• You must *type* your solutions. *LATEX*, Microsoft Word, and plain ascii are all acceptable. Submit your solutions *via email* to cs221-hw@seas.harvard.edu. If you use *LATEX*, please submit both the compiled file (.pdf) and the source (.tex). Please name your files PS0-yourlastname.*.

• Strive for clarity and conciseness in your solutions, emphasizing the main ideas over low-level details. Do not despair if you cannot solve all the problems! Difficult problems are included to stimulate your thinking and for your enjoyment, not to overwork you. *’ed problems are extra credit.

Any students who have not completed CS121 or equivalent with a grade of B+ or higher are required to complete this problem set (on time, with no late days). Other students are encouraged to solve the problems for review and submit solutions for feedback.

**Problem 1. (NP-completeness)** In the BOUNDED HALTING problem, we are given a pair $(M, t)$ where $M$ is a Turing machine and $t$ is an integer and have to decide whether there exists an input (of length at most $t$) on which $M$ halts within $t$ steps.

Show that BOUNDED HALTING is NP-complete.

**Problem 2. (coNP)** Let coNP = \{L : \overline{L} \in NP\}, the class of languages whose complement is in NP.

1. Show that a language $L$ is complete for NP iff $\overline{L}$ is complete for coNP. (Here completeness is with respect to poly-time mapping reductions, aka Karp reductions.)

2. Show that if NP \(\neq\) coNP, then P \(\neq\) NP.

3. Let Tautology = \{\(\phi \) : \(\phi\) a boolean formula s.t. \(\forall a, \phi(a) = 1\}\}. Show that Tautology is coNP-complete.

**Problem 3. (Why Languages?)**

1. Given a function \(f : \Sigma^* \rightarrow \Sigma^*\) such that \(|f(x)| \leq \text{poly}(|x|)\) for all $x$, show that there is a language $L$ such that $f$ can be computed in poly-time given a black box (i.e. an “oracle”) for deciding $L$, and $L$ can be decided in poly-time given a black box (i.e. an “oracle”) for computing $f$. That is, $f$ and $L$ are equivalent under Cook reductions.
2. A search problem is a mapping \( S \) from strings (“instances”) to sets of strings (“valid solutions”). An algorithm \( M \) solves a search problem \( S \) if for every input \( x \) such that \( S(x) \neq \emptyset \), \( M(x) \) outputs some solution in \( S(x) \). An NP search problem is a search problem \( S \) such that there exists a polynomial \( p \) and a polynomial-time algorithm \( V \) such that for every \( x, y \):

- \( y \in S(x) \Rightarrow |y| \leq p(|x|) \) and
- \( y \in S(x) \iff V \) accepts \((x, y)\)

It is widely believed that there is no polynomial-time algorithm for integer factorization. Under this assumption and also using the fact that PRIMES is in \( \text{P} \), exhibit two NP search problems \( S \) and \( T \) such that the corresponding languages, \( \{ x : S(x) \neq \emptyset \} \) and \( \{ x : T(x) \neq \emptyset \} \), are identical yet \( S \) is solvable in polynomial time and \( T \) is not.

3. An NP optimization problem is given by a polynomial-time computable objective function \( \text{Obj} : \Sigma^* \times \Sigma^* \rightarrow \mathbb{Q}^{\geq 0} \), where \( \mathbb{Q}^{\geq 0} \) is the set of nonnegative rational numbers and \( \text{Obj}(x, y) = +\infty \) if \( |y| > p(|x|) \) for some polynomial \( p \). The problem is: given an input \( x \), find \( y \) minimizing \( \text{Obj}(x, y) \). An example is the problem of finding the shortest tour in an instance of the Travelling Salesman Problem.

Prove that the following are equivalent:

- \( \text{P} = \text{NP} \)
- Every NP search problem can be solved in polynomial time.
- Every NP optimization problem can be solved in polynomial time.