1. **Mother of Two Children.** Write a (nonempty) computer program $P$ whose output is $\langle P \rangle \langle P \rangle$—that is, $P$’s own source code printed twice in succession. You can either describe $P$ in pseudocode, or (for extra credit) implement $P$ in your favorite programming language, and include its output.

2. **Beyond the Halting Problem.** Let $L = \{ \langle M \rangle : \exists x \text{ such that } M(x) \text{ runs forever} \}$.
   (a) Show that $L \leq_T \text{SUPERHALT}$ (where \text{SUPERHALT} is the halting problem for Turing machines $P$ with HALT oracles).
   (b) Show that \text{SUPERHALT} $\leq_T L$. [*Hint: Do there exist positive integers $t, k$ such that $P^{\text{HALT}}$ halts in $t$ steps, and every time $P$ queries the HALT oracle, the machine $Q$ that $P$ asks about either halts in at most $k$ steps or else runs forever?]*

3. **Countable and Uncountable.**
   (a) Recall that a Turing degree is the set of all languages Turing-equivalent to a given language. Show that every Turing degree contains infinitely many languages.
   (b) Show that every Turing degree contains only a countable infinity of languages.
   (c) Show that, if the sets $S_1, S_2, S_3, \ldots$ are each countably infinite, then their union $S_1 \cup S_2 \cup S_3 \cup \ldots$ is countably infinite as well.
   (d) Is the set of all Turing degrees a countable or uncountable set? Why?

4. **Polynomial-Time Reducibility.** Show that if $A \leq_p B$ and $B \leq_p C$, then $A \leq_p C$. If the reduction from $A$ to $B$ blows up the instance sizes by a $p(n)$ factor, and the reduction from $B$ to $C$ blows up the instance sizes by a $q(n)$ factor, then by what factor will the reduction from $A$ to $C$ blow up the instance sizes?

5. **Circuit Complexity.** Define a XOR-circuit to be a circuit with $n$ input bits and $m$ output bits, which is built entirely out of XOR gates with two input wires each. You can assume that the constants 0 and 1 are always available as input bits.
   (a) Show that a function $f : \{0,1\}^n \to \{0,1\}^m$ is computable by a XOR-circuit, if and only if $f$ has the form $f(x_1, \ldots, x_n) = (y_1, \ldots, y_m)$, where each $y_i$ is the sum mod 2 of some subset of the $x_i$’s (and possibly the constant 1).
   (b) Give an example of a function $f : \{0,1\}^2 \to \{0,1\}$ that cannot be computed by a XOR-circuit.
   (c) Show that, if $f : \{0,1\}^n \to \{0,1\}^m$ is computable by a XOR-circuit at all, then it’s computable by a XOR-circuit with at most $nm$ XOR-gates.
(d) Show that there are exactly $2^{m(n+1)}$ functions $f : \{0,1\}^n \rightarrow \{0,1\}^m$ computable by XOR-circuits.

(e) Show that there are at most $(n+T)^{2T+m}$ functions $f : \{0,1\}^n \rightarrow \{0,1\}^m$ computable by XOR-circuits with $T$ XOR-gates.

(f) Combining parts d and e, show that there exists a function $f : \{0,1\}^n \rightarrow \{0,1\}^m$ computable by a XOR-circuit, but only by one with $\Omega \left( mn / \log n \right)$ gates.

6. **Equivalence of Search and Decision.** Show that if $P = NP$, then for any language $L \in NP$, there exists a polynomial-time algorithm that not only decides whether $x \in L$, but if the answer is “yes,” also outputs a proof that $x \in L$. [Hint: Can you reduce the task of finding such a proof to a sequence of yes-or-no NP queries? Keep in mind that there might be multiple valid proofs!]

7. **Complexity Classes.** Recall that $PSPACE = \text{DSPACE} \left( n^{O(1)} \right)$ and $\text{EXP} = \text{DTIME} \left( 2^{n^{O(1)}} \right)$. Show that $PSPACE \subseteq \text{EXP}$.

8. **Time Hierarchy Theorem.** Show that $P^{NP} \neq \text{EXP}^{NP}$. [Hint: Recall the discussion about Turing’s proof of the unsolvability of the halting problem being a relativizing proof.]