I hate to admit it, but I have not been able to keep up with the grading of the individual work problems I have been assigning each week. To increase the odds that I eventually get you some feedback on that work, there will be no individual work problem this week. The three problems below should be completed as group work due on Wednesday or Tuesday depending on your group meeting time.

1. Complete problem 7.26 in Sipser:

   Let ϕ be a 3CNF-fomula. An **$\neq$-assignment** to the variable of ϕ is one where each clause contains two literals with unequal truth values. In other words, an $\neq$-assignment satisfies ϕ without assigning three true literals in any clause.

   (a) Show that the negation of any $\neq$-assignment to ϕ is also an $\neq$-assignment.

   (b) Let $\neq SAT$ be the collection of 3CNF-formulas that have $\neq$-assignments. Show that we obtain a polynomial time reduction from $3SAT$ to $\neq SAT$ by replacing each clause c_i

   $$(y_1 \vee y_2 \vee y_3)$$

   with the two clauses

   $$(y_1 \vee y_2 \vee z_i) \text{ and } (\overline{z_i} \vee y_3 \vee v)$$

   where z_i is a new variable for each clause c_i and v is a single additional new variable.

   (c) Conclude that $\neq SAT$ is NP-complete. (Hint: Given parts (a) and (b) this step should be trivial.)

2. In lecture 19, I presented an argument that 3-SAT $\leq_p$ SUBSET-SUM and, in the text (see page 320), Sipser provides a similar argument. The mappings described in both arguments take a boolean formula in 3-conjunctive normal form and map it to an instance of the subset sum problem (a list of positive integers and a target sum). The goal in defining this mapping is to show that SUBSET-SUM is NP-complete using the knowledge that 3SAT is NP-complete.

   While both of these proofs are based on reductions from 3-SAT, one could instead show that SUBSET-SUM is NP-complete by demonstrating a reduction to SUBSET-SUM from any other language that is known to be NP-complete. Consider, in particular, the language $\neq$ SAT defined in problem 7.26 (page 324) of Sipser's text which is the preceding problem on this assignment. Based on the result of this problem, we know that $\neq$ SAT is NP-complete. So, one could instead show that SUBSET-SUM is NP by showing that $\neq$ SAT $\leq_p$ SUBSET-SUM. To do this, you would need to describe a mapping from boolean formulas to subset sum problems.

Wait a miniute!!! You might already have what you need! The mapping I described in class (along with the one Sipser describes in the text) is already a mapping from boolean formulas to subset sum problems. Can you use exactly the same mapping to show that $\neq$ SAT $\leq_p$ SUBSET-SUM?

For this problem, I want you to either

- explain why either the mapping I described in class or the one Sipser described (your choice!) is sufficient to show that $\neq$ SAT $\leq_p$ SUBSET-SUM, or
- explain why either of these mapping (your choice again!) is not sufficient to show that $\neq$ SAT $\leq_p$ SUBSET-SUM, and explain how it could be modified (as little as possible) so that it did show that $\neq$ SAT $\leq_p$ SUBSET-SUM.

Hints: A) You might find the example I worked in class helpful. B)Even though I said "your choice" repeatedly, I suspect using the construction from class will be easier.

3. In the lecture videos, I proved that many of the examples of language in NP that we considered (SAT, 3-SAT, and SUBSET-SUM) were particularly interesting because they were all examples of NP-complete problems. Since most computer scientists believe that P is not equivalent to NP, searching for an example of a problem in NP that is so hard that it cannot be decided in deterministic polynomial time seems like a reasonable goal. Therefore, a lot of focus is placed on the hardest problems in NP - the NP-complete problems. In fact, although I did not included a proof of this fact, 3-dimensional matching (a.k.a. the problem I described in terms of a dinner party whose members hated to order the same item as any of their companions), is also NP-complete. So, in fact, all of the examples of languages in NP we have considered are NP-complete.

   One might worry that the problems in NP that are not NP-complete might feel unappreciated!

   Alternately, one might wonder if there are any examples of languages that belong to NP but are not NP-Complete. That is, is there any L such that $L \in NP$ but L is not NP-complete? This turns out to be another open question because...

   If there exists a non-trivial $L \in NP$ that is not NP-complete then it must be the case that $P \neq NP$. That is, if we could prove that some non-trivial language $L \in NP$ was not NP-complete it would be another way to prove that $P \neq NP$.

   Prove that this statement is true.

   The "non-trivial" requirement addresses the same issue it addressed in the statement of Rice's Theorem. L is "non-trivial" as long as it is any language other than $\emptyset$ and Σ^*.