1. Monsters in disguise
Prove the NP-Completeness of the following by showing them to be generalizations of known NP-Complete problems

i) HITTING SET: Given a family of sets \( \{ S_1, S_2, \ldots, S_n \} \) and a budget \( b \), find a set \( H \) of size at most \( b \) such that \( H \cap S_i \neq \emptyset \) for all \( i \).

ii) SET PACKING: Given a collection \( C \) of finite sets \( \{ S_1, S_2, \ldots, S_n \} \) and an integer \( K \leq n \), find \( K \) disjoint sets in \( C \).

iii) MULTIPROCESSOR SCHEDULING: You are given a set \( A \) of tasks and a positive integer \( l(a) \forall a \in A \), which is the length of each task. Also, the input includes a number \( m \) of the processors and a deadline \( D \). Find a partition \( \{ A_1, A_2, \ldots, A_m \} \) of the tasks between the processors such that

\[
A = A_1 \cup A_2 \cup \ldots \cup A_m
\]

and

\[
\max_{1 \leq i \leq m} \left\{ \sum_{a \in A_i} l(a) \right\} \leq D
\]

i.e. the each processor finishes all the tasks allotted to it before the deadline.

2. Its difficult to dominate
For a graph \( G = (V, E) \), a set \( D \subseteq V \) is said to be a dominating set if every \( v \in V \) is either in \( D \) or is adjacent to some vertex in \( D \). Give a graph \( G \) and a budget \( b \), the DOMINATING SET problem is to find a dominating set in the graph of size at most \( b \), if one exists. Prove that it is NP-Complete. (Hint: Does this remind you of VERTEX-COVER?)

3. Branch, if you must...
0/1-ILP is the problem of Integer Linear Programming when the variables are allowed to take only the value 0 and 1. This is NP-Complete (recall the reduction from 3SAT in the last handout, and also the fact that this is a generalization of ZOE). Design a branch-and-bound algorithm for 0/1-ILP specifying clearly your subproblems and method of finding a lowerbound (or upperbound if the given problem is a maximization problem).

4. Adding colors to life
\( k \)-COLORABILITY is the problem of finding an assignment of 1 color to each vertex of a given graph \( G \), out of a total of \( k \) colors, such that no two adjacent vertices have the same color. These are some of the hardest NP problems to even approximate - the best known algorithm may use as many as \( O(n^{0.2111}) \) colors to color a graph which is actually 3-colorable. 2-colorability, however, can be solved in polynomial time (recall previous homeworks).

Here we construct a reduction to show that 3-COLORABILITY is NP-Complete by reducing 3SAT to 3-COLORABILITY. We have the following components in the graph to “simulate” a 3SAT formula:
i) A triangle to represent the states \texttt{true}, \texttt{false} and a third state \texttt{don’t-care}. This is because all 3 vertices of a triangle must be colored differently and we can interpret each color as stated above. We’ll now use the numbers 1, 2 and 3 for the colors corresponding to \texttt{true}, \texttt{false} and \texttt{don’t-care}.

ii) For each variable \( x \) we have two vertices, one for \( x \) and one for \( \overline{x} \).

iii) A “sort of” OR-gate as shown below, which has the property that \( y_2 \) can be of color 1 (\texttt{true}) if and only if one of \( a \) and \( b \) is true, where \( a \) and \( b \) are restricted to be either \texttt{true} or \texttt{false}.

\[ \begin{align*}
  & a \quad y_1 \\
  & b \quad y_3 \\
  & \quad y_2
\end{align*} \]

We now carry out the reduction in steps:

a) Prove the property of the OR-gate.

b) Connect the variables to the truth-triangle appropriately to ensure that each vertex \( x_i \) and \( \overline{x_i} \) is only colored \texttt{true} or \texttt{false}. Also, ensure that \( x_i \) and \( \overline{x_i} \) do not get the same color.

c) Use the OR-gate to construct a gadget to check if a 3-clause is \texttt{true} using the colors of the 3 literals appearing in it.