Problem Reduction Terminology. For two problems X and Y we will often ask,

> "Show that a polynomial time algorithm solving X implies a polynomial time algorithm solving Y."

If we can do this, then we write "$Y \leq X$."

1. Write down in your own words what "$Y \leq X$" means here.
2. Write down the contrapositive of what you wrote in (1).
3. For some problem X, suppose we can show that X has a poly-time algorithm.
   - (True/False) $\mathsf{3SAT} \leq X$
   - (True/False) $X \leq \mathsf{3SAT}$
   - (True/False) $\mathsf{2SAT} \leq X$
   - (True/False) $X \leq \mathsf{2SAT}$
4. Write down the CNF formula for "$x = y$"

Problem $\mathsf{4SAT} \leq \mathsf{3SAT}$**.** Show that a polynomial time algorithm for 3SAT implies a polynomial time algorithm for 4SAT.

How would you format your answer on an exam?

Problem . Let the HALFSAT problem be defined as follows.

HALFSAT: Given a formula in CNF form with $2n$ variables and no negations, determine whether the formula can be satisfied by setting at most n variables to TRUE.

Here is an example of a HALFSAT instance:

The formula

$$X = (x_1 \vee x_5) \wedge (x_2 \vee x_3 \vee x_6) \wedge (x_1 \vee x_4) \wedge (x_4) \wedge (x_2 \vee x_5)$$

can be satisfied by setting x_2, x_4, x_5 to TRUE (and x_1, x_3, x_6 to FALSE).
Thus, X is an instance of HALFSAT.

Prove that a poly time algorithm for HALFSAT would imply a poly time algorithm for 3SAT (That is, 3SAT $\leq$ HALFSAT).

*Solution:*Suppose $\mathcal{A}$ is a polynomial time algorithm for HALFSAT. Then, construct a polynomial time algorithm $\mathcal{B}$ for 3SAT as

$\mathcal{B}(X : \text{3SAT})$

//Assume there are n variables, m clauses.

1. Let $Y = X$.
2. Generate n fresh variables $y_1, \ldots, y_n$.
3. For each clause $C = (c_1 \vee c_2 \vee c_3)$ in Y :
   (a) If any c_j is a negation ($c_j = \bar{x}_i$ for some $i \in [n]$):
   Replace $c_j = \bar{x}_i$ with (un-negated) variable y_i in Y.
4. For $i = 1, \ldots, n$:
   (a) Set $Y = Y \wedge (y_i \vee x_i)$
5. Output $\mathcal{A}(Y)$.

Reduction Runtime. We iterate over each clause $C = (c_1 \vee c_2 \vee c_3) \in Y$ and check if any c_j is a negation. If it is, we do $O(1)$ time modifications to the original formula X, per negated c_j, taking $O(nm)$ time. We then iterate over each variable, adding clause $(y_j \vee x_j)$ to Y. All together, the resulting formula Y can be computed in polynomial time. Lastly, since the resulting formula Y is at most a constant factor larger than the original formula X, running $\mathcal{A}(Y)$ takes at most polynomial time.

Correctness. Let Make be the algorithm consisting of steps $1-4$ of algorithm $\mathcal{B}$. We need to argue $X \in$ 3SAT iff Make$(X) \in$ HALFSAT.

We need to argue that X has a satisfying 3SAT solution iff $Y =$ Make(X) has a satisfying HALFSAT solution. Justin says: What are we trying to show here? The forward direction,

if X is 3SAT satisfiable, then our transformation Make(X) is HALFSAT satisfiable, is more clear. We are mapping "yes" 3SAT instances to "yes" HALFSAT instances. The backward direction, that for some X instance, Make(X) being HALFSAT satisfiable implies that X is 3SAT satisfiable, may be a bit more mystifying. It helps to think of the contrapositive, that if X is *not* 3SAT *satisfiable,* then Make(X) is also not 3SAT satisfiable. Viewed in this way, we are showing that "no" 3SAT instances are mapping to "no" HALFSAT instance! Proving both directions means that all instances, whether they are "yes" or "no," map correctly, meaning that we can immediately use helper function $\mathcal{A}$ for HALFSAT.

($\rightarrow$) Suppose X is 3SAT satisfiable. Then, for Y = Make(X) to be HALFSAT satisfiable, we need to show (1) Y is a CNF with no negations, (2) Y has a satisfying assignment, and (3) that satisfying assignment has at most half of the variables set to be true.

(1) is true since we replace each negated variable $\bar{x}_i$ with a new un-negated variable y_j in step 3, and all new variable added in step 4 are un-negated.

To show (2), we first observe that any new variable y_i that does not replace a negated variable $\bar{x}_i$ in the original CNF X only shows up in the new formula Y in clause $(y_i \vee x_i)$. Hence, these particular y_i can be set to `False` if x_i = `True` and vice versa. Now, suppose that variable y_i replaced a negated variable $\bar{x}_i$. Indeed, if $\bar{x}_i$ = `True` $\iff$ x_i = `False`, then by the added clause

$$(x_i \vee y_i) = (\texttt{False} \vee y_i) = y_i,$$

we set y_i = `True`. On the other hand, if $\bar{x}_i$ = `False` $\iff$ x_i = `True`, by the same added clause,

$$(x_i \vee y_i) = (\texttt{True} \vee y_i) = \texttt{True}$$

y_i can be set to `False`.

Lastly, (3) is true by observing above, that any satisfying assignment where x_i = `True`, we can set y_i = `False` and vice versa. Then, if there are n_T such x_i variables set to `True`, then there are $(n - n_T)$ such y_i variables set to `True`. In total, there are $(n_T + (n - n_T)) = n$ variables set to true out of $2n$ total variables. Thus, Make(X) is HALFSAT satisfiable.

($\leftarrow$) Suppose Make(X) is HALFSAT satisfiable. We want to show that X is 3SAT satisfiable.

We make the same observations as in the forward direction, that any new variable y_i that does not replace a negated variable $\bar{x}_i$ in the original CNF X only shows up in the new formula Y in clause $(y_i \vee x_i)$. Hence, we can set these x_i = `True`, satisfying clauses in the original X which have a non-negated x_i.

Now, for the clauses where negated $\bar{x}_i$ is replaced with y_i, we claim that y_i = `True` iff x_i = `False`. If this is true, then the original must also be satisfiable (If y_i = `False`, then the original clause including $\bar{x}_i$ must be true by some other variable. If y_i = `True` then the original clause is satisfied by $\bar{x}_i$). Thus, we have derived a satisfying assignment of the original x_i variables and showed that X is 3SAT satisfiable.

Indeed, we show that $y_i = \texttt{True}$ iff $x_i = \texttt{False}$. To see this, we first make the observation that $x_i \neq y_i$. That is, $x_i = \texttt{True}$ iff $y_i = \texttt{False}$. Assume for the sake of contradiction that either (1) $x_i = y_i = \texttt{False}$ or (2) $x_i = y_i = \texttt{True}$. The first case cannot happen since we have clauses $(x_i \vee y_i)$. So, suppose the second case is true. Then, since $\mathsf{Make}(X)$ is $\mathsf{HALFSAT}$ satisfiable, at most half of variables $x_1, \ldots, x_n, y_1, \ldots, y_n$ is true. By our assumption that $x_i = y_i = \texttt{True}$, there exist a pair $x_j = y_j = \texttt{False}$ by pigeonhole principle. Again, since we have clauses $(x_j \vee y_j)$, we reach a contradiction.

□