1. Let $L$ be the language of all string of balanced parentheses, that is, all strings of the characters "(" and ")" such that each "(" has a matching ")". Use the Pumping Lemma to show that $L$ is not regular.

Assume that $L$ is regular. Then by the Pumping Lemma, there is some pumping length $n$, such that any word $w$ in $L$ of length at least $n$ can be split into $w = xyz$ satisfying the following conditions:

- $|xy| \leq n$,
- $|y| > 0$,
- $xy^iz \in L$ for all $i > 0$.

We let $w$ be the word with $n$ left parentheses, followed by $n$ right parentheses. Then by the first condition we know that $y$ consists only of left parentheses. By the second condition, we know that $y$ is nonempty. So the string $xyyz$ must have more left parentheses than right parentheses. Therefore, it must be unbalanced, so the third condition of the pumping lemma fails.

Thus $L$ cannot be regular. QED.

2. Let $L = \{0^n1^{2n}|n > 1\}$. Show that $L$ is not regular.

Assume $L$ is regular. Let $p$ be the pumping length, and let $w$ be the word $0^p1^{2p}$. Then when we break our word into $w = xyz$ as in the pumping lemma, we know that $|xy| \leq p$, and that $|y| \geq 0$. Therefore, $y$ contains only zeros, and is nonempty.

Now consider the string $w' = xyyz$. This will have more zeros than $w$, but the same number of ones. Therefore, it cannot be in the language. This contradicts the pumping lemma, so $L$ cannot be regular. QED.

3. Given two languages, $L$ and $M$, define the exclusive-or of $L$ and $M$ as the set of all strings $w$ such that $w$ is in $L$ and not in $M$ or $w$ is in $M$ and not in $L$. Show that the exclusive-or of two regular languages is regular.

Let $D_L$ be the DFA that accepts $L$, and let $D_M$ be the DFA that accepts $M$. Then we will construct a DFA that accepts the exclusive-or of $L$ and $M$.

We use the product DFA construction, which builds a DFA whose states are all of the pairs
of states in $D_L$ and $D_M$. Then we make a state in our product DFA a final state if and only if exactly one of its components is a final state in its original DFA.

Thus this DFA will accept a string whenever exactly one of the original DFA’s accepts. QED.

4. Suppose $L$ is a regular language over the alphabet $\{0,1\}$. Describe an algorithm to test whether $L = \{0,1\}^*$.

We perform a depth-first search on the DFA for $L$, to see if any rejecting state is reachable from the start state. If so, then $L$ is not $\{0,1\}^*$. But otherwise, every string must be accepted, so $L = \{0,1\}^*$.

5. Give an algorithm to tell, for two regular languages $L$ and $M$ over the alphabet $\{0,1\}$, whether there is a string from this same alphabet that is in neither $L$ nor $M$.

Let $D_L$ be a DFA accepting $L$, and let $D_M$ be a DFA accepting $D_M$. We build the product DFA $D$ for $D_L$ and $D_M$, and set the final states of $D$ to be those states in which both components are not final states. Then a string is accepted by $D$ if and only if it is rejected by both $D_L$ and $D_M$.

Finally, we test if the language of $D$ is empty. If so, then $L \cup M = \{0,1\}^*$. Otherwise, we can find a string that is not in $L$ or $M$. 