Exercise 1: The Class $\mathcal{P}$ $(2+3+2+3$ Points$)$

$\mathcal{P}$ is the set of languages which can be decided by an algorithm whose runtime can be bounded by $p(n)$, where $p$ is a polynomial and $n$ the size of the respective input (problem instance). Show that the following languages (problems) are in the class $\mathcal{P}$. Since it is typically easy (i.e. feasible in polynomial time) to decide whether an input is well-formed, your algorithm only needs to consider well-formed inputs. Use the $\mathcal{O}$-notation to bound the run-time of your algorithm.

(a) $\text{PALINDROME} := \{ w \in \{0, 1\}^* \mid w \text{ is a Palindrome} \}$

(b) $\text{List} := \{ \langle A, c \rangle \mid A \text{ is a finite list of numbers which contains two numbers } x, y \text{ such that } x + y = c \}$.

(c) $\text{3-Clique} := \{ \langle G \rangle \mid G \text{ has a clique of size at least } 3 \}$

(d) $\text{17-DominatingSet} := \{ \langle G \rangle \mid G \text{ has a dominating set of size at most } 17 \}$

Remark: A dominating set for a graph $G = (V, E)$ is a set $D \subseteq V$ such that for every vertex $v \in V$, $v$ is either in $D$ or adjacent to a node in $D$.

Remark: A clique in a graph $G = (V, E)$ is a set $Q \subseteq V$ such that for all $u, v \in Q : \{u, v\} \in E$.

Exercise 2: The Class $\mathcal{NP}$ $(3$ Points$)$

Consider the following problem, called SUBSET-SUM. Given a collection $S$ of integers $x_1, \ldots, x_k$ and a target $t$, it is required to determine whether $S$ contains a sub-collection that adds up to $t$. Then, the problem can be given by

$\text{SUBSET-SUM} = \left\{ \langle S, t \rangle \mid S = \{x_1, \ldots, x_k\}, \text{and for some } \{y_1, \ldots, y_l\} \subseteq \{x_1, \ldots, x_k\} \text{ we have } \sum_i y_i = t \right\}$

Show that SUBSET-SUM is in $\mathcal{NP}$. 
Exercise 3: The Class \( \mathcal{NP} \)C

(7 Points)

Let \( L_1, L_2 \) be languages (problems) over alphabets \( \Sigma_1, \Sigma_2 \). Then \( L_1 \leq_p L_2 \) (\( L_1 \) is polynomially reducible to \( L_2 \)), iff a function \( f : \Sigma_1^* \to \Sigma_2^* \) exists, that can be calculated in polynomial time and

\[
\forall s \in \Sigma_1 : s \in L_1 \iff f(s) \in L_2.
\]

Language \( L \) is called \( \mathcal{NP} \)-hard, if all languages \( L' \in \mathcal{NP} \) are polynomially reducible to \( L \), i.e.

\[
L \text{ is } \mathcal{NP} \text{-hard } \iff \forall L' \in \mathcal{NP} : L' \leq_p L.
\]

The reduction relation ‘\( \leq_p \)’ is transitive (\( L_1 \leq_p L_2 \) and \( L_2 \leq_p L_3 \) \( \Rightarrow \) \( L_1 \leq_p L_3 \)). Therefore, in order to show that \( L \) is \( \mathcal{NP} \)-hard, it suffices to reduce a known \( \mathcal{NP} \)-hard problem \( \bar{L} \) to \( L \), i.e. \( \bar{L} \leq_p L \).

Finally a language is called \( \mathcal{NP} \)-complete (\( \iff \): \( L \in \mathcal{NP} \text{C} \)), if

1. \( L \in \mathcal{NP} \) and
2. \( L \) is \( \mathcal{NP} \)-hard.

Show \( \text{HittingSet} := \{\langle U, S, k \rangle | \text{universe } U \text{ has subset of size } \leq k \text{ that hits all sets in } S \subseteq 2^U \} \in \mathcal{NP} \text{C}.\)

Use that \( \text{VertexCover} := \{\langle G, k \rangle | \text{Graph } G \text{ has a vertex cover of size at most } k \} \in \mathcal{NP} \text{C}.\)

Remark: A hitting set \( H \subseteq U \) for a given universe \( U \) and a set \( S = \{S_1, S_2, \ldots, S_m\} \) of subsets \( S_i \subseteq U \), fulfills the property \( H \cap S_i \neq \emptyset \) for \( 1 \leq i \leq m \) (\( H \) ’hits’ at least one element of every \( S_i \)).

A vertex cover is a subset \( V' \subseteq V \) of nodes of \( G = (V, E) \) such that every edge of \( G \) is adjacent to a node in the subset.

Hint: For the poly. transformation (\( \leq_p \)) you have to describe an algorithm (with poly. run-time!) that transforms an instance \( \langle G, k \rangle \) of \( \text{VertexCover} \) into an instance \( \langle U, S, k \rangle \) of \( \text{HittingSet} \), s.t. a vertex cover of size \( \leq k \) in \( G \) becomes a hitting set of \( U \) of size \( \leq k \) for \( S \) and vice versa(!).