Assignment#2 Key
1a. \( \text{NoLeadingOrLaggingAs}(L) = \{ x \mid \text{a}^*\text{x}\text{a}^* \text{ is in } L \text{ and } x \text{ has no leading or lagging a’s } \} \)

- Approach 1: Let \( L \) be a Regular language over the finite alphabet \( \Sigma \). For each \( a \in \Sigma \), define \( f(a) = \{a,a’\} \), \( g(a) = a’ \) and \( h(a) = a, h(a’) = \lambda \), \( f \) is a substitution, \( g \) and \( h \) are homomorphisms.

\( \text{NoLeadingOrLaggingAs}(L) = h(f(L) \cap (g(a^*) (\Sigma-\{a\}) \Sigma^* (\Sigma-\{a\}) g(a^*) ) ) \)

- Why this works:
  \( f(L) \) gets us every possible random priming of letters of strings in \( L \).
  \( (g(a^*) (\Sigma-\{a\}) \Sigma^* (\Sigma-\{a\}) g(a^*) ) \) gets every word composed of a sequence of zero of more \( a \)’s followed by a non-\( a \) character from \( \Sigma \) then a sequence of characters from \( \Sigma \) that ends in a non-\( a \) character from \( \Sigma \) followed by a sequence of zero of more \( a \)’s. Intersecting this with \( f(L) \) gets strings of the desired form that occur in \( L \). Applying the homomorphism \( h \) erases all primed letters resulting in every string the language \( \text{NoLeadingOrLaggingAs}(L) \) that we sought. This works as Regular Languages are closed under intersection, concatenation, \( * \), substitution and homomorphism.
1b. \[ \text{MidThird}(L) = \{ y \mid \text{there exists a } x \text{ and } z, \ |x| = |y| = |z| \text{ and } xyz \text{ is in } L \} \]

- Let \( L \) be a Regular language over the finite alphabet \( \Sigma \). Assume \( L \) is recognized by the DFA \( A_1 = (Q, \Sigma, \delta_1, q_1, F) \). Define the NFA \( A_2 = ((Q \times Q \times Q \times Q \times Q) \cup \{ q_0 \}, \Sigma, \delta_2, q_0, F') \), where
  \[
  \delta_2(q_0, \lambda) = \text{union}(q, r \in Q) \{ <q_1, q, q, r, r> \} \]
  and
  \[
  \delta_2(<s, t, u, v, w>, b) = \text{union}(a, c \in \Sigma) \{ <\delta_1(s, a), \delta_1(t, b), u, \delta_1(v, c), w> \}, s, t, u, v, w \in Q
  
  \]
  \[
  F' = \text{union}(q \in Q) \{ <q, q, r, f, r> \}, f \in F
  
  \]

- Why this works:
  The first part of a state \(<s, t, u, v, w>\) tracks \( A_1 \) for all possible strings that are the same length as what \( A_2 \) is reading in parallel. We guess it will end up in state \( q \) and so \( u=q \) to remember that guess.
  The second part of state \(<s, t, u, v, w>\) tracks \( A_1 \) as if it has read a string that ended in state \( q \) (\( u=q \)). This part actually reads the mid part of a string divided into thirds.
  The third part of a state \(<s, t, u, v, w>\) tracks \( A_1 \) for all possible strings that are the same length as what \( A_2 \) is reading in parallel. We guess that reading the mid part will end up in state \( r \) (\( w=r \)).

- Thus, we start with a guess (\( q \)) as to what state \( A_1 \) might end up in reading a string of length \( x \).
  The guess is checked by requiring us to start up in state \( q \) in the mid part which reads \( y \), were \( |x|=|y| \).
  We guess that we will end up in state \( r \) after reading \( y \).
  The guess is checked by requiring us to start up in state \( r \) in the third part which simulates reading a string \( z \), where \( |x|=|y|=|z| \).

- The final states check that our guesses were correct, and the third part could end in a final state of \( A_1 \).
2. Use Regular Equations to Solve for $D + E$

A = $\lambda$
B = A₁ + C₁ + E(0+1) + B₀ = 1 + B₀*1 + D(0+1) + B₀ = 1 + C₁(0+1) + B(0*1 + 0)

= 1 + B₀*1(0+1) + B(0*1 + 0) = 1(0*1(0+1) + 0*1 + 0)*
C = B + C₀ = B₀* = 1(0*1(0+1) + 0*1 + 0)* 0*
D = C₁ = 1(0*1(0+1) + 0*1 + 0)* 0*1 = 1(0*1(0+1+\lambda) + 0)* 0*1
E = D
3. \( L = \{ a^m b^{2^n} \mid m, n > 0 \} \)

a.) Use the **Myhill-Nerode Theorem** to show \( L \) is not Regular.

Define the equivalence classes \([ab^{2^i}], i \geq 0\)

Clearly \( ab^{2^i}b^{2^i} \) is in \( L \), but \( ab^{2^j}b^{2^i} \) is not in \( L \) when \( j > i \)

Thus, \([ab^{2^i}] \neq [ab^{2^j}]\) when \( j \neq i \) and so the index of \( R_L \) is infinite.

By Myhill-Nerode, \( L \) is not Regular.
3. \( L = \{ a^m b^{2^n} \mid m, n > 0 \} \)

b.) Use the **Pumping Lemma for CFLs** to show \( L \) **is not** a CFL  

Me: \( L \) is a CFL  

PL: Provides \( N > 0 \)  

Me: \( z = a b^{2^N} \)  

PL: \( z = uvwxy, \ |vwx| \leq N, \ |vx| > 0, \) and \( \forall i \geq 0 \ uv^iwx^iy \in L \)  

Me: If \( vwx \) includes the one \( a \) then set \( i = 0 \) and we get a string with no \( a \)'s which is not in \( L \). Thus, we can assume \( vwx \) is over \( b \)'s only. Since \( 0 < |vwx| \leq N \), setting \( i = 2 \) adds at least one \( b \) and at most \( N \) \( b \)'s. However, the smallest member of \( L \) longer than \( a b^{2^N} \) has \( 2^N \) more \( a \)'s and so \( uv^2wx^2y \) cannot be in \( L \). Thus, the Pumping Lemma shows \( L \) to not be a CFL.
3. \( L = \{ a^m b^{2^n} \mid m, n > 0 \} \)

c.) Present a CSG for \( L \) to show it is context sensitive
\[
G = ( \{ S, A, B, C, D, E, X, <a>, <b> \}, \{ a, b \}, R, S )
\]
\[
S \rightarrow <a>bb \quad \text{// Base case of } a^*b^2
\]
\[
S \rightarrow Bb<b> \quad \text{// One } b \text{ and a pseudo } b \text{ } <b> \text{ that is a sentinel; gets all cases about } b^2
\]
\[
Bb \rightarrow XbC \quad \text{// } C \text{ will shuttle left doubling } b \text{'s; first } b \text{ doubled later}
\]
\[
Cb \rightarrow bbC \quad \text{// Double the number of } b \text{'s}
\]
\[
C<b> \rightarrow Db<b> \quad \text{// Double but keep that sentinel at end}
\]
\[
C<b> \rightarrow Ebb \quad \text{// Done doubling so get rid of sentinel}
\]
\[
bD \rightarrow Db \quad \text{// Shuttle } D \text{ back to start}
\]
\[
XD \rightarrow Bb \quad \text{// Change } XD \text{ back to } Bb \text{ to continue increasing } b \text{'s}
\]
\[
bE \rightarrow Eb \quad \text{// Shuttle } E \text{ back to start}
\]
\[
XE \rightarrow <a>b \quad \text{// Switch to generating } a \text{'s}
\]
\[
<a> \rightarrow a<a> | a \quad \text{// Start with } a^+"