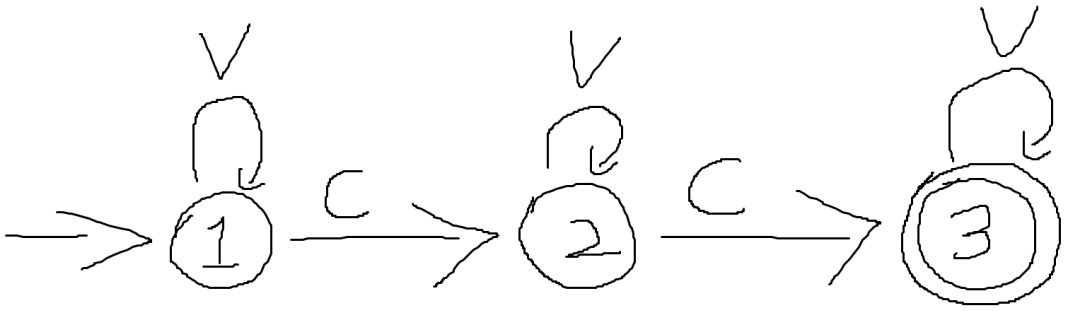
Question 1B:

In order to convert an SLG to an FSA, we need to first create a state for each of the symbols in the SLG, and then add an additional “starting” state. From the starting state, we then need to draw a transition to each of the states that correspond to the “starting” symbol given in the SLG, and label these transitions with the “starting” symbol. Then, for each bigram (a, b) in the SLG’s allowable bigrams T, create a transition from state a to state b and label this transition with symbol b. Finally, for each state that corresponds to a final symbol in the SLG, create a transition from that state to one of the given “final” states from SLG.

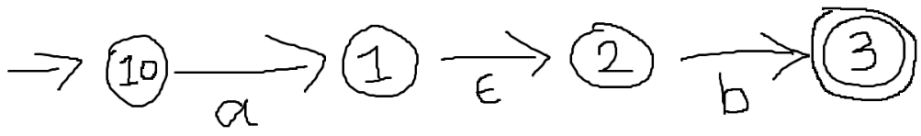
Question 1D:

One example of a language that a FSA can generate but a SLG cannot is “string with exactly two C’s and any number of V’s”. The reason why SLG cannot generate this string language is because it can only generate the string based on the adjacent pairs of symbols instead of counting how many times a certain symbol occurs in the string.
In an example sequence “CVC”, SLG has no way to know how many times it has seen C, because it only looks at the last two symbols to decide. This could result in multiple C’s added exceeding the limit of two, or rejecting strings that follow the rule of containing only two C’s but don't follow the adjacent symbols.



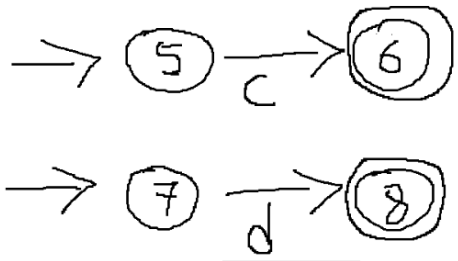
Question 2E:

1. $(a \cdot b)$



---

2. $(c \mid d)$



---

3. $(a \cdot b)^*$

---

4. $((((a \cdot b)^*) \cdot (c \mid d))$