♢ **Best before:** October 31.

1. Consider the language *LPATH* as defined in problem 7.21 on page 324 of Sipser.
   We define the related optimization problem *FIND-LPATH* as follows.

   FIND-LPATH
   Input: A graph G and two vertices a, b.
   Question: Find a longest (simple) path in G from a to b. Return "None" if no such path exists.

   Prove that *FIND-LPATH* can be solved in polynomial time if and only if $LPATH \in$ P.

   Hint: To find a longest path, you should first find the length of a longest path.

2. (a) Prove that NP is closed under union, intersection, concatenation and (Kleene) star.
   (b) Discuss the closure of NP under complement and homomorphisms.
   (c) Define some new language operations and prove whether NP is closed under them.
   (d) Is NP closed under "inverses" of the above operations?
   For example, here is a question for an inverse of union.
   Suppose $L_1 \in$ NP and $L_1 \cup L_2 \in$ NP. Does it follow that $L_2 \in$ NP?
   (e) Is co-NP closed under any of the above language operations?

3. Recall that a *tautology* is a boolean formula that is satisfied by every truth assignment.
   Let $TAUT = \{\langle\phi\rangle \mid \phi \text{ is a tautology}\}$. Prove that $TAUT \in$ co-NP.

4. Prove that if NP $\neq$ co-NP, then NPC $\cap$ co-NP $= \emptyset$ (NPC is the class of all NP-complete problems).

5. Prove that if P $=$ NP, then every language $A \in$ P, except $A = \emptyset$ and $A = \Sigma^*$, is NP-complete.
   Prove also that $\emptyset$ and Σ^* are not NP-complete, independent of whether P $=$ NP.

6. Given 2 CNF formulas F_1 and F_2, we say that F_1 is a *subformula* of F_2 if F_1 is equal to F_2 with zero or more clauses removed.

   Consider this maximization version of the boolean satisfiability problem, where we try to satisfy as many clauses as possible with a single truth assignment.

   FIND-MAX-SAT
   Input: A CNF formula F.
   Question: Find a satisfiable subformula of F with the largest number of clauses.

   Find a language in NP and prove that it is in P if and only if there is a polytime algorithm for *FIND-MAX-SAT*.

   You must show that your language is in NP.

7. Consider problem 7.29 on page 325 of Sipser (about graph colouring).
   For each $k \in \mathbb{N}$, we define the decision problem k-*COL* as follows.

   k-*COL*
   Input: A graph G.
   Question: Can the nodes of G be coloured with no more than k colours so that no adjacent nodes have the same colour?[1]

   Prove that both 1-*COL* and 2-*COL* are in P.

[1] Graphs that can be coloured in this way are said to be k-*colourable*. A way to colour such a graph is called a k-*colouring*.