Question 1
Recall that a dominating set for a graph $G$ is a subset of nodes $S$ such that any node $i$ is either in $S$ or is adjacent to at least one node in $S$. We have seen that it is NP-Complete to determine whether or not a graph has a dominating set of size $k$. However, if $G$ is a tree, then this is no longer the case.

Suppose that every node $i$ has a weight $w_i$. Give a polyme algorithm that takes in a tree $T$ and weights for all nodes, and returns the weight of the minimum weight dominating set.

Question 2
Suppose you are given a graph $G = (V, E)$, with costs $c_e$ for each edge $e$, and a set $V' \subseteq V$ of terminal nodes. Your goal is to find a minimum-cost tree $T \subseteq G$ that spans the vertices of $V'$. While $T$ must include all the nodes of $V'$, in building $T$ you can choose to either use or not use any of the other nodes in $V$. The cost of $T$ is just the sum of the costs of the edges in $T$. As shown in the example below, including additional nodes can sometimes actually yield a cheaper tree. Any vertex in $T$ but not in $V'$ is called a Steiner node, and $T$ itself is called a Steiner tree. The problem of determining whether there is a Steiner Tree of cost $\leq C$ is known as the STEINER MINIMUM TREE problem. In the example below, the terminals are the three bold nodes, and the min cost Steiner tree includes the middle non-terminal, or Steiner node.
(a) Show that SMT is NP-Hard, using a reduction from SET COVER.

(b) Suppose the set of terminals $V'$ is only two nodes, namely $t_1$ and $t_2$. Show that SMT is solvable in polytime by giving an algorithm that solves it.

(c) Suppose the set of terminals $V'$ is only three nodes, namely $t_1$, $t_2$, and $t_3$. Show that SMT is solvable in polytime by giving an algorithm that solves it.

(d) Suppose our set of vertices $V$ are the points in the Euclidean plane. In particular, the vertices of $V$ obey the triangle inequality: $d(x, y) + d(y, z) \leq d(x, z)$ $\forall x, y, z \in \mathbb{R}^2$, where $d$ is just the Euclidean distance function. Let the edge $(x, y)$ have cost $d(x, y)$, for any two points $x$ and $y$ in the plane. Now suppose you are given $V' \subseteq V$, a set of points you want to connect using as cheap a tree as possible. Consider finding the minimum cost Steiner tree, and think about what we did in class for the travelling salesman problem. Show that

$$\frac{|\text{cost}(\text{SMT})|}{|\text{cost}(\text{MST})|} \geq 1 \frac{1}{2}$$

where $\text{SMT}$ is the Steiner minimum tree on $V'$, and $\text{MST}$ is the minimum spanning tree on $V'$.

Question 3

Consider the following problem: you are given a set of obscure documents that you wish to classify into different topics. Unfortunately, you aren’t really sure which topics each document falls into, seeing as the material is beyond your area of expertise. What you do have at your disposal though, is a function that, given two documents $x$ and $y$, returns a ‘+’ if $x$ and $y$ are similar, and a ‘−’ if $x$ and $y$ are different, topic-wise. You would like to group, or cluster the documents into as many stacks as necessary so that related materials are in the same stacks, and any two unrelated documents are in different stacks. Unfortunately, this isn’t always possible to do, so you decide you would like to cluster the documents in such a way that you make as few mistakes as possible, such as placing unrelated documents in the same stack, or two related documents in different stacks.

In particular, you are given a complete graph $G = (V, E)$, where each edge $(u, v)$ is either labelled + or −, depending on whether $u$ and $v$ are considered to be similar or
different. Your goal is to produce a partition (clustering) of the vertices that agrees as much as possible with the edge labels. That is, we want a clustering that maximizes the number of agreements: the number of + edges within clusters, plus the number of − edges between clusters, or equivalently, minimizes the number of disagreements: the number of − edges inside clusters plus the number of + edges between clusters.

(a) A perfect clustering is a clustering that correctly places all the edges, i.e. all the positive edges are within clusters, and all negative edges cross between different clusters. Give a simple example demonstrating that a perfect clustering doesn’t always exist.

(b) Suppose some little (honest) birdie flies by and tells you that there is perfect clustering of your graph $G$. Come up with an algorithm that outputs that perfect clustering.

(c) Part (a) tells us it is not always possible to obtain a perfect clustering, which is why we are concerned with finding clusters that maximize agreements or minimize disagreements. Give a 2-approximation for maximizing the number of agreements. I.e. an algorithm that produces a clustering that agrees with at least half the edge labels.

(d) Explain why your algorithm for (c) will not yield a 2-approximation for the objective of minimizing the number of disagreements, i.e. the number of + edges between clusters plus the number of − edges inside clusters.