CS 221: Computational Complexity	**Prof. Salil Vadhan**
Problem Set 0	
Assigned: Tue. Jan. 27, 2014	Due: Fri. Feb. 7, 2014 (5 PM sharp)

- 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 $(M, 1^t)$ 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 $\mathbf{coNP} = \{L : \bar{L} \in \mathbf{NP}\}$, the class of languages whose complement is in **NP**.

1. Show that a language L is complete for **NP** iff $\bar{L}$ is complete for **coNP**. (Here completeness is with respect to poly-time mapping reductions, aka Karp reductions.)
2. Show that if $\mathbf{NP} \neq \mathbf{coNP}$, then $\mathbf{P} \neq \mathbf{NP}$.
3. Let Tautology=$\{\phi$: ϕ a boolean formula s.t. $\forall a, \phi(a) = 1\}$. Show that Tautology is **coNP**-complete.

Problem 3. (Why Languages?)

1. Given a function $f : \Sigma^* \to \Sigma^*$ 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 commputing 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 $\langle x, y \rangle$

   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 **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* $\text{Obj} : \Sigma^* \times \Sigma^* \to \mathbb{Q}^{\geq 0}$, 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:

   - **P** = **NP**
   - Every **NP** search problem can be solved in polynomial time.
   - Every **NP** optimization problem can be solved in polynomial time.