CS 254: Computability and Complexity	**Anonymous submission**
Problem Set #10	*November 19, 2019*

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

   Proof: Each clause contains at least one literal assigned true and one literal assigned false. Therefore flipping them would still result in one literal being assigned true and one being assigned false.

   (b) Let $\neq$ SAT be the collection of 3cnf-formulas that have an $\neq$-assignment show we can polynomial time reduce 3SAT to $\neq$ SAT by replacing clause c_i $(\mathrm{y}_1 \vee \mathrm{y}_2 \vee \mathrm{y}_3)$ with two clauses $(\mathrm{y}_1 \vee \mathrm{y}_2 \vee \mathrm{z}_i)$ and $(\bar{z_i} \vee \mathrm{y}_3 \vee \mathrm{b})$

   Proof: For the reduction to be correct we need it to be that $\emptyset$ is mapped to $\emptyset_2$ then $\emptyset$ is satisfiable iff $\emptyset_2$ has an $\neq$-assignment. We can obtain an $\neq$-assignment if $\emptyset$ satisfiable and we extend the assignment to $\emptyset$ so that z_1 is 1 if both literals y_1 and y_2 in clause c_i are 0 or z_1 is 0 if both literals y_1 and y_2 in clause c_i are 1. We simply assign 0 to b.

   Going the other way if $\emptyset_2$ has an $\neq$-assignment there is a satisfying assignment to $\emptyset$ because we know b is assigned to 0 and therefore y_1 y_2 y_3 cannot all be assigned to 0. Therefore this will have a satisfying assignment.

   (c) Conclude that $\neq$ is NP-complete

   Proof: Since we have shown that 3SAT is reducible to $\neq$SAT all other languages must also be mapping reducible to $\neq$SAT. Since we have show that this reduction is possible in polynomial time it must also be the case that $\neq$SAT $\in$ NP. Therefore $\neq$SAT is NP-complete