functioning of this Automaton is concerned, 2,3,7,8 once reached, give us the same result, and can hence be considered to be a single “macro-state”.



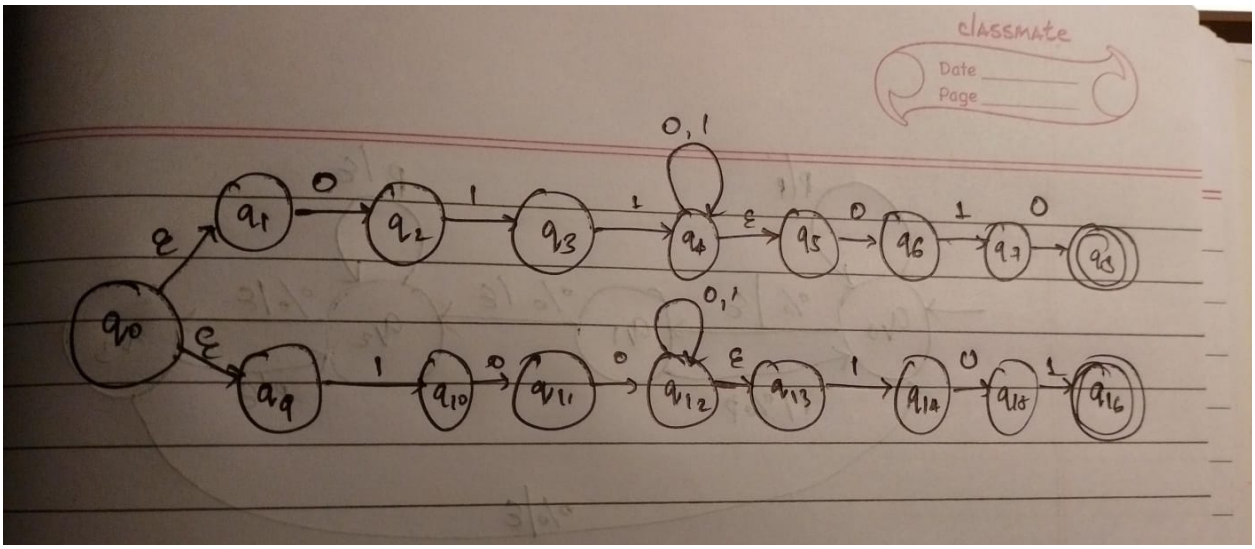
Ans 4:

If done separately, the FSMs would look like:



We combine them using epsilon transitions (as done commonly in union of two Languages), and finally get this combined NFA:

We take δ from this diagram:



Formally,

$$Q = \{q_i \forall i \in \mathbb{N} : 0 \leq i \leq 16\}, \Sigma = \{0, 1\}$$

Hence, the NFA $M = (Q, \Sigma, \delta, q_0, \{q_8, q_{16}\})$

Ans 5:

(i) $L(m) = ab(bab + bb)^* a$

(ii) $L(m) = ((e + a + aa)b)^* aaa(b(e + a + aa))$

Ans 6:

RTP If A is regular, $A_{1/3-1/3}$ is not regular

Proof by counter-example:
Consider the regular language
$L = 0^*10^*10^*$.

If we take $A_{1/3-1/3}$, we know that strings of the form 0^n110^n can be a part of the language if we take the string $s = 0^n10^{n+1}10^n$ from L and split it into equal thirds such that $s = xyz, x = 0^n1, y = 0^{n+1}, z = 10^n$

Strings of the form $0^a110^b, a \neq b$ can not be a part of $A_{1/3-1/3}$ as splitting in equal thirds ensures that the number of 0s before and after 11 are the same. A Finite State Automata can not distinguish between 0^n110^n and $0^a110^b, a \neq b$, therefore the resulting language is irregular.

Ans 7:

Consider a DFA M with k states, let α be the shortest string in $L(M)$. We notice that the minimum accepting run can reach the final state by taking at max $k-1$ transitions. We can then say $|\alpha| \leq k-1$. We shall be using this property in our proof:

Let's take s to be the shortest string in $U = L(M_1) \cup L(M_2)$ and $k = max(k_1, k_2)$.

Since $s \in L(M_1)$ or $s \in L(M_2)$, we can infer the following two cases:

Case 1: $s \in L(M_1) \implies |s| \leq k_1 - 1 \implies |s| < k_1 \leq k$

Case 2 : $s \in L(M_2) \implies |s| \leq k_2 - 1 \implies |s| < k_2 \leq k$

Hence, in both the cases, $s < k$.

Ans 8:

Part 1

Let us assume that L is regular.

According to the pumping lemma this implies $\exists p \geq 1$ such that for $a \in L, a = xyz$ where:

1. $|y| \geq 1$
2. $|xy| \leq p$
3. $\forall n \geq 0, xy^nz \in L$

Taking $w = (^p)^p \implies y = (^n$. Hence, xy^0z must be in L. However $xy^0z = (^{n-p})^n$ is an unbalanced bracket sequence and hence we arrive at a contradiction.

Therefore, L is not a regular language.

Part 2

Let us assume that L is regular.

According to the pumping lemma this implies $\exists p \geq 1$ such that for $a \in L, a = xyz$ where:

1. $|y| \geq 1$
2. $|xy| \leq p$

3. $\forall n \geq 0, xy^nz \in L$

Taking $w = a^{p!} \implies y = a^i, i \in \mathbb{N}, 1 \leq i \leq p$.

According to the pumping lemma, $xy^2z = a^{p!+i} \in L$

But, we know:

$$a^{p!+i} = a^{k!} \implies p! + i = k! \implies p! + i \geq (p+1)! \implies p! + i \geq p! \times (p+1)$$

However we have $1 \leq i \leq p$, so $p! + i < (p+1)!$ Hence we arrive at a contradiction. Therefore the given language L is not regular.

Ans 9:

From the given transitions, we observe that there are no outgoing transitions from S, U, W, Y and Z. Hence, S, U, W, Y, Z are all terminals.

R, T, V, X are the variables, since all variables need to be mapped to a terminal.

Since no variable goes to R, it can be taken as the start state. Moreover, we notice that taking any other variable as the start state would render R unreachable, whereas all are reachable if R is the start state.

Ans 10:

The language A can be thought of as the union of two languages L_1 and L_2 such that

$$L_1 = \left\{a^i b^i c^k \mid i, k \geq 0\right\}, L_2 = \left\{a^i b^k c^k \mid i, k \geq 0\right\}$$

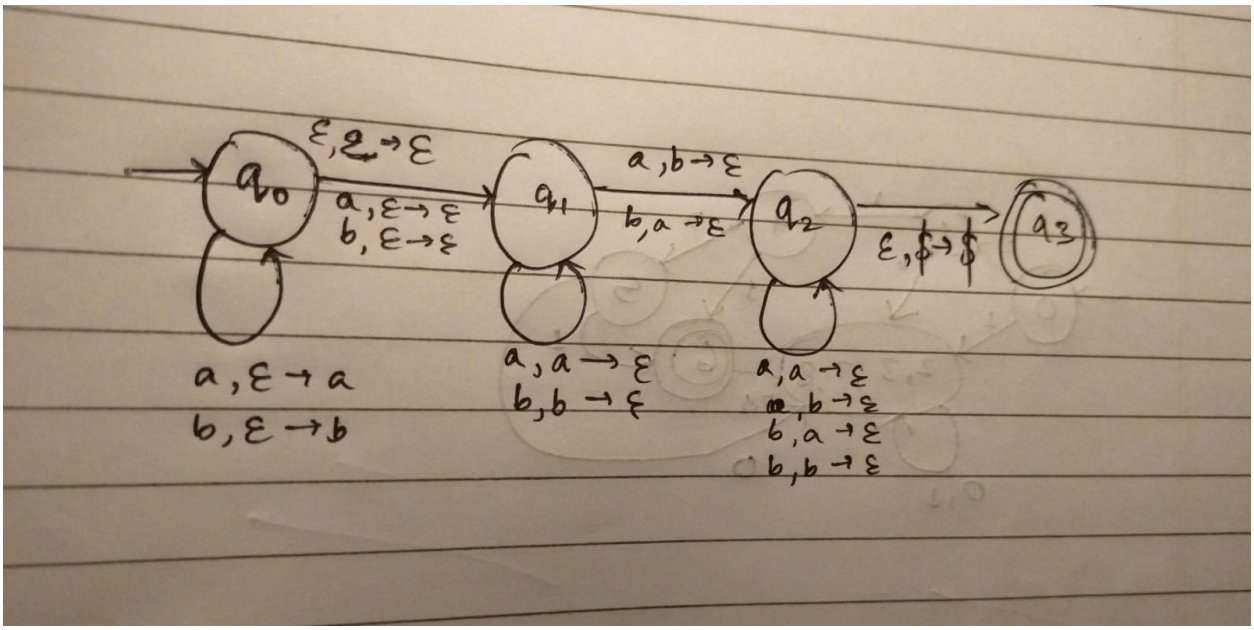
We can now construct the grammar as follows with S as the starting variable:

$$\begin{gathered} S \to S_1 \mid S_2 \\ S_1 \to AB \\ A \to aAb \mid \epsilon \\ B \to Bc \mid \epsilon \\ S_2 \to CD \\ C \to aC \mid \epsilon \\ D \to bDc \mid \epsilon \end{gathered}$$

This grammar, however is ambiguous. This is because strings can be generated in multiple ways. For example, the string abc can be generated in two ways by left-most derivation:

$$S \to S_1 \to AB \to aAbB \to abB \to abBc \to abc$$
$$S \to S_2 \to CD \to aCD \to aD \to abDc \to abc$$

Ans 11:



In the PDA shown above, q_0 state processes first half of the input, after which it goes to state q_1 (in case the length of the input is odd, it accounts for any middle element). When there is any mismatch between the stack and second half of the input, it transitions to q_2. On q_2 it is verified if $q_1$was visited, only after half of the input has been processed by popping elements from the stack after processing the inputs. Then it transitions to a final state q_3 when the stack is empty.

Ans 12:

In the given PDA, it transitions from q_0 to q_1 after putting a # in the stack. Then it reads a from the input and pushes an A symbol to the stack for each a read. It then goes to state q_2 and pops an A from the stack for every b read from the input. It goes to the final state q_3 after popping an A from the stack and pushing an A to the stack.

Therefore, we can say the language of the given PDA is

$$L(M) = \{a^i b^j : i > j \geq 0\}$$

If the total length of the input is 100, we can have $51 \leq i \leq 100$ for if $i < 51$, it would make $j \geq i$. Therefore, the total number of valid strings is $100 - 51 + 1 = 50$.