Solution 1:

Correct Answer: (C)

We will show how to construct the NFA for $L_1$. Let $A$ be the DFA for $L$. Consider the automaton $A'$ where all the edges of $A$ are reversed. Let $A''$ be an automaton with same states as $A$ and we make a transition from a state $p$ to $q$ for every symbol in the alphabet if there exist a two length string $w$ such that $\hat{\delta}(p, w) = q$ in $A'$. Let $D$ be the product automaton of $A$ and $A''$ with a new initial state $q_{start}$ and create epsilon transitions from $q_{start}$ to all the states $q_i$ where $q_i$ is a final state of $A$. In $D$ our final states be all the states of the form $q_iq_j$. It is easy to see now that $L(D) = L_1$.

Let's define $L^R_1$ for $L$ exactly as we defined $L_1$ for $L$. Since $L_2 = (L^R_1)^R$ therefore $L_2$ is also regular.

Solution 2:

Correct Answer: (B)

Let $L$ be a regular language with DFA $A$. Then we can make an NFA for $\text{shift}(L)$ as follows. Let $B_0$ be a DFA same as $A$ and for every state $q_i$ in $A$ except the initial state create an NFA $B_i$ by taking two copies of $A$ and making $\epsilon$ transitions from final states of first copy to initial state of second copy and make $q_i$ in the first copy the initial state of $B_i$ and $q_i$ in the second copy the only final state of $B_i$. Now using all the $B_i$s we can make the NFA by creating a initial state $q_{start}$ and making $\epsilon$ transitions to the initial state of all $B_i$s. The resulting NFA’s language is $\text{shift}(L)$.

Consider the language $L = \{0^n \mid n \geq 1\}$. Then $\text{Incrpad}(L) = \{0^n0^{n(n+1)/2} \mid n \geq 1\}$ which can be proved non-regular using pumping lemma.

Let $L$ be a regular language with DFA $A$. Then we can make an NFA for $\text{unipad}(L)$ as follows. Replace every state $q_i$ of $A$ except the initial state into two states $q'_i$ and $q''_i$ such that there is an edge labeled with $s$ from $q'_i$ to $q''_i$ and all the edges entering $q_i$ enters $q'_i$ and all the edges going out of $q_i$ now go out from $q''_i$. Apart from that delete all the loops labeled with $a$ on initial state $q_0$ and create a new state $p_a$ such that there is an edge labeled with $s$ from $p_a$ to $q_0$ and an edge from $q_0$ to $p_a$ labeled with $a$. Finally, make all states $q''_i$ final if $q_i$ was the final state.

Solution 3:

Correct Answer: (A)

$L$ is be the language of binary strings where every 0 is followed by a 1. Clearly complement of this language is $1^*0^*$. So the minimum DFA for the complement of the language will be as following:

![Diagram of DFA for complement of $L$]

Minimum NFA for the same language will be as following.

![Diagram of NFA for complement of $L$]

Solution 4:

Correct Answer: (C),(D)
Grammar G is as follow:

\[
S \rightarrow aA | bA | a | b \\
A \rightarrow aB | bB | a | b \\
B \rightarrow aS | bS 
\]

Clearly A is not a correct choice since G does not generate \( \epsilon \). Let us try to generate a string \( w = a_1a_2...a_n \) (where \( a_i \in \{a,b\} \)) using this grammar. In first step we can get string of type either \( a_1A \) or \( a_1 \) (where \( a_1 \in \{a,b\} \)). At this step we can stop (by choosing \( a_1 \)) which will give all string of length 1 or we can continue with string \( a_1A \). Now from \( a_1A \) we generate two type of string \( a_1a_2B \) or \( a_1a_2 \) (where \( a_2 \in \{a,b\} \)). Again we can stop here by choosing string \( a_1a_2 \) which will give us all string of length 2 or we can continue with string \( a_1a_2B \). Now at this point from \( a_1a_2B \) we can get string only of type \( a_1a_2a_3S \). Hence we can not get string of length 3 (since there is no production of type \( S \rightarrow \epsilon \)). Now we continue expanding \( S \) present in the string \( a_1a_2a_3S \) in same way as we started. Ultimately we can not get any string \( w \) such that length of not divisible by 3.

From above conclusion option B cannot be correct, because according to option B, string of length 3 should be generated. Option D is correct because we know that the language \( L = \{w \mid w \in \{a,b\}^*, |w| \text{ is divisible by } 3 \} \) is regular and \( L(G) = L \). Hence \( L(G) \) is also regular.

Solution 5:

Correct Answer: (A),(D)

Languages A, B and C are as follow.

\[
A = \{a^n b^n \mid n \geq 0 \} \\
B = \{a^n \mid n \geq 0 \} \\
C = \{b^n \mid n \geq 0 \} 
\]

Clearly Language B and C regular. So option (A) is correct.

Language B.A can be described as \( B.A = \{a^m b^n \mid m \geq n \geq 0 \} \), which we know is not regular (can be proved using pumping lemma). Similarly we can show that C.A is also non-regular.

We can describe \( A.B.C = \{a^n b^m a^n b^k \mid n,m,k \geq 0 \} \), which is also non-regular (can be shown using pumping lemma).

We can describe \( B.A.C = \{a^n b^n \mid n,m \geq 0 \} \), which is regular.

Solution 6:

Correct Answer: (B)

Grammar does not generate string a, hence option (A) and (D) are not correct. Grammar also does not generate \( ab \) hence option (C) is also not correct.

To show option (B) is correct use the same technique described in solution 4. \( S \rightarrow \epsilon \) generates string of length 0. Grammar can not generate string of length 1. We can reach to these four type of string \( aaS, aa, bbS \) and \( bb \). S present in \( aaS \) and \( bbS \) can again be replace with any of \( aa, bb, aaS \) and \( bbS \) and ultimately replace \( S \) with \( \epsilon \) to terminate. So the resultant string will be the combination of substrings \( aa \) and \( bb \), which is nothing but \( (aa + bb)^* \).

Solution 7:

Correct Answer: (B),(D)

We know that \( L_1 \) is regular and \( L_2 \) is not regular. It is also clear \( L_2 \subseteq L_1 \).
\[
\text{if string is just } a \text{ of } 2. \text{ So we can write Case 2: }
\]

**Solution 8:**

**Correct Answer:** (B),(C),(D)

\(A \cap \overline{A} = \emptyset\), hence \(A \cap \overline{A}\) is regular.

\(A^*\) can be described as \(A^* = \{a^n | n > 0\}\), because any string of \(a^n\) can be written as \(n\) concatenation of string \(a^2\). Hence \(A^*\) is also regular.

\(A.\overline{A}\) can also be described as \(A.\overline{A} = \{a^n | n > 0\}/\{aaa\}\).

Take any string \(a^n\) for \(n > 3\).

**Case 1:** If \(n\) is odd then write \(n = 2 + (n-2)\). If \(n\) is odd then \(n-2\) is also odd. \(n-2\) can not be written as a power of 2. So we can write \(a^n\) as a concatenation of \(a^{2^j}\) and \(a^{n-2}\), where \(a^{2^j} \in A\) and \(a^{n-2} \in \overline{A}\).

**Case 2:** If \(n\) is even then write \(n = 1 + (n-1)\). \(n-1\) is odd no.. So it can not be written as a power of 2. So we can write \(a^n\) as concatenation of \(a^{2^j}\) and \(a^{n-1}\), where \(a^{2^j} \in A\) and \(a^{n-1} \in \overline{A}\).

if string is just \(a\), write it as \(a^{2^0}.\epsilon\), where \(a^{2^0} \in A\) and \(\epsilon \in \overline{A}\).

if string is \(aa\) write it as \(a^{2^1}.\epsilon\), where \(a^{2^1} \in A\) and \(\epsilon \in \overline{A}\).

Hence \(A.\overline{A}\) is also regular.

We need the following observation to prove next part.

**Case 3:**

Let \(m\) is some number such that \(m\) can be written as \(m = 2^{j_1} + 2^{j_2} + \ldots + 2^{j_k}\), where all \(j_s\) are distinct then \(m\) cannot be written in another way, as a sum of distinct powers of 2, different from above. Because notice that writing a number as sum of distinct powers of 2 gives a way to write that number in binary form. So if there are two such way then this imply that \(d\) can be written in two different way in binary, which is not possible. Binary representation of a number is unique. Now we are ready to understand the proof.

Now we will show that \(A.A\) is not regular using pumping lemma. Assume \(A.A\) is regular and \(p\) be the pumping length given by pumping lemma. Now take a string \(w = xyz = a^{2^q+2^{q'}}\) where \(q' > q > p\).

We know that \(0 < |y| \leq p\). Let \(|y| = l, l\) can be uniquely written as sum of power of 2 as follow:

\[l = 2^{i_1} + 2^{i_2} + \ldots + 2^{i_k},\]

where all \(i_s\) are distinct.

Now consider the string \(x y y z\). If \(A.A\) is regular then this string should belong to the language i.e we need to show that \(|x y y z|\) can be written as sum of two powers of 2. Since \(|x y z| = 2^q + 2^{q'}\) and \(|y| = 2^{i_1} + 2^{i_2} + \ldots + 2^{i_k}\), let \(w' = x y y z\)

\[|w'| = d = 2^{q'} + 2^{i_1} + 2^{i_2} + \ldots + 2^{i_k},\]

where \(q' \neq q' \neq l, \text{ for } 1 \leq i \leq k\).

So this gives a unique binary representation of \(d\). From this we can say that \(d\) can not be written as sum of just two powers of 2. Because assume it can be written as sum of two powers of 2 then we have
two cases.

**Case 1:** \( d = 2^u + 2^v \) where \( u \neq v \). This is not possible because it will give another binary representation of \( d \).

**Case 2:** \( d = 2^u + 2^v \) where \( u = v \). So \( d = 2^u + 2^u = 2^{u+1} \), this again gives another binary representation of \( d \) which is not possible.

So we conclude that \( d \) can not be written as sum of two power of 2. Hence \( A.A \) is not regular.