1. We say that a propositional formula $\phi$ is in $2\text{CNF}$ if it is a conjunction of clauses, each of which contains exactly 2 literals. The point of this problem is to show that the satisfiability problem for formulas in $2\text{CNF}$ can be solved by a polynomial time algorithm.

First note that any clause with 2 literals can be written as an implication in exactly two ways. For instance $(p \lor \neg q)$ is equivalent to $(q \rightarrow p)$ and $(\neg p \rightarrow \neg q)$, and $(p \lor q)$ is equivalent to $(\neg p \rightarrow q)$ and $(\neg q \rightarrow p)$.

For any formula $\phi$, define the directed graph $G_\phi$ to be the graph whose set of vertices is the set of all literals that occur in $\phi$, and in which there is an edge from literal $x$ to literal $y$ if, and only if, the implication $(x \rightarrow y)$ is equivalent to one of the clauses in $\phi$.

(a) If $\phi$ has $n$ variables and $m$ clauses, give an upper bound on the number of vertices and edges in $G_\phi$.

(b) Show that $\phi$ is unsatisfiable if, and only if, there is a literal $x$ such that there is a path in $G_\phi$ from $x$ to $\neg x$ and a path from $\neg x$ to $x$.

(c) Give an algorithm for verifying that a graph $G_\phi$ satisfies the property stated in (b) above. What is the complexity of your algorithm?

(d) From (c) deduce that there is a polynomial time algorithm for testing whether or not a $2\text{CNF}$ propositional formula is satisfiable.

(e) Why does this idea not work if we have 3 literals per clause?

2. A clause (i.e. a disjunction of literals) is called a Horn clause, if it contains at most one positive literal. Such a clause can be written as an implication: $(x \lor (\neg y) \lor (\neg w) \lor (\neg z))$ is equivalent to $((y \land w \land z) \rightarrow x)$. HORN$\text{SAT}$ is the problem of deciding whether a given Boolean expression that is a conjunction of Horn clauses is satisfiable.

(a) Show that there is a polynomial time algorithm for solving HORN$\text{SAT}$. (Hint: if a variable is the only literal in a clause, it must be set to true; if all the negative variables in a clause have been set to true, then the positive one must also be set to true. Continue this procedure until a contradiction is reached or a satisfying truth assignment is found).

(b) In the proof of the NP-completeness of SAT it was shown how to construct, for every nondeterministic machine $M$, integer $k$ and string $x$ a Boolean expression $\phi$ which is satisfiable if, and only if, $M$ accepts
x within \( n^k \) steps. Show that, if \( M \) is deterministic, than \( \phi \) can be chosen to be a conjunction of Horn clauses.

(c) Conclude from (b) that the problem \textsc{Hornsat} is P-complete under L-reductions.

3. In general \textit{k-colourability} is the problem of deciding, given a graph \( G = (V, E) \), whether there is a colouring \( \chi : V \rightarrow \{1, \ldots, k\} \) of the vertices such that if \((u, v) \in E\), then \( \chi(u) \neq \chi(v) \). That is, adjacent vertices do not have the same colour.

(a) Show that there is a polynomial time algorithm for solving 2-colourability.

(b) Show that, for each \( k \), \textit{k-colourability} is reducible to \( k+1 \)-colourability. What can you conclude from this about the complexity of 4-colourability?

4. \textsc{Polylogspace} is the complexity class

\[ \bigcup_k \text{SPACE}((\log n)^k). \]

(a) Show that, for any \( k \), if \( A \in \text{SPACE}((\log n)^k) \) and \( B \leq_L A \), then \( B \in \text{SPACE}((\log n)^k) \).

(b) Show that there are no \textsc{Polylogspace}-complete problems with respect to \( \leq_L \). (Hint: use (a) and the space hierarchy theorem).

(c) Which of the following might be true: \( P \subseteq \text{Polylogspace} \), \( P \supseteq \text{Polylogspace} \), \( P = \text{Polylogspace} \)?