1. In the lecture, a proof was sketched showing a $\Omega(n \log n)$ lower bound on the complexity of the sorting problem. It was also stated that a similar analysis could be used to establish the same bound for the Travelling Salesman Problem. Give a detailed sketch of such an argument. Can you think of a way to improve the lower bound?

2. Consider the language **Unary-Prime** in the one letter alphabet \{a\} defined by

   $$\text{Unary-Prime} = \{a^n \mid n \text{ is prime}\}.$$ 

   Show that this language is in **P**.

3. We say that a propositional formula $\phi$ is in **2CNF** 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 **2CNF** 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 **2CNF** propositional formula is satisfiable.

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

4. 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 $((y \land w \land z) \rightarrow x))$. **HORNSAT** 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 HORNSAT. 
(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 HORNSAT is P-complete under 
L-reductions.

5. In general $k$-colourability is the problem of deciding, given a graph 
$G = (V, E)$, whether there is a colouring $\chi : V \to \{1, \ldots, k\}$ 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 $k$, $k$-colourability is reducible to $k + 1$-colourability. 
What can you conclude from this about the complexity of 4-colourability?