1. Show step-by-step how the string 00001001001 would be compressed by the SEQUITUR algorithm.

Q1

(Writing only keys, not position values, or pointer to next higher position values)

Letter	Aux	Production Table	Hash
0	0	{}	{}
0	00	{}	{00}
0	000	{}	{00, 00}
0	0000 becomes AA	{A→00}	{AA}
1	AA1	{A→00}	{AA, A1}
0	AA10	{A→00}	{AA, A1, 10}
0	AA100 becomes AA1A	{A→00 marked}	{AA, A1, 1A}
1	AA1A1 becomes ABB	{A→00 marked, B→A1}	{AB, BB}
0	ABB0	{A→00 marked, B→A1}	{AB, BB, B0}
0	ABB00 becomes ABBA	{A→00 marked, B→A1}	{AB, BB, BA}
1	ABBA1 becomes ABBB	{A→00 marked, B→A1 marked}	{AB, BB, BB}

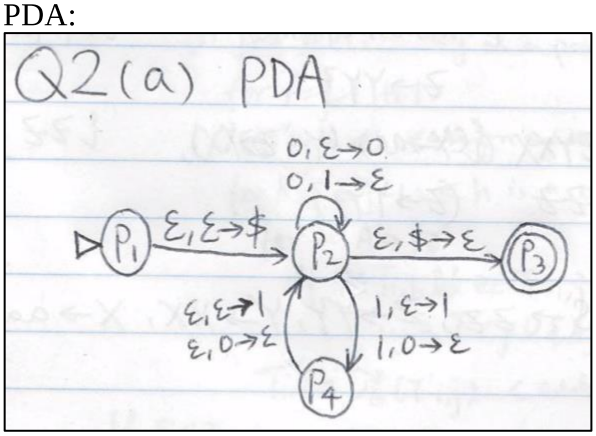
Initially, we get the grammar {A→00, B→A1, S→ABBB}

~~It is the final answer as well.~~

The compressed string is R00R#01R#0#1#1#1

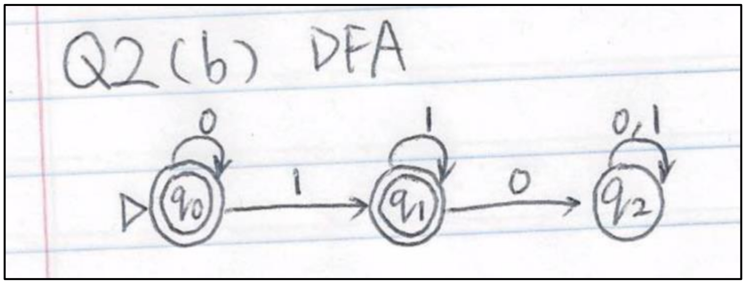
2. Draw a PDA for the language L over {0,1} consisting of strings with twice as many 0's as 1's (0.5pt). So 001010001 would be in this language. Next draw a DFA recognizing 0⋆1⋆ (0.5pt). Use the algorithm from class to draw a PDA for the intersection of these two languages (1pt).

PDA:



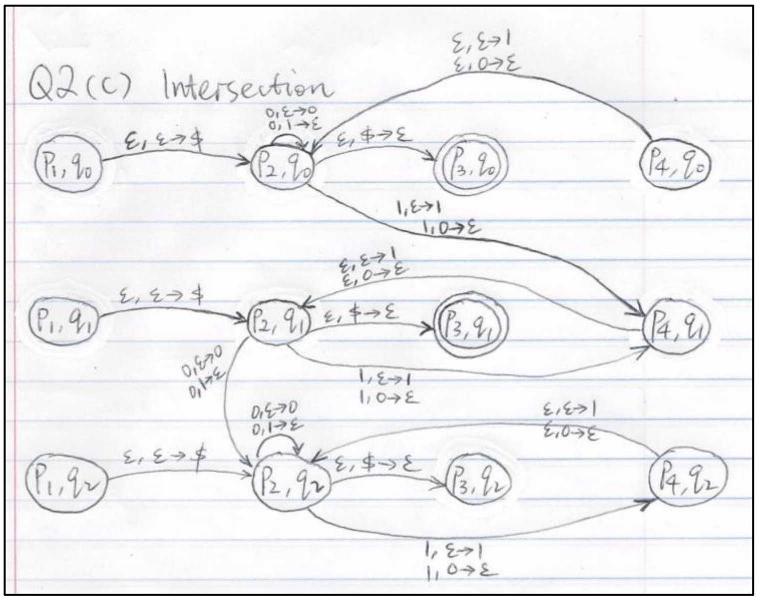
If the PDA reads a 0, it pushes 0 or pops 1. If it reads a 1, it pushes two 1's or pops two 0's or one of each. When the stack is empty, a string that has twice as many 0's as 1's will be accepted.

DFA:



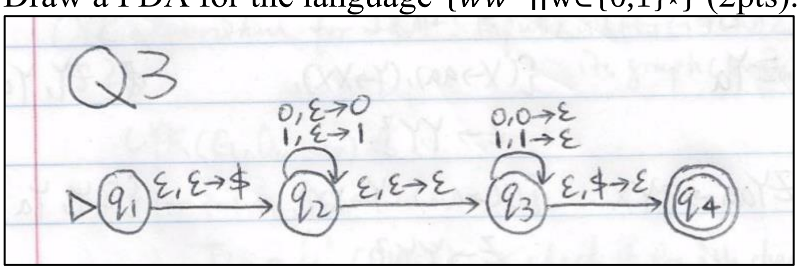
0*1*. q0 is a start state and a final state. q1 is a final state. q2 is a trap state.

Intersection:



Strategy: Cartesian product of the two graphs (PDA & DFA)

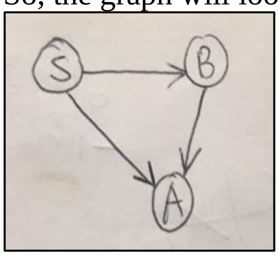
3. Draw a PDA for the language $\{ww^R||w\in\{0,1\}\star\}$ (2pts).



Initially we put a special symbol '$' into the empty stack. At state q2, the w is being read. In state q3, each 0 or 1 is popped when it matches the input. If any other input is given, the PDA will go to a dead state. When we reach that special symbol '$', we go to the accepting state q4.

4. Show step-by-step how the algorithm from class for checking if a language of CFG is infinite would operate on the grammar you got for problem (1) in this group.
Solution:

- The grammar I got for problem (1) is A->00, B->A1, S->ABBB
- First, eliminate ε-rules from G. There is no ε-rules, so move on to the next step.
- Then, eliminate unit-productions from G. There are no unit-productions, so move on to the next step.
- Then, eliminate useless symbols from G. There are no useless symbols, so move on to the next step.
- S is the start symbol. The left derivation of the grammar:
  S->ABBB ->00A1A1A1->00001001001
- We then construct a graph where (A,B) is an edge for two variables A, B in the graph iff A→xBy for some production in G.
- For A->00, it contains no edges because A only gives terminals. We cannot construct a graph.
- For B->A1, we construct a graph where **(B,A) is an edge** for two variables B, A in the graph iff B->xAy for some production in G, where x=null, y=1
- For S->ABBB, we construct a graph where **(S,A) is an edge** for two variables S, A in the graph iff S->xAy for some production in G, where x=null, y=BBB.
  We can also construct a graph where **(S,B) is an edge** for two variables S, B in the graph iff S->xBy for some production in G, where x=A, y=BB; or x=AB, y=B; or x=ABB, y=null.
- So, the graph will look like:



- There is no cycle in the graph, so it can be concluded that the given grammar is not infinite.

5. (a) Prove the language {w−w|w∈{0,1}⋆} is not CFL. Here - is a symbol in the TM's alphabet (1pt). (b) Give the formal definition as a 6-tuple of a TM recognizing the language {w−w|w∈{0,1}⋆} (0.5pt). Here - is a symbol in the TM's alphabet. (c) Show formally that your machine accepts the string 001−001 (0.5pt).

(a) Proof: L={w−w|w∈{0,1}*} is not CFL using pumping lemma.

Suppose C was CFL. Then it has a pumping length p. Consider the string $s = 0^p1^p - 0^p1^p$

According to the theorem taught in the class, if A is a context free language, then there is a number p (the pumping length) where, if s is any string in A of length at least p, then s may be divided into five pieces s = uvxyz satisfying the conditions:

1. For each i ≥ 0, uv^ixy^iz is in C,
2. |vy| > 0, and
3. |vxy| ≤ p.

There are 5 cases to consider:

Case 1: Neither v or y contains -. Otherwise, uv^0xy^0z does not contain -. Thus, the string s can't be a part of L.

Case 2: If both v and y are nonempty and happen on the right side of -, the string s = uv^0xy^0z is not on the right side of -. Thus, the string s can't be a member of L.

Case 3: If both v and y are nonempty and happen on the left side of -, the string s = uv^0xy^0z is not on the left side of -. Thus, the string s can't be a member of L.

Case 4: If only one of either v or y is non-empty, we can look upon them as they both occurred at the same side of -. Thus, the string s can't be a member of L.

Case 5: If both v and y are nonempty and include the symbol -, then the third pumping lemma condition $|vxy| \leq p$. We have v that consists of 1's and y that consists of 0's. Then uv^2xy^2z contains more 1's on the right side than the left side.

Since the string can't be pumped using the pumping lemma conditions, the language L is not CFL. Proven.

(b) A Turing Machine (TM) is a 6-tuple M = (Q, ∑, Γ, δ, q_0, H), where:

$Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7\}$

$\Sigma = \{0, 1, -\}$

$\Gamma = \{0, 1, -, X, Y, B\}$

$q_0 = \{q_0\}$

$H = \{q_7\}$

$\delta = \{$ $\delta(q_0, 0) = (q_1, X, R)$

$\delta(q_0, 1) = (q_4, X, R)$

$\delta(q_1, 0) = (q_1, 0, R)$

$\delta(q_1, 1) = (q_1, 1, R)$

$\delta(q_1, -) = (q_2, -, R)$

$\delta(q_2, 0) = (q_3, Y, L)$

$\delta(q_2, Y) = (q_2, Y, R)$

$\delta(q_3, 0) = (q_3, 0, L)$

$\delta(q_3, 1) = (q_3, 1, L)$

$\delta(q_3, Y) = (q_3, Y, L)$

$\delta(q_3, -) = (q_3, -, L)$

$\delta(q_3, X) = (q_0, X, R)$

$\delta(q_0, -) = (q_6, -, R)$

$\delta(q_6, Y) = (q_6, Y, R)$

$\delta(q_6, B) = (q_7, B, R)$

$\delta(q_7, B) = (q_7, B, R)$

$\delta(q_4, 0) = (q_4, 0, R)$

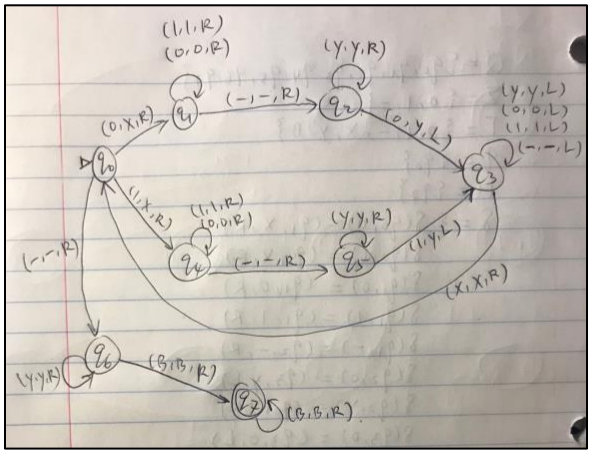
$\delta(q_4, 1) = (q_4, 1, R)$

$\delta(q_4, -) = (q_5, -, R)$

$\delta(q_5, \cancel{b}Y) = (q_5, -, R)$

$\delta(q_5, 1) = (q_3, Y, L)$ $\}$.

Turning Machine is:



(c) Consider the string 001-001

1. ↓
   001-001⊔⊔⊔..
   Head starts at leftmost variable, see pointer

2. X01-001⊔⊔⊔..
   Record symbol and overwrite it with X
   Now continue to scan to the right, reject immediately if we encounter blank ⊔ before our -

3. ↓
   X01-X01⊔⊔⊔..
   When we encounter our middle symbol -, move right one more time and see if the current variable matches our recorded one (if it doesn't match, reject)
   Overwrite current symbol with X

4. ↓
   XX1-001⊔⊔⊔..
   We are now on our second iteration of the algo, do the same thing, now moving one cell to the right and recording it with an X.

5. ↓
   XX1-XX1⊔⊔⊔..

6. ↓
   XXX-XX1⊔⊔⊔..

7. ↓
   XXX-XXX⊔⊔⊔..

8. ↓
   XXX-XXX⊔⊔⊔..

Continue to move, if 0 or 1 is encountered, reject. As we can see, a blank ⊔ is encountered, so we can ACCEPT. Thus our string is accepted by our machine and we have shown it formally using our algorithm.