1. (6 points) For function $f : \{1, \ldots, n\} \to \{1, \ldots, n\}$ and an integer $k \geq 1$, define $f^k : \{1, \ldots, n\} \to \{1, \ldots, n\}$ to be the function obtained by iterating $f$ for $k$ times. (For example, if $f(x) = n + 1 - x$, then $f^k(x) = n + 1 - x$ for odd $k$ and $f^k(x) = x$ for even $k$.)

Give an algorithm that given input $f$ and $k$, computes $f^k$ in time polynomial in $n$ and $\log k$.

2. (8 points) This problem is about a special form of the satisfiability problem called HornSAT. We first state some useful definitions.

A positive literal is a Boolean variable. A negative literal is a negated Boolean variable.

A Horn CNF formula (for this problem) is defined as a Boolean formula in conjunctive normal form (a cnf-formula) where each clause has at most 3 literals of which at most one is a positive literal.

Let

$$\text{HornSAT} = \{\langle \phi \rangle \mid \phi \text{ is a satisfiable Horn CNF formula} \}$$

Prove that HornSAT ∈ P.

[Hint: First consider the clauses containing just one literal, then consider the others.]

3. (6 points) Show that, if P ≠ NP, then P ≠ co-NP.

4. (10 points)

Suppose that NASA is planning to send the unmanned spaceship Plutonic to explore the (dwarf) planet Pluto.

Due to a severe budget crunch, Plutonic is being designed with sensors and actuators of very limited capability. It “knows” and controls its physical parameters (e.g., its position, velocity, etc.) only through a set of Boolean variables $x_1, x_2, \ldots, x_n$ representing Boolean predicates on its environment. For example, $x_{42} = 1$ might mean that it is at a safe distance from Jupiter. We refer to $(x_1, x_2, \ldots, x_n)$ as a state. Plutonic can change its state by flipping the values of the $x_i$s. Flipping bits is expensive (uses up scarce power), so the fewer $x_i$s it flips, the better it is.

Plutonic “knows” that an error has occurred if a Boolean function $E(x_1, x_2, \ldots, x_n)$ evaluates to 1. If an error occurs in state $s$, Plutonic tries to flip some $x_i$s to reach a different state.
\( \hat{s} = (\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_n) \) such that \( E(\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_n) = 0 \). To conserve power, Plutonic needs to flip the fewest number of bits to move from an error state \( s \) to some non-error state \( \hat{s} \) (note: \( E(s) = 1, E(\hat{s}) = 0 \)).

NASA engineers know that both \( E \) and its negation \( \overline{E} \) are always satisfiable. We will call such a Boolean function \( E \) an error Boolean function. What makes matters hard is that \( E \) can be an arbitrary error Boolean function that varies over Plutonic’s lifetime, so they must design Plutonic’s control software to flip the minimum number of bits for any error Boolean function \( E \) to move from any error state \( s \) to some non-error state.

Consider the problem of finding the minimum number of bits to flip, expressed as a language:

\[
A = \{ (E, k, s) \mid E \text{ is an error Boolean function requiring less than } k \text{ bit flips to move from error state } s \text{ to some non-error state} \}
\]

Prove that this language is NP-complete.

[Hints: (1) Use a reduction from SAT; (2) remember that you can pick \( k \) and \( s \) while performing the reduction – this choice is important.]