Some Midterm Practice Problems

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
- $\langle X \rangle_\Sigma$
- 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 kCOL 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, kSAT, EkSAT, and NAEkSAT 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
- $\mathsf{TIME}(f(n))$
- $\mathsf{NTIME}(f(n))$
- 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 ($A \leq_m^P B$)
- NP-hard
- NP-complete
- search-to-decision reduction
- "padding" (we didn't give a completely formal definition, but give the concept)

1. Let $L \in \mathsf{NP}$. First, show that if $L = \emptyset$ or $L = \{0,1\}^*$ then L is not NP-hard. Otherwise, show that $\mathsf{P} = \mathsf{NP}$ 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 poly($T(n)$).

3. Suppose $L \in \mathsf{NP}$. Show that $L^* \in \mathsf{NP}$.

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: "$\leq_m^P$ is reflexive", "$\leq_m^P$ is symmetric", "$\leq_m^P$ 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 ϕ, if all ϕ's clauses have width at most 2 then use the polynomial-time algorithm for 2SAT to decide it. Otherwise, pick any clause, say $(\ell_i \vee \ell_j \vee \ell_k)$, and recursively decide $\phi_{\ell_i=\mathsf{T}}$, $\phi_{\ell_i=\mathsf{F},\ell_j=\mathsf{T}}$, and $\phi_{\ell_i=\mathsf{F},\ell_j=\mathsf{F},\ell_k=\mathsf{T}}$, accepting iff at least one recursive call accepts. (Here the ℓ's are "literals" — either a variable or its negation — and the things that look like $\phi_{\ell=\dots}$ 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 $T(const.) = const.$ 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 $\mathsf{TIME}(\cdot)$ with $\mathsf{NTIME}(\cdot)$. 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 $\mathsf{EEXP} = \cup_{k \in \mathbb{N}} \mathsf{TIME}(2^{2^{n^k}})$, and define NEEXP to be the nondeterministic analogue. Prove that $\mathsf{EXP} = \mathsf{NEXP}$ implies $\mathsf{EEXP} = \mathsf{NEEXP}$.