1. Describe a polynomial-time reduction from 3SAT to ILP-FEASIBILITY. Your reduction implies that ILP-FEASIBILITY is NP-hard.

Solution: Let Φ be an arbitrary 3CNF formula, with variables $x_1, x_2, \ldots, x_n$. We create an integer linear program with one *integer* variable $z_i \in \{0,1\}$ for each *boolean* variable x_i of Φ, and one inequality constraint for each clause in Φ. Specifically, for each clause, we create a linear inequality by

- replacing each positive literal x_i with z_i,
- replacing each negative literal $\overline{x_i}$ with $(1-z_i)$,
- replacing each $\vee$ with $+$, and
- requiring the resulting expression to be at least 1.

The integer values 0 and 1 correspond to the boolean values FALSE and TRUE, respectively; thus, the expression $1-z$ is equivalent to Boolean negation, and addition is (crudely) equivalent to Boolean OR. To enforce $z_i \in \{0,1\}$ for every index i, we add constraints $z_i \geq 0$ and $z_i \leq 1$ and (automatically, because this is an integer linear program) $z_i \in \mathbb{Z}$ for every index i. The objective function doesn't matter, because we only care about feasibility.

For example, if $\Phi = (x_1 \vee x_2 \vee x_3) \wedge (x_2 \vee \overline{x_3} \vee \overline{x_4}) \wedge (\overline{x_1} \vee x_3 \vee x_4) \wedge (x_1 \vee \overline{x_2} \vee \overline{x_4})$, the resulting ILP is

$$
\begin{array}{ll}
\text{minimize} & \langle\langle \textit{Whatever} \rangle\rangle \\
\text{subject to} & z_1 + z_2 + z_3 \geq 1 \\
& z_2 + (1-z_3) + (1-z_4) \geq 1 \\
& (1-z_1) + z_3 + z_4 \geq 1 \\
& z_1 + (1-z_2) + (1-z_4) \geq 1 \\
& 0 \leq z_i \leq 1 \quad \text{for all } i \\
& z_i \in \mathbb{Z} \quad \text{for all } i
\end{array}
$$

Now I claim that Φ is satisfiable if and only if the ILP is feasible.

$\Rightarrow$ Suppose Φ is satisfiable. Fix arbitrary values to the variables x_i that satisfy Φ. For each index i, define $z_i = [x_i]$; that is, $z_i = 1$ if $x_1 = \text{TRUE}$, and $z_i = 0$ if $x_i = \text{FALSE}$. We immediately have $1 - z_i = [\overline{x_i}]$ for all i. It follows that the left side of every inequality in our ILP is an integer between 0 and 3.

Now consider an arbitrary clause of Φ. At least one literal x_i or $\overline{x_i}$ in this clause is TRUE, so the corresponding term z_i or $(1-z_i)$ in the corresponding ILP constraint is equal to 1, and therefore the constraint is satisfied. We conclude that the integer vector $(z_1, z_2, \ldots, z_n)$ satisfies all constraints of the ILP, which means the ILP is feasible.

$\Leftarrow$ Conversely, suppose the ILP is feasible. Fix an arbitrary feasible vector $(z_1, z_2, \ldots, z_n)$; by definition every z_i is either 0 or 1. For each index i, define $x_i = (z_i = 1)$; that is, $x_1 = \text{TRUE}$ if z_i, and $x_i = \text{FALSE}$ if $z_i = 0$. We immediately have $\overline{x_i} = (z_i = 0) = ((1-z_i) = 1)$ for all i.

Now consider an arbitrary constraint in the ILP. Because each term z_i or $(1-z_i)$ is either 0 or 1, this constraint must contain at least one term z_i or $(1-z_i)$ that is equal to 1. The corresponding literal x_i or $\overline{x_i}$ is TRUE, and therefore the corresponding clause is satisfied. We conclude that $x_1, x_2, \ldots, x_n$ satisfy every clause of Φ, which means Φ is satisfiable.

The reduction runs in polynomial time. ∎

> **Rubric:** 10 points = 3 for reduction + 3 for "if" proof + 3 for "only if" proof + 1 for "polynomial time". (This is the standard NP-hardness rubric.) This is more detail than necessary for full credit.

2. (a) Describe a polynomial-time reduction from the *undirected* Hamiltonian cycle problem to the *directed* Hamiltonian cycle problem. Prove your reduction is correct.

Solution: Given an undirected graph $G = (V, E)$, we construct a directed graph $G' = (V, E')$ by replacing each undirected edge uv with two directed edges $u \to v$ and $v \to u$. I claim that G contains a Hamiltonian cycle if and only if H contains a Hamiltonian cycle.

Suppose G contains a Hamiltonian cycle $\gamma = v_0 \to v_1 \to v_2 \to \cdots \to v_{n-1} \to v_0$, where $v_i v_{i+1 \bmod n} \in E$ for every index i. The definition of G' immediately implies that $v_i \to v_{i+1 \bmod n} \in E'$ for every index i. If follows that γ is also a cycle in G', and therefore a Hamiltonian cycle in G'.

Suppose G' contains a Hamiltonian cycle $\gamma = v_0 \to v_1 \to v_2 \to \cdots \to v_{n-1} \to v_0$, where $v_i \to v_{i+1 \bmod n} \in E'$ for every index i. The definition of G' immediately implies that $v_i v_{i+1 \bmod n} \in E$ for every index i. If follows that γ is also a cycle in G, and therefore a Hamiltonian cycle in G.

The reduction runs in polymomial time. ■

Rubric: 3 points = 1 for reduction + 1 for if proof + 1 for only if proof + 0 for "polynomial time"

(b) Describe a polynomial-time reduction from the *directed* Hamiltonian cycle problem to the *undirected* Hamiltonian cycle problem. Prove your reduction is correct.

Solution: Given an directed graph $G = (V, E)$, we construct an undirected graph $G' = (V', E')$ as follows:

- $V' = \{v^-, v^\circ, v^+ \mid v \in V\}$
- $E' = \{v^- v^\circ, v^\circ v^+ \mid v \in V\} \cup \{u^+ v_- \mid u \to v \in E\}$

That is, every vertex in G is represented by a path of three vertices in G', and every directed edge in G is represented by an edge from the end of one vertex chain to the beginning of another. I claim that G contains a Hamiltonian cycle if and only if G' contains a Hamiltonian cycle.

First, suppose G contains a Hamiltonian cycle $\gamma = v_0 \to v_1 \to v_2 \to \cdots \to v_{n-1} \to v_0$, where $v_i \to v_{i+1 \bmod n} \in E$ for every index i. Then G' contains the Hamiltonian cycle $v_0^\circ \to v_0^+ \to v_1^- \to v_1^\circ \to \cdots \to v_{n-1}^+ \to v_0^- \to v_0^\circ$, obtained by replacing every directed edge $u \to v$ in γ with the path $u^\circ \to u^+ \to v^- \to v^\circ$ in G'.

Conversely, suppose G' contains a Hamiltonian cycle γ'. This cycle must pass through every middle vertex v°; the cycle must visit the neighbors of v° immediately before and after visiting v°. Orient γ' so that it contains some subpath $v^- \to v^\circ \to v^+$. Every neighbor of a "positive" vertex v^+ except v° is a "negative" vertex w^-; thus, by induction, γ' traverses every vertex gadget in the order $v^- \to v^\circ \to v^+$. We conclude that $\gamma' = v_0^\circ \to v_0^+ \to v_1^- \to v_1^\circ \to \cdots \to v_{n-1}^+ \to v_0^- \to v_0^\circ$ for some indexing of the vertex gadgets. The corresponding cycle $\gamma = v_0 \to v_1 \to v_2 \to \cdots \to v_{n-1} \to v_0$ is a Hamiltonian cycle in G.

The reduction runs in polynomial time. ■

Rubric: 6 points = 2 for reduction + 2 for if proof + 2 for only if proof + 0 for "polynomial time"

(c) Which of these two reductions implies that the *undirected* Hamiltonian cycle problem is NP-hard?

Solution: The second one. ■

Rubric: 1 point

3. Suppose you are given a magic black box that can determine in polynomial time, whether an arbitrary given 3CNF formula is satisfiable. Describe and analyze a polynomial-time algorithm that either computes a satisfying assignment for a given 3CNF formula or correctly reports that no such assignment exists, using the magic black box as a subroutine.

Solution: First, suppose we actually have a more powerful black box SATISFIABLE that does not require every clause of the input formula to have exactly three distinct literals. Then we can construct a satisfying assignment for any 3CNF formula in polynomial time as follows. In each iteration, we add a one-literal clause to the formula, consisting either of a variable x_i or its negation $\overline{x_i}$. The key insight is that every satisfying assignment for the formula $\Phi \wedge x_i$ is a satisfying assignment for Φ such that $x_i = \text{TRUE}$.

```
FindSatAssignment(Φ):
  if ¬Satisfiable(Φ)
      return None
  for i ← 1 to n
      if Satisfiable(Φ ∧ x_i)
          Φ ← Φ ∧ x_i
          X[i] ← True
      else
          Φ ← Φ ∧ ¬x_i
          X[i] ← False
  return X[1..n]
```

To adapt this strategy to our original black box, we introduce two new variables y and z, and then in each iteration we add four clauses to the formula. Specifically, for each index i, we define two CNF formulas

$$\Phi_T = \Phi \wedge (x_i \vee y \vee z) \wedge (x_i \vee \overline{y} \vee z) \wedge (x_i \vee y \vee \overline{z}) \wedge (x_i \vee \overline{y} \vee \overline{z}) \text{ and}$$
$$\Phi_F = \Phi \wedge (\overline{x_i} \vee y \vee z) \wedge (\overline{x_i} \vee \overline{y} \vee z) \wedge (\overline{x_i} \vee y \vee \overline{z}) \wedge (\overline{x_i} \vee \overline{y} \vee \overline{z}).$$

Every satisfying assignment for Φ also satisfies either Φ_T (if $x_i = \text{TRUE}$) of Φ_F (if $x_i = \text{FALSE}$). Conversely, every satisfying assignment for Φ_T (or Φ_F) is a satisfying assignment for Φ such that $x_i = \text{TRUE}$ (or $x_i = \text{FALSE}$, respectively).

```
FindSatAssignment(Φ):
  if ¬3Sat(Φ)
      return None
  for i ← 1 to n
      Φ_T ← Φ ∧ (x_i ∨ y ∨ z) ∧ (x_i ∨ ¬y ∨ z) ∧ (x_i ∨ y ∨ ¬z) ∧ (x_i ∨ ¬y ∨ ¬z)
      Φ_F ← Φ ∧ (¬x_i ∨ y ∨ z) ∧ (¬x_i ∨ ¬y ∨ z) ∧ (¬x_i ∨ y ∨ ¬z) ∧ (¬x_i ∨ ¬y ∨ ¬z)
      if 3Sat(Φ_T)
          Φ ← Φ_T
          X[i] ← True
      else
          Φ ← Φ_F
          X[i] ← False
  return X[1..n]
```

Suppose the original input formula Φ has n variables and $m = O(n^3)$ clauses. Then every formula passed to 3SAT has $n+2$ variables and at most $m + 4n = O(n^3)$ clauses. Thus, the algorithm runs in polynomial time. ■

Rubric: 10 points = 4 for reduction using more powerful black box + 4 for full reduction + 2 for time analysis. This is not the only correct solution.