PES University, Bangalore

(Established under Karnataka Act No. 16 of 2013)

UE19CS205

SAMPLE PAPER-I SOLUTION FOR

IN SEMESTER ASSESSMENT (ISA-1)- B.TECH III SEMESTER

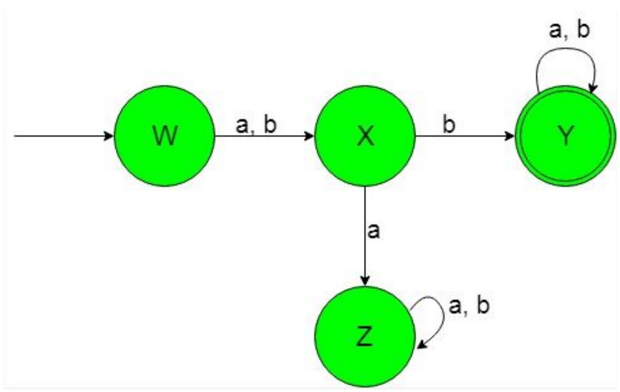
October, 2020

Automata Formal Languages & Logic

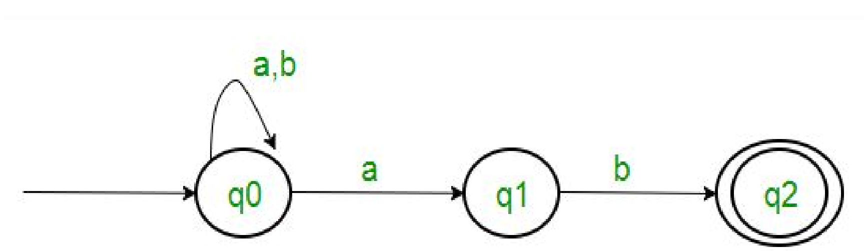
Time: 2 Hrs | Answer All Questions | Max Marks: 60

1. a **Construct a DFA accepting a set of strings over {a, b} in which the second symbol from the left-hand side is always ‘b’.** (4)

Solution: The language contains the following strings:
L = {ab, aba, abaa, bb, bb, bbbbb…..}
Here, we can see that each string of the language contains ‘b’ as the second symbol from the left-hand side.



b **Convert the following NFA to its equivalent DFA using subset construction method.** (6)



Solution:-

δ' (Transition Function of DFA)

State	a	b
q0	{q0,q1}	q0

Now { q0, q1 } will be considered as a single state. As its entry is not in Q', add it to Q'.
So Q' = { q0, { q0, q1 } }

Now, moves from state { q0, q1 } on different input symbols are not present in transition table of DFA, we will calculate it like:
δ' ({ q0, q1 }, a) = δ (q0, a) ∪ δ (q1, a) = { q0, q1 }
δ' ({ q0, q1 }, b) = δ (q0, b) ∪ δ (q1, b) = { q0, q2 }

Now we will update the transition table of DFA.

δ' (Transition Function of DFA)

State	a	B
q0	{q0,q1}	q0
{q0,q1}	{q0,q1}	{q0,q2}

Now { q0, q2 } will be considered as a single state. As its entry is not in Q', add it to Q'.
So Q' = { q0, { q0, q1 }, { q0, q2 } }

Now, moves from state {q0, q2} on different input symbols are not present in transition table of DFA, we will calculate it like:
δ' ({ q0, q2 }, a) = δ (q0, a) ∪ δ (q2, a) = { q0, q1 }
δ' ({ q0, q2 }, b) = δ (q0, b) ∪ δ (q2, b) = { q0 }
Now we will update the transition table of DFA.

δ' (Transition Function of DFA)

State	a	B
q0	{q0,q1}	q0
{q0,q1}	{q0,q1}	{q0,q2}
{q0,q2}	{q0,q1}	q0

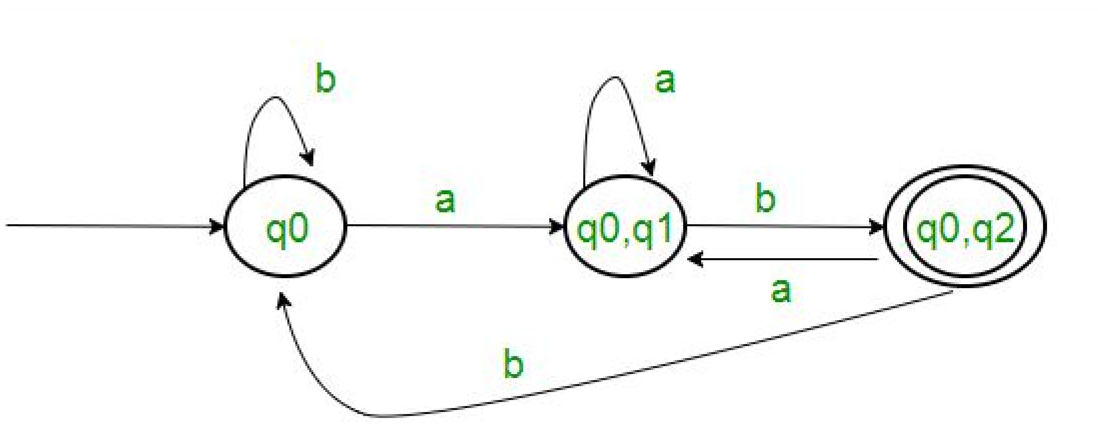
As there is no new state generated, we are done with the conversion. Final state of DFA will be state which has q2 as its component i.e., { q0, q2 }

Following are the various parameters for DFA.

Q' = { q0, { q0, q1 }, { q0, q2 } }

∑ = (a, b)

F = { { q0, q2 } } and transition function δ' as shown above. The final DFA for above NFA has been shown in Figure 2.



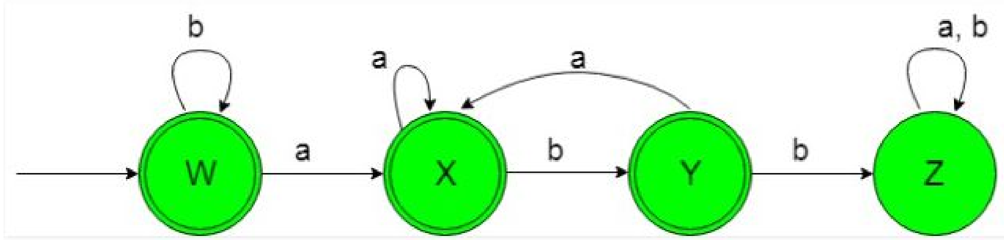
2 a **Construct a DFA accepting a set of strings over {a, b} in which every 'a' is never followed by 'bb'.** 4

Solution: The language contains the following strings:

L = { λ , a, aa, aaa…, ab, abaaabaa, b, bb, bbb…}

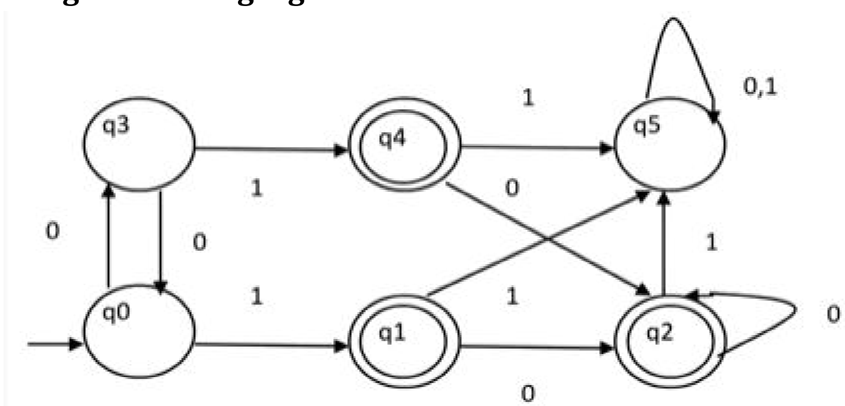
Here we can see that each string of the language containing 'a' is never followed by 'bb'.

The state transition diagram of the language L is given as:



Note :
Please note this language is not a compliment of the language = {where every a is followed by bb}
The reason is as both the languages will have the following strings in common : { λ , b, bb, any no. b's}
Since the intersection of these languages is non-empty, they cannot be a complement of each other. Hence we cannot take a compliment of the DFA that accepts the language ={where every a is followed by bb}. As the complement of the DFA does not result in a language ={where a is never followed by bb}.

b **Convert the given DFA to its equivalent DFA with a minimum number of states using table filling algorithm.** 6



Solution :
DFA Transition table:

	0	1
→ q0	q3	q1*
q1*	q2*	q5
q2*	q2*	q5
q3	q0	q4*
q4*	q2*	q5
q5	q5	q5

Using Table filling Algorithm

q1*	X				
q2*	X				
q3		X	X		
q4*	X			X	
q5	X	X	X	X	X
	q0	q1*	q2*	q3	q4*

Distinguishable pairs (Pair of Final and Non-Final State) are marked in green color.

We must mark the following states as distinguishable due to the following reasons:

δ ((q0,q5) , 1) = {q1,q5}

δ ((q3,q5) , 1) = {q4,q5}

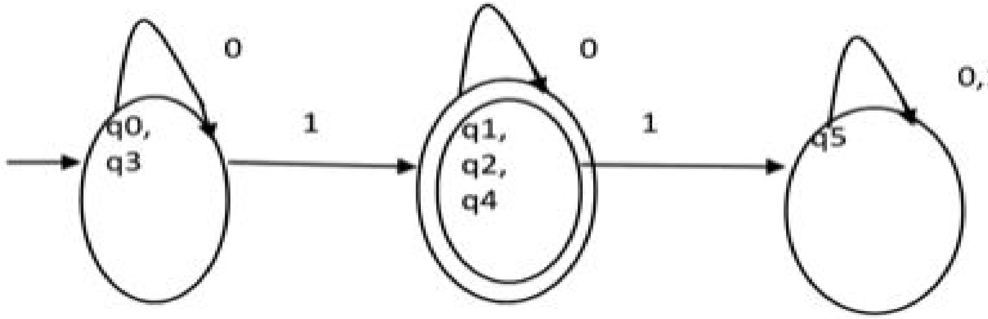
We must however make a check for all the pairs of states(which are not yet marked) and check whether they can be marked as distinguishable pairs or not.

The following pairs of states can be merged as:

	0	1
→ q0q3	q0q3	q1*q4*
q1*q2*q4*	q2*	q5

Hence, the Minimized automata will have the following transition table and transition diagram:

	0	1
→ q0q3	q0q3	q1*q2*q4*
q1*q2*q4*	q1*q2*q4*	q5
q5	q5	q5

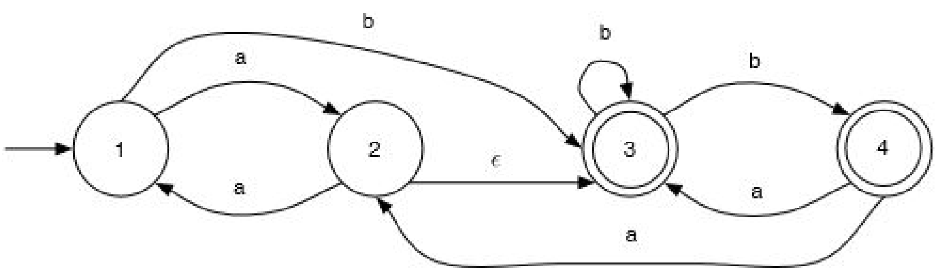
<table>
<tr><td></td><td></td><td>
</td><td></td></tr>
<tr><td>3.</td><td>a</td><td>Construct a regular expression for the following languages

i) L = {w | w Ɛ {a, b}* and w has exactly one pair of consecutive a’s}

Solution: (b + ab)* aa (b + ba)*

ii) L = { $a^n b^m$ | n, m are two integers such that (n+m) is even}, Σ = {a, b}

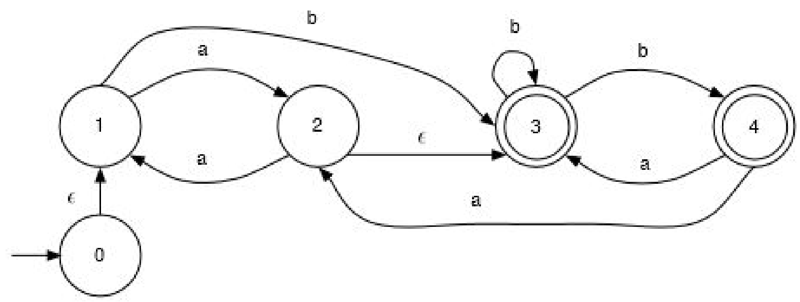
Solution: (aa)* (bb)* + a (aa)* b (bb)*</td><td>4</td></tr>
<tr><td></td><td>b</td><td>Obtain Regular Expression for the following automata using state elimination technique. The removal of the states should be in ascending order of their names (s.t. 1,2,3,4)



Solution:-</td><td>6</td></tr>
</table>

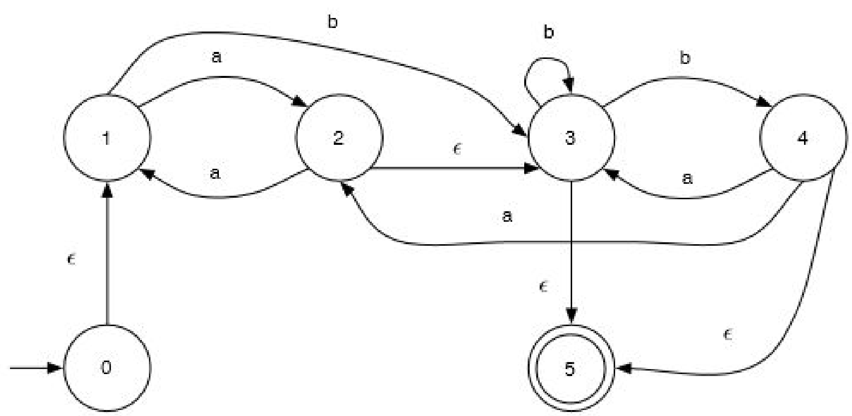
Step 1

If the start state is an accepting state or has transitions in, add a new non-accepting start state and add an ϵ-transition between the new start state and the former start state.



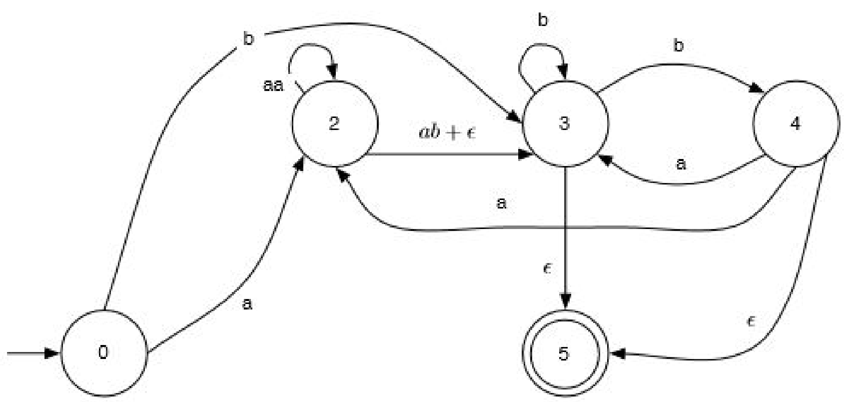
Step 2

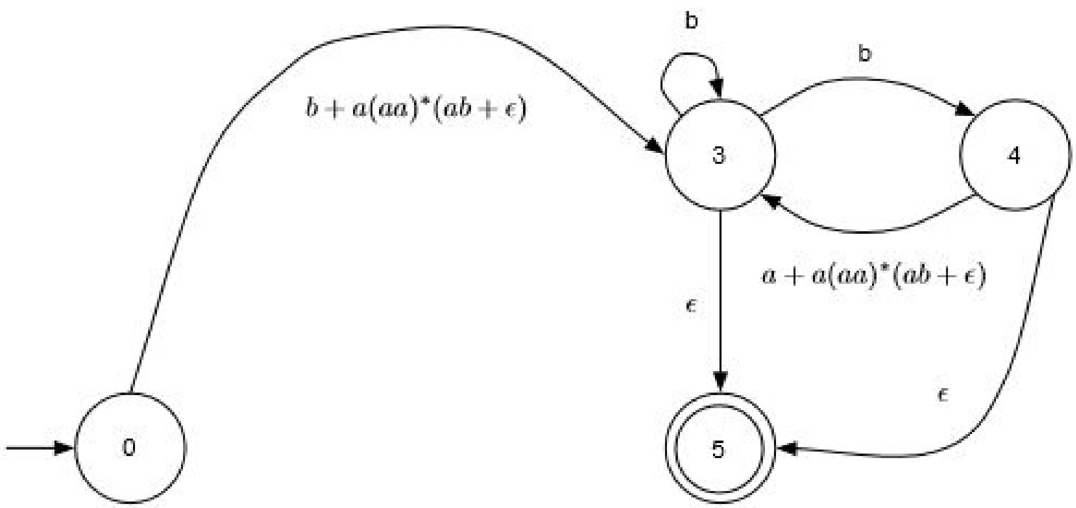
If there is more than one accepting state or if the single accepting state has transitions out, add a new accepting state, make all other states non-accepting, and add an ϵ-transition from each former accepting state to the new accepting state.

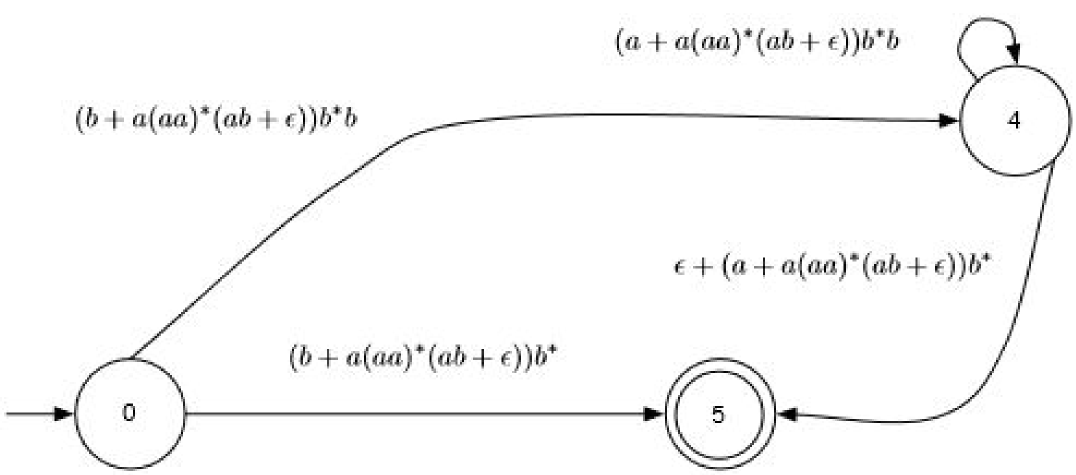


Step 3

For each non-start non-accepting state in turn, eliminate the state and update transitions The following illustrations depict the removal of states 1, 2, 3, and 4 in that order.







$$(b+a(aa)^*(ab+\epsilon))b^*+$$

$$((b+a(aa)^*(ab+\epsilon))b^*b)((a+a(aa)^*(ab+\epsilon))b^*b)^*(\epsilon+(a+a(aa)^*(ab+\epsilon))b^*)$$

4. a **State and prove the pumping lemma for the regular languages** 5

Solution:

Let L be an infinite regular language. Then there exists some positive integer m such that any $w \in L$ $|w| \geq m$ can be decomposed as

$$w = xyz$$

with

$$|xy| \leq m,$$

and

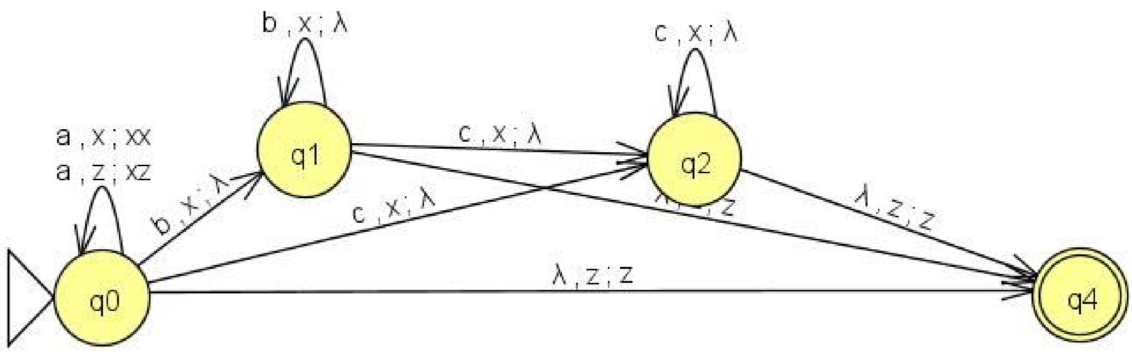
$$|y| \geq 1,$$

such that

$$w_i = xy^iz,$$

is also in L for all $i = 0, 1, 2, \ldots$.

		Proof: If L is regular, there exists a dfa that recognizes it. Let such a dfa have states labeled $q_0, q_1, q_2, \ldots, q_n$. Now take a string w in L such that $\|w\| \geq = n+1$. Since L is assumed to be infinite, this an always be done. Consider the set of states the automaton goes through as it processes w, say $$q_0, q_i, q_j, \ldots, q_f.$$ Since this sequence has exactly $\|w\| + 1$ entries, at least one state must be repeated, and such a repetition must start no later than the nth move. Thus, the sequence must look like $$q_0, q_i, q_j, \ldots, q_r, \ldots, q_r, \ldots, q_f,$$ indicating there must be substrings x, y, z of w such that $$\delta^*(q_0, x) = q_r, \quad \delta^*(q_r, y) = q_r, \quad \delta^*(q_r, z) = q_f,$$ with $\|xy\| \leq n+1 = m$ and $\|y\| \geq 1$. From this it immediately follows that $$\delta^*(q_0, xz) = q_f,$$ as well as $$\delta^*(q_0, xy^2z) = q_f, \quad \delta^*(q_0, xy^3z) = q_f,$$ and so on, completing the proof of the theorem. ■	
	b	**Prove that family of regular languages are closed under union and concatenation operations** **Solution:** Let us consider two regular languages L1 and L2 which can be represented by regular expressions r1 and r2 respectively. ● Using these we can construct a new regular expression r1+r2 . This new regular expression denotes the language L(r1+r2) = L(r1) U L(r2) = L1 U L2 and hence, family of regular languages are closed under union operation. ● We can construct another regular expression r1r2 which represent the language L(r1r2) = L(r1).L(r2)=L1.L2 and hence family of regular languages are closed under concatenation operation	5
5.	a	**Obtain a CFG for the language L={ $a^n b^m c^k$: n,m,k>=0 and m=n+k}** **Solution:** S-> AB A-> aAb | λ B->bBc | λ	4
	b	Construct a PDA for the language L={ $a^n b^m c^k$: n,m,k>=0 and n=m+k} **Solution:**	6

<table>
<tr>
<td></td>
<td></td>
<td>


Logic:
Since the #a's = #b's + #c's

For every a in the string we push an X on to the stack.
Then, we pop an X for every b.
Similarly, we pop an X for every c.
When the total # of b's and c's match the #a's, stack will be empty (it will only have Z which marks the bottom of the stack), we will accept the input string.

In the whole process we ensure that all a's must come first then b's and then c's. We must also take care of the powers (n,m,k>=0) and handle all cases appropriately.
</td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td>a</td>
<td>
Check whether the following grammar is ambiguous or not. If it is ambiguous then obtain an unambiguous grammar for the language accepted by this grammar
S-> AB | aaB
A→ aA | a
B→ b
Solution:
The grammar is ambiguous since for the string w=aab there exists 2 different leftmost derivations
<table>
<tr>
<td>LMD1:
$S \Rightarrow^{lm}$ aaB
$S \Rightarrow^{lm}$ aab</td>
<td>LMD2:
$S \Rightarrow^{lm}$ AB
$S \Rightarrow^{lm}$ aAB
$S \Rightarrow^{lm}$ aaB
$S \Rightarrow^{lm}$ aab</td>
</tr>
</table>
The unambiguous grammar will be:
S-> aAb
A-> aA | λ
</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>b</td>
<td>
Convert the following grammar into Chomsky Normal Form
S → a | aA | B.
A → aBB | λ
B → Aa | b
Solution:
Step1 : Eliminate λ-production.
The nullable variable(s) are = { A}
We get the following grammar :
</td>
<td>6</td>
</tr>
</table>

		S-> a | aA |B A-> aBB B-> Aa| a| b **Step 2 : Eliminate unit production i.e. S-> B** S-> a | aA |Aa|b A-> aBB B-> Aa| a| b **Step 3 : Eliminate useless production.** There are no useless production/symbols as all grammar satisfies derivability and reachability aspects **Step 4: Conversion to CNF** S-> a | XA | AX | b X-> a A-> XY Y-> BB B-> AX | a | b	

Acknowledgement : The sample paper solution is prepared by Dr. Pooja Agarwal and Dr. KarthiK S.