NEW FEATURE: If your homework is typeset (as opposed to handwritten), you will receive 1 bonus point.

1. **(Problems in NP.)** (10 points.)
   
   (a) (5 points.) Let $HMLCS$ ("half-length multi longest common subsequence") be the language of all lists $\langle \omega_1, \omega_2, \ldots, \omega_m \rangle$, where:
   
   - $\omega_1, \ldots, \omega_m \in \{0, 1\}^\ell$ for some $\ell \in \mathbb{N}$;
   - $\omega_1, \ldots, \omega_m$ have a common subsequence $z \in \{0, 1\}^k$ with $k \geq \ell/2$.

   Prove that $HMLCS \in \text{NP}$.
   
   (b) (5 points.) Let $DCF$ ("different circuit functionality") denote the language of all pairs $\langle C_1, C_2 \rangle$ where:
   
   - $C_1$ and $C_2$ are Boolean circuits;
   - $C_1$ and $C_2$ have the same number of input gates (we will refer to this number as $n$, bearing in mind it’s not the same as $|\langle C_1, C_2 \rangle|$);
   - $C_1$ and $C_2$ do not compute the same function $\{0, 1\}^n \rightarrow \{0, 1\}$.

   Prove that $DCF \in \text{NP}$.

2. **(NP in EXP.)** (10 points.) Prove $\text{NP } \subseteq \text{EXP}$.

3. **(Unit Clauses and Horn-SAT.)** In this problem, you will want to use the convention that an “empty clause” (i.e., a clause containing 0 literals) is equivalent to $\bot$ (False). This makes sense: the definition of an “OR”-clause is that it is $\top$ (True) iff at least one literal in it is $\top$; so if there are no literals in it, it’s indeed vacuously $\bot$. Similarly, you will want to use the convention that an “empty CNF” (i.e., one with no clauses in it) is equivalent to $\top$. Again this makes sense: the definition of an “AND” of clauses is that it is $\top$ iff all of its clauses are $\top$; so if it has no clauses, it’s indeed vacuously $\top$.

   (a) (5 points.) Let $C$ be a CNF formula for which we are interested in deciding satisfiability. A *unit clause* in $C$ is simply a clause with one literal, so either $x_i$ or $\overline{x_i}$. If $C$ has a unit clause, say $x_i$, the following is an “obvious” thing to do: for every clause where $x_i$ appears, delete that clause; and, for every clause where $\overline{x_i}$ appears, delete $\overline{x_i}$ from that clause. Similarly, if $C$ contains the unit clause $\overline{x_i}$, the “obvious” thing to do is delete every clause containing $\overline{x_i}$ and delete $x_i$ from every clause. In either case, doing this “obvious” thing is called doing unit clause propagation.

   Prove that doing unit clause propagation preserves the satisfiability of $C$; i.e., when you do it, if $C$ was satisfiable before then it is satisfiable afterward, and if it was unsatisfiable before then it is unsatisfiable afterward.

   (b) (5 points.) A formula $C$ is called a *Horn-CNF* if every clause contains at most one positive literal. Prove that HORN-SAT, the task of deciding whether an input Horn-CNF is satisfiable or not, is solvable in polynomial time. (Hint: Given a Horn-CNF, split into two cases: (i) every clause contains at least one *negative* literal; (ii) there is a clause containing zero negative literals.)
4. **(XOR-SAT.)** XOR-SAT is a problem similar to CNF-SAT, except instead of all the clauses being ORs of literals, all the clauses are XORs of literals. E.g., an input might look like this:

\[(x_1 \oplus x_2 \oplus x_3 \oplus x_4) \land (\overline{x}_2 \oplus x_3) \land \cdots \land (\overline{x}_7 \oplus x_8 \oplus x_n),\]

where \(\oplus\) denotes XOR. As usual, the XOR-SAT problem is to determine if there is a truth assignment to the variables that satisfies the whole formula.

(a) (2 points.) Show that we can equivalently think of an XOR-SAT input as a *system (collection) of equations mod 2*; meaning a system where the variables are supposed to take values in \(\{0, 1\}\), and every equation is of the form

\[x_{i_1} + x_{i_2} + \cdots + x_{i_k} = c \pmod{2}.\]

Here \(c \in \{0, 1\}\), and the left-hand side is the sum of zero or more *distinct* variables. The equations may have different numbers of variables on the LHS, and different RHS’s.

(b) (2 points.) Suppose \(E_1\) and \(E_2\) are equations as above. Explain how \(E_1 + E_2\) can also be thought of as such an equation. Also, show that an assignment satisfies both \(E_1, E_2\) if and only if it satisfies both \(E_1, E_1 + E_2\).

(c) (2 points.) Given a system of equations as above, show how it can be transformed, in polynomial time, to an equivalent system in which \(x_1\) appears in at most one equation. (Here “equivalent” is in the sense of satisfiability: the transformed system is satisfiable if and only if the original system is satisfiable.)

(d) (1 point.) Suppose we are given a system of equations in which \(x_1\) appears in at most one equation — call it \(E_1\), if it exists. Show that in polynomial time we can transform the system into an equivalent one in which furthermore \(x_2\) appears in at most *two* equations, at most one of which is not \(E_1\). (Hint: Your proof can begin, “Ignoring \(E_1\) (if it exists), again...”)

(e) (3 points.) Write the words “Et cetera.” Now assume we have an (equivalent) system of equations in which \(x_1\) appears in at most one equation (call it \(E_1\)), \(x_2\) appears in at most two equations (call them \(E_1, E_2\)), \(x_3\) appears in at most three equations (call them \(E_1, E_2, E_3\)), ... , \(x_n\) appears in at most \(n\) equations (call them \(E_1, \ldots, E_n\)). (Each \(E_i\) might “not exist”.) Explain why the original system is unsatisfiable if the final system includes the equation “0 = 1” and why the original system is satisfiable otherwise. (For the latter: show how to actually find a satisfying assignment — the key phrase is “back-substitution”. If one of the \(E_i\)’s does not exist, simply add in the equation \(x_i = 0\) and show that nothing is harmed.)