1. a) We start by defining the base cases i.e $n \leq 2$. These are all 0 because the size of a word which contains each of the three letters at least once must be greater than 2. For $n \geq 3$ we can use the principle of inclusion and exclusion. 3^n is the total number of words which can be built using the three letters, $3 \cdot 2^n$ are the words which are made only using 2 of the three letters. Finally we add the 3 words which formed only using one letter to compensate for doubles that were subtracted in the second term.

   - n $= 0 \Rightarrow 0$
   - n $= 1 \Rightarrow 0$
   - n $= 2 \Rightarrow 0$
   - n $\geq 3 \Rightarrow 3^n - 3 \cdot 2^n + 3$

   b) We define:

   - $a_n :=$ # of words which end in 1 with length n and do not contain 11 as a subword.
   - $b_n :=$ # of words which end in 0 with length n and do not contain 11 as a subword

   The solution to are task is: $sol_n := a_n + b_n$ We now derive the following two formulas:

   i. $b_n = b_{n-1} + a_{n-1} = sol_{n-1}$
   because we can append a 0 or a 1 to a word of size n-1 which ends in 0.

   ii. $a_n = b_{n-1} = b_{n-2} + a_{n-2} = sol_{n-2}$
   because words of size n-1 must end in a 0 for us to be able to append a 1

   for $n \geq 3$
   $\Rightarrow$

   - $sol_0 = 0$
   - $sol_1 = 2$
   - $sol_2 = 3$
   - $sol_{n \geq 3} = sol_{n-2} + sol_{n-1}$

2. a) We assume there is a non-empty finite language $L \neq \{\lambda\}$ satisfying $L^2 = L$ and hence it must contain a largest element x according to the canonical ordering.
   $\Rightarrow$ by definition of concatination there is an element $x' \in L^2$ such that $x' = x \cdot x$ (
   $\Rightarrow$ since $|x'| > |x|$ and x was the largest element in L it follows that $x' \notin L$
   $\Rightarrow$ No non-empty finite Language L exists such that $L \neq \{\lambda\}$ and $L^2 = L$

   b) We define the three languages as follows:

   - $L_1 = \{0\}^*$
   - $L_2 = \{0\}$
   - $L_3 = \{00\}$

   $\Rightarrow (L_2 \cap L_3) = \{0\} \cap \{00\} = \emptyset$
   $\Rightarrow L_1 \cdot \emptyset = \emptyset$
   $\Rightarrow L_1 \cdot (L_2 \cap L_3)$ is finite

   $L_1 L_2 = L_1^+$ and $L_1 L_3 = L_1 \backslash \{\lambda, 0\}$
   $\Rightarrow L_1^+ \cap L_1 \backslash \{\lambda, 0\} = L_1 \backslash \{\lambda, 0\}$
   $\Rightarrow L_1 \backslash \{\lambda, 0\}$ is by definition infinite

3. We must prove: An infinite language L is recursive $\iff$ there is an algorithm enumerating L
   " $\Rightarrow$ "
   Assume L is recursive, then there exists and Algorithm A which solves the Decision Problem (Entscheidungsproblem)
   $\Rightarrow$ We iterate through each element $x \in \sum^*$ in canonical order and decide with A if we add it to our enumeration.
   $\Rightarrow$ There is an algorithm enumerating L
   " $\Leftarrow$ "
   Assume there is an algorithm A which enumerates L
   We define an algorithm B which, when given an arbitrary $x \in \sum^*$ goes through the enumeration in canonical order and checks if x is equal to the current element e in the enumeration. If $|x| > |e|$ then B outputs "x not in L". Since $|x| < \infty$, B will terminate in a finite amount of time. If x == e then we output "x in L".
   $\Rightarrow$ B solves the Decision Problem for L
   $\Rightarrow$ L is recursive