This lab is on reductions. The first problem emphasizes the care one needs in making sure that a reduction is correct. The second one is about the notion of self-reductions; how one can reduce search and optimization problems to decision versions in many settings.

1. Let $G = (V, E)$ be a graph. A set of edges $M \subseteq E$ is said to be a matching if no two edges in M intersect at a vertex. A matching M is *perfect* if every vertex in V is incident to some edge in M; alternatively M is perfect if $|M| = |V|/2$ (which in particular implies $|V|$ is even). See Wikipedia article for some example graphs and further background.

   The PerfectMatching problem is the following: does the given graph G have a perfect matching? This can be solved in polynomial time which is a fundamental result in combinatorial optimization with many applications in theory and practice. It turns out that the PerfectMatching problem is easier to solve in *bipartite* graphs. A graph $G = (V, E)$ is bipartite if its vertex set V can be partitioned into two sets L, R (left and right say) such that all edges are between L and R (in other words L and R are independent sets). Here is an attempted reduction from general graphs to bipartite graphs.

   Given a graph $G = (V, E)$ create a bipartite graph $H = (V \times \{1, 2\}, E_H)$ as follows. Each vertex u is made into two copies $(u, 1)$ and $(u, 2)$ with $V_1 = \{(u, 1) \mid u \in V\}$ as one side and $V_2 = \{(u, 2) \mid u \in V\}$ as the other side. Let $E_H = \{((u, 1), (v, 2)) \mid (u, v) \in E\}$. In other words we add an edge betwen $(u, 1)$ and $(v, 2)$ iff (u, v) is an edge in E. Note that $((u, 1), (u, 2))$ is not an edge in H for any $u \in V$ since there are no self-loops in G.

   Is the preceding reduction correct? To prove it is correct we need to check that H has a perfect matching if and only if G has one.

   - Prove that if G has perfect matching then H has a perfect matching.
   - Consider G to be K_3 the complete graph on 3 vertices (a triangle). Show that G has no perfect matching but H has a perfect matching.
   - Extend the previous example to obtain a graph G with an even number of vertices such that G has no perfect matching but H has.

   Thus the reduction is incorrect although one of the directions is true.

2. The traveling salesman problem can be defined in two ways:

   - The Traveling Salesman Problem
     - Input: A weighted graph G
     - Output: The tour $(v_1, v_2, \ldots, v_n)$ that minimizes $\sum_{i=1}^{n-1} (d[v_i, v_i + 1]) + d[v_n, v_1]$
   - The Traveling Salesman *Decision* Problem
     - Input: A weighted graph G and an integer k
     - Output: TRUE if there exists a TSP tour with cost $\leq k$, FALSE otherwise

   Suppose we are given an algorithm that can solve the traveling salesman decision problem in (say) linear time. Give an efficient algorithm to find the actual TSP tour by making a polynomial number of calls to this subroutine.

3. A **Hamiltonian cycle** in a graph is a cycle that visits every vertex exactly once. A **Hamiltonian path** in a graph is a path that visits every vertex exactly once, but it need not be a cycle (the last vertex in the path may not be adjacent to the first vertex in the path.)

   Consider the following three problems:

   - *Directed Hamiltonian Cycle* problem: checks whether a Hamiltonian cycle exists in a *directed* graph,
   - *Undirected Hamiltonian Cycle* problem: checks whether a Hamiltonian cycle exists in an *undirected* graph.
   - *Undirected Hamiltonian Path* problem: checks whether a Hamiltonian path exists in an *undirected* graph.

   (a) Give a polynomial time reduction from the *directed* Hamiltonian cycle problem to the *undirected* Hamiltonian cycle problem.

   (b) Give a polynomial time reduction from the *undirected* Hamiltonian Cycle to *directed* Hamiltonian cycle.

   (c) Give a polynomial-time reduction from undirected Hamiltonian *Path* to undirected Hamiltonian *Cycle*.

4. An ***independent set*** in a graph G is a subset S of the vertices of G, such that no two vertices in S are connected by an edge in G. Suppose you are given a magic black box that somehow answers the following decision problem *in polynomial time*:

   - INPUT: An undirected graph G and an integer k.
   - OUTPUT: TRUE if G has an independent set of size k, and FALSE otherwise.

   (a) Using this black box as a subroutine, describe algorithms that solves the following *optimization* problem *in polynomial time*:

   - INPUT: An undirected graph G.
   - OUTPUT: The *size* of the largest independent set in G.

   (b) Using this black box as a subroutine, describe algorithms that solves the following *search* problem *in polynomial time*:

   - INPUT: An undirected graph G.
   - OUTPUT: An independent set in G of maximum size.

To think about later:

5. Formally, a ***proper coloring*** of a graph $G = (V, E)$ is a function $c\colon V \to \{1, 2, \dots, k\}$, for some integer k, such that $c(u) \neq c(v)$ for all $uv \in E$. Less formally, a valid coloring assigns each vertex of G a color, such that every edge in G has endpoints with different colors. The ***chromatic number*** of a graph is the minimum number of colors in a proper coloring of G.

   Suppose you are given a magic black box that somehow answers the following decision problem *in polynomial time*:

   - INPUT: An undirected graph G and an integer k.
   - OUTPUT: TRUE if G has a proper coloring with k colors, and FALSE otherwise.

   Using this black box as a subroutine, describe an algorithm that solves the following ***coloring problem*** *in polynomial time*:

   - INPUT: An undirected graph G.
   - OUTPUT: A valid coloring of G using the minimum possible number of colors.

   *[Hint: You can use the magic box more than once. The input to the magic box is a graph and **only** a graph, meaning **only** vertices and edges.]*