No warranty is made or implied regarding whether these are good problems, or whether they are harder, easier, or about the same difficulty level as the problems on the midterm.

0 Write the definition of the following terms:

- alphabet
- string
- \((X)_{\Sigma}\)
- decision problem
- function problem
- search problem
- language
- Boolean circuit
- Boolean formula
- Boolean function
- the PATH problem
- the PALINDROMES problem
- the BOUNDED-ACCEPTANCE\(_{f(n)}\) problem
- the \(k\)COL problem (for \(k \geq 2\))
- the LCS (longest common subsequence) problem
- the CLIQUE and \(k\)-CLIQUE problems
- the HAMILTONIAN-PATH problem
- CIRCUIT-SAT, (FORMULA-)SAT, CNF-SAT, \(k\)SAT, \(k\)kSAT, and NAE\(k\)SAT problems
- the CIRCUIT-EVAL problem
- CNF formula, DNF formula, literal
- Church–Turing Thesis
- Extended Church–Turing Thesis
- Turing Machine
- transition function
- configuration
- computation trace of a Turing Machine
- decider
- Turing Machine \(M\) decides language \(L\)
• Turing Machine $M$ runs in time $f(n)$
• $f(n)$ is $O(g(n))$
• multitape Turing Machine
• nondeterministic pseudocode / Turing Machines (including what it means for such a machine to “accept string $x$” and what its “running time” is)
• universal Turing Machine
• $\text{TIME}(f(n))$
• $\text{NTIME}(f(n))$
• $P$
• $NP$
• $EXP$
• $NEXP$
• $V$ is a verifier for language $L$
• polynomial-time verifier
• Exponential Time Hypothesis (ETH), Strong Exponential Time Hypothesis (SETH)
• polynomial-time mapping reductions ($A \leq^P_m B$)
• $NP$-hard
• $NP$-complete
• search-to-decision reduction
• “padding” (we didn’t give a completely formal definition, but give the concept)

1. Let $L \in NP$. First, show that if $L = \emptyset$ or $L = \{0, 1\}^\ast$ then $L$ is not $NP$-hard. Otherwise, show that $P = NP$ implies $L$ is $NP$-complete.

2. A “two-dimensional Turing Machine” is one where the tape — rather than being a one-dimensional bi-infinite grid with cells indexed by $\mathbb{Z}$ — is a two-dimensional bi-infinite grid with cells indexed by $\mathbb{Z} \times \mathbb{Z}$. Assume it allows head movements of North, South, East, and West. Write explicitly what a transition function would look like. Sketch an appropriate definition of “configuration” and an appropriate definition of “NextConfig” (i.e., the function used in defining computation trace). Sketch a proof that a two-dimensional Turing Machine running in time $T(n)$ can be simulated by a one-dimensional Turing Machine running in time $\text{poly}(T(n))$.

3. Suppose $L \in NP$. Show that $L^\ast \in NP$.

4. Write pseudocode for checking if an input number (written in binary) is a perfect square. Assuming two $n$-bit integers can be multiplied in time $M(n)$, analyze the running time of your algorithm as a function of $M(n)$. Can you get a faster running time if you allow your pseudocode to be nondeterministic?

5. Complete the proof (begun in Lecture 12) that INDEPENDENT-SET is $NP$-complete.

6. Write careful proofs/disproofs of each of the following statements: “$\leq^P_m$ is reflexive”, “$\leq^P_m$ is symmetric”, “$\leq^P_m$ is transitive”. (Look it up on Wikipedia if you forget what those terms about relations mean.)
7. Show that if \( f(n) \) and \( g(n) \) are time-constructible, then so is \( f(n) + g(n) \).

8. Analyze the running time and correctness of the following 3SAT algorithm (which has the flavor of “search-to-decision”) due to Monien and Speckenmeyer. Given a 3SAT instance \( \phi \), if all \( \phi \)’s clauses have width at most 2 then use the polynomial-time algorithm for 2SAT to decide it. Otherwise, pick any clause, say \((\ell_i \lor \ell_j \lor \ell_k)\), and recursively decide \( \phi_{\ell_i = \text{T}}, \phi_{\ell_j = \text{T}}, \phi_{\ell_k = \text{T}} \), and \( \phi_{\ell_i = \text{F}, \ell_j = \text{F}, \ell_k = \text{T}} \), accepting if at least one recursive call accepts. (Here the \( \ell \)’s are “literals” — either a variable or its negation — and the things that look like \( \phi_{\ell=...} \) are sub-3CNFs you get by plugging in values for literals and simplifying.) You may like to prove/use the fact that the recurrence \( T(m) = T(m-1) + T(m-2) + T(m-3) \), with \( T(\text{const.}) = \text{const.} \) solves to \( T(m) = O(c^n) \), where \( c \) is the real solution of \( c^3 - c^2 - c - 1 = 0 \).

9. Write down the Time Hierarchy Theorem, but replace every instance of \( \text{TIME}(\cdot) \) with \( \text{NTIME}(\cdot) \). Now go through the proof of the theorem — does the proof still work? (Remark: depending on time, we may prove the Nondeterministic Time Hierarchy Theorem in this course.)

10. Define \( \text{EEXP} = \bigcup_{k \in \mathbb{N}} \text{TIME}(2^{2^{nk}}) \), and define \( \text{NEEXP} \) to be the nondeterministic analogue. Prove that \( \text{EXP} = \text{NEXP} \) implies \( \text{EEXP} = \text{NEEXP} \).