CS 142 & Math 167
Fall 2008
Homework 3
Due Monday, February 7

1. **[15 Points] Logic.** In the next several weeks, we are going to be looking at problems which involve Boolean functions. First, very briefly sketch the proof that every Boolean function can be expressed as a circuit using AND, OR, and NOT gates. Then prove that AND and OR gates alone do not suffice to express all Boolean functions. *A good approach is to show a specific Boolean function which cannot be expressed using AND and OR gates alone. Your proof must be rigorous. Saying, "clearly there's no way to express this function with AND and OR gates" is not a valid proof.*

2. **[15 Points] Funky Properties of P and NP.**

   (a) Prove that if every problem in NP is NP-complete then P=NP.

   (b) Professor I. Lai of the Pasadena Insitute of Technology has stated the following theorem: "Let L_1 and L_2 be any two languages in NP such that $L_1 \leq_p L_2$. Then it is necessarily the case that $L_2 \leq_p L_1$." Prove that if Lai's Theorem is true then P = NP.

   (c) Prove that if P = NP then *all* languages other than $\emptyset$ and Σ^* are NP-hard. A language L is said to be *NP-hard* if every language in NP is polynomial-time reducible to L. Thus, the difference between NP-hardness and NP-completeness is that to be NP-hard the language doesn't necessarily need to be in NP.

3. **[15 Points] An Odd Language!** Let

   $L = \{< M, > w, 0^t | \ M$ is nondeterministic TM that accepts string w within t time steps$\}$

   Prove that language L is NP-complete by showing that *every* problem in NP can be poly-time reduced to L. (This is not the usual way of doing an NP-completeness proof; normally we would perform a reduction from a single known NP-complete problem. Nonetheless, here I would like you to perform a proof from every possible language in NP.)

4. **[15 Points] Closure under Kleene Star!** Recall that if L is a language then $L^0 = \{\epsilon\}$, $L^1 = L$, and $L^i = LL^{i-1}$ (the concatenation of L and L^{i-1}). The

Kleene Closure (pronounce "Clean Knee")[1] of L is denoted L^* and is defined to be

$$\bigcup_{i \geq 0} L^i$$

In other words, L^* is the language made up of all strings w where w can be written as the concatenation of 0 or more strings in L.

(a) (Warmup) Show that if L is recursive then L^* is recursive.

(b) (Warmup) Show that if L is recursively enumerable then L^* is recursively enumerable.

(c) (Moderate) Show that if L is in NP then L^* is in NP.

(d) (A bit more challenging) Show that if L is in P then L^* is in P.

5. **[15 Points] Cook's Theorem Revisited!** In class we examined Cook's Theorem that SATISFIABILITY is NP-complete. The proof begins by first observing that for any language L in NP there exists a non-deterministic Turing Machine M and a polynomial $p(n)$ such that on input string w, M runs in time $p(|w|)$ and determines whether or not w is in the language. Then, we showed that for any string w of length n we can build (in polynomial time) an instance of the SATISFIABILITY problem which is satisfiable if and only if M accepts w in time $p(n)$.

   The reduction involved 6 groups of clauses called $G_1, \ldots, G_6$. Looking back at your notes, you will recall that G_5 enforced the constraint that the contents of the tape at time $i+1$ follow from the contents at time i according to the δ function of the Turing Machine M. In class, we only showed the clauses in G_5 which enforce that the symbol under the tape head changes in accordance to the δ function but we did not show the clauses that enforce that all other cells on the tape (those not under the tape head) are unchanged.

[1] From the History of Mathematics Archives at University of St. Andrews, Scotland: "Stephen C. Kleene studied for his first degree at Amherst College. He went on to receive a doctorate from Princeton University in 1934, supervised by Church, for a thesis entitled "A Theory of Positive Integers in Formal Logic." Then Kleene taught at Princeton until he joined the University of Wisconsin at Madison in 1935. He became a full professor at the University of Wisconsin at Madison in 1948 and remained on the staff there until he retired in 1979. Kleene's research was on the theory of algorithms and recursive functions. He developed the field of recursion theory with Church, Godel, Turing and others. He contributed to mathematical Intuitionism which had been founded by Brouwer. His work on recursion theory helped to provide the foundations of theoretical computer science. By providing methods of determining which problems are soluble, Kleene's work led to the study of which functions can be computed."

(a) Using the variable notation that we defined in class ($Q_{i,k}, H_{i,j}$, and $S_{i,j,k}$) give the additional clauses that are needed to enforce that tape cells not under the tape head are unchanged after each time step.

(b) Derive an upper-bound on the number of clauses that you have introduced as a function of $p(n), r$, and g. (Look back at the class notes to remind yourself what these are.) Is the added number of clauses a polynomial in n?

6. **[25 Points] Two NP-completeness Proofs.** We will not spend a great deal of time in this course proving the NP-completeness of various problems. However, this problem will help stretch your NP-completeness muscles and will show the NP-completeness of two problems that we will study further in class. The cool thing about these problems is that they are both *number problems*, not graph problems or logic problems.

   (a) The Subset Sum problem is formulated as follows: Given a multiset S (a set where some items may occur more than once) of positive integers and a target T, does there exist a subset of $S' \subseteq S$ such that the sum of items in S' adds exactly up to T? Prove that this problem is NP-complete via a reduction from either 3-SAT or 3-Coloring.

   Yikes! This problem is all about numbers! How are we going to reduce a logic problem like 3-SAT or a graph problem like 3-Coloring to a number problem!? Here's a hint: Given an instance of 3-SAT, let $x_1, \ldots x_v$ denote the v variables and $C_1, \ldots, C_k$ denote the k clauses. The numbers in our Subset Sum instance will have $v + k$ digits and these digits will be 0's or 1's. Imagine that we have $v = 4$ variables and $k = 2$ clauses. The k least significant digits will correspond to our clauses and the v most significant digits will correspond to our variables. Now, we introduce one $v+k$-digit number for x_i being **true** and another for x_i being **false**. In our example, let clause $C_1 = (x_1 \vee \overline{x_2} \vee x_4)$. Then, as indicated in the picture below, we have a number corresponding to x_1 being **true**. This is the number labeled x_1, T. It has a 1 in the x_1 column and a 0 in every other x_i column. In addition, it has a 1 in the C_1 column because x_1 appears unnegated in clause C_1. Similarly, x_2 appears negated in clause C_1 so notice that the number x_2, F has a 1 in the C_1 column. In addition, this picture suggests that x_1 appears negated in clause C_2 and x_2 (unnegated) does not appear in any clause.

	x_1	x_2	x_3	x_4	C_1	C_2
x_1, T	1	0	0	0	1	0
x_1, F	1	0	0	0	0	1
x_2, T	0	1	0	0	0	0
x_2, F	0	1	0	0	1	0

I have not shown the entire construction! I haven't shown the numbers for all of the variables. Moreover, you may need to introduce some other numbers in your construction of the set S. Finally, I have not specified the value of the target T. Although the numbers constructed here have only 0's and 1's, these numbers probably should not be interpreted in base 2 but rather some higher base that you will need to determine. A very similar idea can be used to do a reduction from 3-COLORING. You'll get 5 bonus points for doing that reduction *instead* of the 3-SAT reduction.

(b) The BIN PACKING Problem is the following: We are given a collection of objects $S = \{a_1, \ldots, a_n\}$, a function $s : S \to \mathbb{Z}_+$ (the set of positive integers) indicating the "size" of each object, a bin capacity C such that C is greater than or equal to the size of the largest object, and a target T. The question is whether or not it is possible to pack all of the objects in S into T or fewer bins, each of capacity C. Prove that BIN PACKING is NP-complete from any problem that we (that's you on homework or me in class) have shown to be NP-complete.

7. **[20 Point OPTIONAL Bonus Problem] Primality Testing is in NP!** Usually, proving that a problem is in NP is pretty easy. Sometimes though, it's actually quite interesting and challenging! This problem explores one such problem.

PRIMES is the language of all prime numbers in binary. Prove that PRIMES is in NP. *Hint:* The only number theory that you need to do this is the following result which you may use without proof: "A number $p > 1$ is prime iff there exists a number $1 < r < p$ such that $r^{p-1} = 1 \bmod p$ and for all prime divisors q of $p - 1$, $r^{\frac{p-1}{q}} \neq 1 \bmod p$.