1. Let $G = (V, E)$ be an undirected graph. To avoid some simple but unnecessary special-case reasoning, let us assume that $G$ is connected.

By a *dominating set* $S \subseteq V$, we mean a set of vertices of $G$ such that every vertex $v \in V$ is either also in $S$ or is a neighbor of a vertex in $S$.

The DOMINATING SET PROBLEM is this:

**Problem name:** DOMINATING SET

**Instance:** An undirected graph $G = (V, E)$ and a positive integer $k$.

**Question:** Does $V$ contain a dominating set $S$ of size $k$?

(a) Carefully state the difference between DOMINATING SET and VERTEX COVER.

(b) Draw an example of a graph together with a dominating set that is not a vertex cover.

(c) Show that DOMINATING SET is in NP.

(d) Show that DOMINATING SET is NP-hard. You can do this by reducing VERTEX COVER to it, in the following way: If $G = (V, E)$ is an undirected (and connected) graph, construct $G' = (V', E')$ as follows: For every edge $(x, y)$ in $E$, add a new vertex $z$ with edges $(x, z)$ and $(z, y)$. Show that this leads immediately to a reduction

$$
\text{VERTEX COVER} \leq_{P} \text{DOMINATING SET}
$$

(e) Conclude that DOMINATING SET is NP-complete.

2. Here is a variant of 3-SAT which we will call NAE-3-SAT (“Not-All-Equal 3-SAT”):

**Problem name:** NAE-3-SAT

**Instance:** A Boolean expression in conjunctive normal form, where each clause is composed of 3 literals.

**Question:** Is there an assignment of truth values to each variable such that each clause has at least one literal that is True and at least one literal that is False?

In fact, let us consider a small generalization of this problem: For each integer $k \geq 2$, we define a problem NAE-$k$-SAT

**Problem name:** NAE-$k$-SAT

**Instance:** A Boolean expression in conjunctive normal form, where each clause is composed of $k$ literals.
**Question:** Is there an assignment of truth values to each variable such that each clause has at least one literal that is **True** and at least one literal that is **False**?

(a) Show that NAE-\(k\)-SAT is in NP for each \(k\).

(b) Show that

\[
\text{NAE-4-SAT} \leq_p \text{NAE-3-SAT}
\]

(In fact the same argument shows that for each \(k \geq 2\),

\[
\text{NAE-}(k + 1)\text{-SAT} \leq_p \text{NAE-}k\text{-SAT}
\]

But all we need is the result for \(k = 3\).)

(c) Now we need to show that 3-SAT \(\leq_p\) NAE-4-SAT. Here’s how: Suppose we have a 3-SAT expression. Call it \(\phi\). Introduce a new Boolean variable \(s\), different from any of the variables in \(\phi\). Let \((l_1 \lor l_2 \lor l_3)\) be any clause in \(\phi\). Construct a new clause \((l_1 \lor l_2 \lor l_3 \lor s)\). Do this for every original clause in \(\phi\), and create a new Boolean expression \(\phi'\) from these new clauses by “and”-ing them all together. And remember that \(s\) is the same in each of these new clauses.

(d) It should be clear that you can create \(\phi'\) from \(\phi\) by a polynomial-time algorithm, but you do need to say so, right?

(e) We then need to show that \(\phi\) is satisfiable iff \(\phi'\) has a NAE-4-SAT solution.

Here are some hints:

i. If \(\phi\) is satisfiable, then there is an assignment of truth values to the original variables so that at least one literal in each clause of \(\phi\) is **True**. Then you can just set \(s = \text{False}\), right?

ii. Going in the other direction, suppose \(\phi'\) has an NAE-4-SAT solution. There are two possibilities:

- \(s = \text{False}\). In this case, you should be able to show that \(\phi\) is 3-SAT satisfiable.
- \(s = \text{True}\). In this case, we know that at least one of the original literals in each clause is **False**. So if we reverse the truth values of those original variables, we see that \(\phi\) has a 3-SAT solution.

You need to write this all out clearly, so that someone other than you can understand it.

(f) Finally, you need to put all this together to show that NAE-3-SAT is NP-complete.