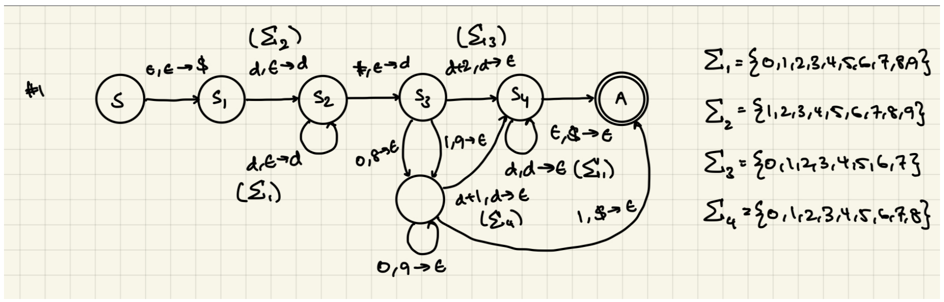
Ishaan Mantripragada.

Problem 1

To show that the language can be recognized by a pushdown automata, consider the figure below where $\Sigma_1 = \{0,1,2,3,4,5,6,7,8,9\}$, $\Sigma_2 = \{1,2,3,4,5,6,7,8,9\}$, $\Sigma_3 = \{0,1,2,3,4,5,6,7\}$, and $\Sigma_4 = \{0,1,2,3,4,5,6,7,8\}$:



Beginning in the start state, S, we have a transition $\epsilon, \epsilon \to \$$ from S to S_1 to initialize the stack. We have another transition $d, \epsilon \to d$ from state S_1 to S_2. For this transition, $d \in \Sigma_2$ to remove any cases of leading 0's. We then have a transition $d, \epsilon \to d$ from S_2 to itself to read the rest of the string. For this transition, $d \in \Sigma_1$. Then, a transition $\#, \epsilon \to d$ from S_2 to S_3 to read the # symbol. S_3 then has three different transitions. First, $d+2, d \to \epsilon$ from S_3 to S_4 where $d \in \Sigma_3$. This represents the general case without any need to carry over remainders. Second, $0, 8 \to \epsilon$ and $1, 9 \to \epsilon$ from S_3 to another state which accounts for carry over from the $i+2$ term. On that new state, a self transition $0, 9 \to \epsilon$ to account for the case that the number has multiple 9's ($999 \to 1001$). From that new state, a transition $1, \$ \to \epsilon$ to the accept state is needed since we have reached an acceptable string. Also from that new state, a transition $d+1, d \to \epsilon$ where $d \in \Sigma_4$ to S_4 is needed for the carry over as well. Going back to S_4, we have a self loop $d, d \to \epsilon$ accounts for the rest of the reversed number. Finally, a transition from $\epsilon, \$ \to \epsilon$ to the accept state will confirm that the string can be recognized. Thus, we have shown a pushdown automata that accepts the language.

Problem 2

The language defined by $S \to aSb \mid bSa \mid SS \mid \epsilon$ is the set of all strings from the alphabet $\Sigma = \{a, b\}$ containing the same number of a's and b's, including the empty string.

To prove that it generates that language, we will use the hint to show that the shortest non-empty prefix of any string x cannot begin and end with the same symbol. We do this by induction.

Base Case: The base case in this case is the empty string which is defined in the language. For the non-empty string, the shortest strings produced by the grammar are ab and

ba. Neither string starts and ands with the same character.

Induction Hypothesis: Any string x generated by the grammar has the shortest non-empty prefix that does not begin and end with the same symbol.

Induction Step: We must show that the Induction Hypothesis holds for any string x' which is generated using the grammar from x. If x' was generated using the first two rules ($S \to aSb \mid bSa$), then x' starts and ends with different symbols. Any prefix of x' will either be a prefix of the inner S (which by induction hypothesis does not begin and end with the same symbol) or will include either the starting a and ending b or starting b and ending a, thus not beginning and ending with the same symbol. If x' was generated using the third rule ($S \to SS$), then x' is the concatenation of two strings from the language. The shortest prefix of x' will be the shortest prefix of the first S that was used to create x'. By the induction hypothesis, this S has the shortest non-empty prefix that does not begin and end with the same symbol.

Therefore, the grammar generates strings where the shortest non-empty prefix does not begin and end with the same symbol. This is precisely the language described by the grammar which is the set of all strings with equal numbers of a's and b's.

Problem 3

Let $A = \{0^a1^a2^b \mid a, b \geq 0\}$ and $B = \{0^a1^b2^b \mid a, b \geq 0\}$. Then we can define $C = A \cap B = \{0^a1^a2^a \mid a \geq 0\}$.

a. To show that A and B are context free, we can show that they can be defined using context free grammars. A can be generated by the following rules:

- $S \to 0S1 \mid X$
- $X \to 2X \mid \epsilon$

B can be generated by the following rules:

- $S \to 0S \mid Y$
- $Y \to 1Y2 \mid \epsilon$

b. To show that C is not a context free language, we can use the pumping lemma as shown in class. The pumping lemma states that for any context-free language, there exists a pumping length p such that any string s in the language with length at least p can be divided into five parts, $s = uvxyz$, such that for every $i \geq 0$, uv^ixy^iz is in the language, $|vy| > 0$ and $|vxy| \leq p$. We can assume C is context free and can be split into the string $uvxyz$. However, if we pump on v and x, it changes the number of 0's and 1's in the original string. This new string is not part of C. Therefore, since we have reached a contradiction, it follows that C is not a context free language.

Problem 4

a. Let us consider the language $L = \{a^i b^j c^k d^l \ : \ i = 0 \,\text{or}\, j = k = l\}$ To prove that L satisfies the conditions of the CFL pumping lemma, we have two cases to consider. First, if $i = 0$. In this case, j, k, and l can be any value. Let us define a string s in the language that can be divided into $uvxyz$ where the substrings v and y only contain one character. Then, pumping on v or y would result in a string that is still in L. The other case to consider is if $i \neq 0$ and $j = k = l$. Let us define a string $s = aaaaaaaaabbccdd$ in L. This string can be divided into $uvxyz$ such that both v and y only contain a's. In this case, if we pumped on v and y, then the resulting string is still in L. Thus, the CFL pumping lemma is incapable of proving that L is not a CFL.

b. Assume L is a context-free language. Therefore, there exists a CFG G that generates L. We can convert G to Chomsky Normal Form (CNF) without loss of generality. In CNF, each production is either of the form $A \to BC$ or $A \to a$, where A, B, C are nonterminal symbols, and a is a terminal symbol. The pumping length p can be determined based on the number of nonterminal symbols in G. Specifically, we can set p to be a number larger than the total possible number of parse trees with up to p leaves. Let w be a string in L with length at least p. According to the lemma, we mark p or more positions in w. We can construct a parse tree for w using the CFG G. Since w has at least p marked positions, and the tree has at most p leaves, there must be some path from the root to a leaf where a nonterminal is repeated (due to the Pigeonhole Principle). We then choose a path from the root to a marked position by always traveling to the child node with a greater number of marked descendants. This ensures that we are always moving towards a section of the tree that is relevant to the marked positions. On this path, find a repeated nonterminal symbol A. Let the upper occurrence of A derive a string uvy and the lower occurrence derive vy. Here, v and y are strings corresponding to the portion of w derived from the repeated nonterminal A. Define "branch nodes" as internal nodes whose left and right children both have marked descendants. This is important because it ensures that the string derived from these nodes interacts with the marked positions. We can now pump the string by replicating the subtree rooted at the lower occurrence of A any number of times. This results in the strings uv^ixy^iz, maintaining the structure of the tree and thus staying within the language L. To make sure it holds, we can review the conditions of Ogden's Lemma:

- For all $i \geq 0$, $uv^ixy^iz \in L$ due to the repetition of the subtree.
- vy contains at least 1 marked position (by the choice of the path)
- vxy contains at most p marked positions (since we are only pumping a part of the string that corresponds to a subtree with at most p leaves).

Thus, we have shown that Ogden's Lemma holds for any CFL L.

c. From part (a), $L = \{a^i b^j c^k d^l \ : \ i = 0 \,\text{or}\, j = k = l\}$. We can show that L is not a CFL by Ogden's Lemma. We will do this by contradiction. Let us assume that L is a CFL. Let $w = a^i b^j c^k d^l$ where $i = 1$ and $j = k = l = n$. Thus, the $w = ab^nc^nd^n$. We can mark all b's in w, and we can divide $w = uvxyz$ where vy contains at least 1 marked symbol. It is easy to see that if $v \neq y$, then the resulting string, $w = uv^ixy^iz \notin L$. Therefore, v and y must only contain the same character. Since we only mark b's and either v or y contains a marked character, then from the original split, $uvxyz$, either v is contained in b's or y is contained in the b's. Now, if we pump on both v and y, the number of b's increases. For the resulting string, uv^ixy^iz to be in L, then the number of c's and d's must also increase. However, only one of them can increase since we only pump on v and y. Thus, since we have reached a contradiction, our original assumption that L is a CFL is false.

Problem 5

To show that CFL's are not closed under complement, we can give an example of a CFL whose complement is not a CFL. Using the example from problem 3, the languages $A = \{0^a1^a2^b \| : a, b \geq 0\}$ and $B = \{0^a1^b2^b \mid a, b \geq 0\}$ are context free languages, but $C = A \cap B = \{0^a1^a2^a \mid a \geq 0\}$ is not.

We know that the intersection, $A \cap B = \overline{\overline{A} \cup \overline{B}}$. If CFL's were closed under complement, then $\overline{A}$ and $\overline{B}$ would both be CFL's and their union would also be a CFL (since CFL's are closed under union). However, since their union $\overline{\overline{A} \cup \overline{B}}$ is equivalent to C which we know is not a context free language, we have shown that CFL's are not closed under complement.