1. Let A be an NP-complete language and B be in P. Prove that if $A \cap B = \phi$, then $A \cup B$ is NP-complete. What can you say about the complexity of $A \cup B$ if A and B are not known to be disjoint?

2. Show that a DNF formula can be converted in polynomial time to a CNF formula, with possibly more number of variables , preserving satisfiability. Show that if $\mathsf{P} \neq \mathsf{NP}$, there cannot be a polynomial time algorithm that switches a CNF formula to an DNF formula preserving satisfiability.

Solution:
Part a
The formula ϕ in DNF will be as $P_1 \vee P_2 \vee P_3..$. Every P_i will be of the form $x_1 \wedge x_2 \wedge x_3..$ Now , take every clause of the form $P_i \vee P_j$.
If either P_i or P_j is a variable, then directly distribute the expression over the $\vee$ to get an expression in CNF.
Else, introduce a new symbol y_i and make the clauses $(y_i \vee P_i) \wedge (\overline{y_i} \vee P_j)$.
This will preserve the satisfiability of the $(P_i \vee P_j)$ expression as follows:

- If both P_i and P_j is false, putting any value to y_i will preserve satisfiability.
- Similarly if both P_i and P_j is true, then putting any value to y_i will preserve satisfiability.
- Suppose one of them is true and other is false. WLG let P_i be true. Then putting y_i as true will preserve satisfiability.

Now, since y_i is a variable, the subclauses can now be distributed in the same way as above. This will take atmost length of P_i time. And total time taken to convert will be $|\text{number of clauses}| * |\text{Time taken to expand each clause}|$. And this will take polynomial time in length of the expression.

Part b
We can show this by proving the contrapositive of the statement. i.e. If there exists a polynomial time conversion from CNF to DNF, then $\mathsf{P} = \mathsf{NP}$.

Consider, any formula in the CNF. Now, use the algorithm and convert it into a DNF. The formula will be of the form, $P_1 \vee P_2..$ and having the same satisfiability as the CNF. Now, scan through each of the P_i and see if any clause has both the variables x_i and $\overline{x_i}$. If not, assign to all variables in that clause of form x_j a true

and $\overline{x_j}$ a false value and for the variables not in that clause any value.Then report satisfiable. Else, if there is no such clause, report unsatisfiable.
Hence, SAT problem can be solved in polynomial time, since the scanning takes atmost the length of the clause. And since, SAT is NP-Complete, Every problem in NP can be decided in polynomial time. And hence $\mathsf{P} = \mathsf{NP}$.

3. Give a polynomial time algorithm for 2-SAT problem that we stated in class. Given a CNF formula ϕ, where each clause has atmost two literals, test satisfiability. Is 2-SAT in NL? Argue.

4. Consider the complexity class DP (D stands for *difference*) as the set of problems L such that $L = A \cap B$ where A and B are two languages in NP and coNP respectively. Argue that $\mathsf{DP} \subseteq \mathsf{P}^{\mathsf{NP}}$. Argue that EXACT-CLIQUE is DP-complete.

5. Cook-Levin Theorem is proved by reducing any $L \in \mathsf{NP}$ to SAT. Is there any relation between the number of accepting paths of the non-deterministic machine (deciding L) and the number of satisfying assignments of the formula produced by the reduction? Check the same for the NP-completeness proof for INDEPENDENT SET PROBLEM that we did in class.