Jay Patel
HW5
CS 331
Professor Oliver

Q1.

As we know that w is accepted by L (M). But in order to show the construction for G, we can tell that there is w that belongs to L (G) so w $\in L\ (G)$; using construction proof:
For G's construction: (V, Σ, R, S)
So, for V: $\{V_i\ |q_i\ \in Q\}$
R: $\{V_i \to aV_j | \delta(q_i, a) = q_j\}$ U $\{V_i \to\ \epsilon | q_i\ \epsilon\ F\}$
S= $V_0\ where\ q_0\ is\ the\ start\ state\ for\ DFA\ M$

So, we know that w is accepted by L (M), $s_n\ \in F$. There will be transition in the path for the computation, that will give us, $s_i \to\ s_j$, which exists in our grammar. Same for the states that are accepting path that is $V_n \to \in$
Therefore, in order to get to s_n you will need to follow the accept states and the rules. As we did for above, we do the same for non-accepting states.
Which then concludes these two,
L (M) = L (G) for any DFA (M) and L (M) will accept and reject the same string as L (G)

In other words, as the TA explained
I checked if w belongs to L (M) and then that goes and check for w belongs to L (G)
Which then checks for c0….ck belongs to F

For the second case: we check for w belongs to L (G) and then we go to w belongs to L (M) which then follows to S → W with S = V1 → Wk → w

Q2.
Grammar:
S = u|v
U = au|x
V = vc|y
X = bXc|∈
Y = aYb|∈

Context free grammar is ambiguous because strings ‘abc’ can draw with 2 different parse trees.

```
 S                               S
 ↓                               ↓
 U              OR               V
 ↓                               ↓
↙↘                                 ↘
a  u                             V     C
    ↓                            ↓
   ↙↘                            Y
   s x c                         ↓
    ↓                           ↙↘
     ∈                         a  y  b
                                  ↓
                                  ∈
```

Q3.

a. L = $\{a^n * \ b^m : n \# m\}$ U {(a U b)*ba (a U b)*}
Lets say L1 to be our Left most
And L2 to be our Right most

For L1:
S1 → as|b|T|u
T → aT|a
U → ub|b

For L2:
S2 → RbaR
R → RR|a|b|^2

So, L1 U L2

S → S1|S2
S1 → as|b|T|u
S2 → RbaR
T → aT|a
U → ub|b
R → RR|a|b|^2

The context free grammar of L is:
S → TR
T → 0T0| 1T1 | #R
R → RR|0|T|E

b. The context free grammar is in the following,

P → A|T#A#T|T#A|A#T
A → aAb|bAa|#|#T#
T → aT|bT|#T|∈

Where;
A = the matched strings
T = non-terminal that as terminal of a and b.

We can tell that we can have a string of any length a and b that has can be separated by # no matter what the position is so it can be either left, right or in the middle.