Ishaan Mantripragada.

Problem 1

Let's define a related language, $L_f = w\#f(w) : w \in \Sigma$, where w is in the original language and # is any random separator not in Σ. The language L_f consists of strings where each string is a concatenation of some word w from Σ^*, the separator #, and the function f applied to w. Note that Γ must include all symbols from Σ and the separator #. To compute $f(x)$ for some input x, the computer program can proceed as follows:

1. Generate strings of the form $x\#y$, where y varies over possible outputs in Σ^*.
2. For each generated string, use the procedure that decides L_f to check if it belongs to L_f.
3. The correct output $f(x)$ is identified when $x\#f(x)$ is found to belong to L_f.

To decide if a string s is in L_f, (1) Split s into two parts at the separator #, resulting in w and y. (2) Use the procedure to compute $f(w)$. (3) Check if $f(w)$ equals y. If yes, s belongs to L_f; otherwise, it does not.

Problem 2

a. To prove that L is a regular language if and only if L is recognized by an all-paths-NFA, we must prove this in both directions (1,2).

1. If L is regular, then by definition, there is an FA that can recognize it. Since an all-paths-NFA is also a FA, it can recognize a regular language where for any accepted input, it exactly ends on the accept state.
2. We can define an FA that can recognize the same language as an all-paths-NFA. Let the all-paths-NFA be defined as $M = (Q, \Sigma, \delta, q_0, F)$. We can construct an FA defined as $M' = (Q', \Sigma', \delta', q_0', F')$. The alphabet remains the same ($\Sigma = \Sigma'$). As defined in lecture, $E(S) = \{q \in Q : q \text{ reachable from S by traveling along 0 or more } \epsilon \text{ transitions}\}$. Therefore, $q_0' = E(\{q_0\})$ and $\delta'(R, a) = \cup_{r\in R}(E(\delta(r, a)))$ for each state $R \in Q'$. Finally, $F' = \{R \in Q' : R \text{ is an accept state of M}\}$. The key difference from lecture is that every state R must be an accept state due to the conditions of the all-paths-NFA.

b. From part (a), there exists an all-paths-NFA that accepts A and B. We can define M_a as the all-paths-NFA that accepts A, and M_b as the all-paths-NFA that accepts B. We can also define an all-paths-NFA, M_c, that accepts the intersection C. To make sure that any string from C is accepted by M_c, we must ensure that any permutation is accepted. To do this, we have the start state of M_c point to the start states of M_a and M_b via ϵ transitions. By this construction of M_c, we have defined an all-paths-NFA that accepts the intersection, C. Therefore, we can conclude that C is a regular language by part (a).

c. In the original NFA $M = (Q, \Sigma, \delta, q_0, F)$, the set of accepting states is F. The language L recognized by M consists of all strings over the alphabet Σ that lead the automaton from the start state q_0 to any of the accepting states in F following the transition function δ. In the flipped NFA $M_{flip} = (Q, \Sigma, \delta, q_0, F')$, the set of accepting states is $F' = Q - F$, meaning that the accept states in M_{flip} are the states that are not accept states in M. Thus, the language L_{flip} that is recognized by M_{flip} consists of all strings that would lead to a reject state in M. Thus, L_{flip} is the complement of L.

Problem 3

We do this proof by contradiction using the pumping lemma. Let L be a regular language consisting of all palindromes. By the pumping lemma, there exists an integer, p (pumping length), for which every $w \in L$ with $|w| \geq p$ can be written as $w = xyz$ such that

1. for every $i \geq 0$, $xy^iz \in L$
2. $|y| > 0$
3. $|xy| \leq p$

Let $w = aa...aba...aa$ where there are p a's on each side of the central b. Since $|w| \geq p$, we can rewrite w as xyz such that the pumping lemma holds. However, when $i > 0$ and we pump on y, the resulting string is no longer a palindrome. Thus, we have reached a contradiction meaning that L is not a regular language.

Problem 4

(a) L_n is a regular language for all $n \geq 1$ if there exists an FA that accepts the string. We can define an FA with a start state and $n-1$ accept states where the transition from one state to the next is the unary symbol, 0. However, the n^{th} transition from the final accept state would point to the start state which is not accepted. This FA accepts all strings whose length is not divisible by n. Thus, since we have defined an FA that accepts L_n, we have shown that L_n is regular.

(b) We do this by contradiction using the pumping lemma. Let PRIMES be a regular language consisting of all strings whose length is a prime number. We can define a string $w = 0...0$ where w has n total 0's. By the pumping lemma, we can define $w = xyz$ where all xy^iz for $i \geq 0$ and $|y| > 0$ are also in PRIMES. The total number of 0's in this string is $n + (i-1) \cdot (y\_count)$. Therefore, $n + (i-1) \cdot (y\_count)$ must also be prime for all $i \geq 0$. However, if we choose an n such that $n = i - 1$, the total number of primes becomes $(i-1) + (i-1) \cdot (y\_count) = (i-1) \cdot (1 + y\_count)$. For all $i > 1$, this number is divisible by both $(i-1)$ and $(1 + y\_count)$. This means that w is no longer in PRIMES and thus, PRIMES is not a regular language.