1. (16 pts.) Let $L$, $L_1$, and $L_2$ be languages over $\Sigma = \{0, 1\}$, and suppose that $L$ can be written as $L = L_1 \cap L_2$ (i.e., $L$ is equal to the intersection of $L_1$ and $L_2$). Prove or disprove: If $L$ is regular, then it must be that at least one of $L_1$ and $L_2$ is regular.

**Answer:** False. Take $L_1$ to be any nonregular language, and let $L_2$ be its complement, which also must nonregular since the regular class is closed under complement. However, $\emptyset = L_1 \cap L_2$ is regular.

2. (17 pts.) Demonstrate that the language

$$L = \{0^n 1^m 0^m \mid n, m \geq 0\}$$

over $\Sigma = \{0, 1\}$ is not regular.

**Answer**

$L$ cannot be pumped. Suppose to the contrary, and let $p$ be as required by the pumping lemma for regular languages. Consider $s = 1^p 0^p$ (i.e., $n = 0$, and $s$ is in $L$) and observe that $y$ can contain only 1's. Hence pumping $y$ either up or down results in an unequal number of 1's and terminating 0's, and hence in a string that is not in $L$.

3. (17 pts.) Again consider the language

$$L = \{0^n 1^m 0^m \mid n, m \geq 0\}$$

over $\Sigma = \{0, 1\}$. Let $s$ be any string in $L$ of length at least 2. Show how $s$ can be divided into $uvwxy$ so as to satisfy the pumping lemma (with $p = 2$) for CFL's.

**Answer**

If $n > 0$, simply let $v$ be the first 0 in $s$, with $uxy$ empty and $z$ the rest of the string. Otherwise, $m > 0$, so let $v$ be the last 1, $x$ empty, and $y$ the next 0.

4. (22 pts) Let $M_1$ be an arbitrary NFA with input alphabet $\Sigma$. Construct a second NFA $M_2$ such that both:

- $L(M_2) = L(M_1)$, and
- $M_2^c$ has as its language $\Sigma^*$,

where $M_2^c$ is the "complement machine" of $M_2$ formed by reversing the accept and reject states of $M_2$.

**Answer**

To form $M_2$, simply add an e-transition from $q_0$ into new state $q_{\text{new}}$, where $q_{\text{new}}$ is not an accepting state of $M_2$. $q_{\text{new}}$ also has transitions defined to loop back to itself on each input symbol. Clearly, $M_2$ satisfies (a) (the modification does not change the set of strings accepted, since choosing to take the new e-transition out of $q_0$ leads to a rejecting path). (b) is satisfied since $q_{\text{new}}$ is an accepting state of the complement machine $M_2^c$, and hence taking the new e-transition out of $q_0$ allows $M_2^c$ to accept all strings.
5. (28 pts) Describe a PDA with input alphabet $\Sigma = \{a, b\}$ whose language is

$$L = \{ w \mid w \text{ is } (a, b)^* \text{ and } w \text{ contains exactly three more } a\text{'s than } b\text{'s } \}.$$ 

Rather than giving state transitions, you may describe the PDA in terms of pushing and popping the stack, checking the top symbol on the stack, checking for stack empty, and checking for end of input. Argue that the language of your PDA is as required.

ANSWER
The PDA is identical to the PDA in Exercise Set 3, Question 2, except that after our new machine has read all its input it performs a sequence of e-transitions (null input) to check that exactly 3 $a$’s are on the stack at this point. In particular, there are new transitions $\delta(q, e, a) = (q_{pop1}, e)$ from each $q$ in the original machine’s $Q$ (or just from the original machine’s final accept states). Similarly, we have null transitions from new $q_{pop1}$ to new $q_{pop2}$ provided there is an ’$a’ on the stacktop, and same for $q_{pop2}$ to new $q_{pop3}$. If we arrive in $q_{pop3}$ and $\$ is on top of stack, we go into $q_{new\_accept}$, i.e., $\delta(q_{pop3}, e, \$) = (q_{new\_accept}, e)$. This $q_{new\_accept}$ is the only accepting state of the new machine, and it has no transitions out. Note that since there are no transitions out of the $q_{popj}$ that can read an input, if the new machine goes into these states prior to end of input, the path is not an accepting path.

The invariant from the original machine still holds up to the point of taking a new null transition into $q_{pop1}$. Consequently, at the time this null transition is taken, the number of $a$’s on the stack represents the surplus of $a$’s in the input. The new null transitions then verify that this number is 3.