Worth: 12% **Due:** By 9:59pm on Tuesday 25 November

Remember to write the *full name* and *student number* of *every group member* prominently on your submission.

---

*Please read and understand the policy on Collaboration given on the Course Information Sheet. Then, to protect yourself, list on the front of your submission **every** source of information you used to complete this homework (other than your own lecture and tutorial notes, and materials available directly on the course webpage). For example, indicate clearly the **name** of every student from another group with whom you had discussions, the **title** of every additional textbook you consulted, the **source** of every additional web document you used, etc.*

*For each question, please write up detailed answers carefully. Make sure that you use notation and terminology correctly, and that you explain and justify what you are doing. Marks **will** be deducted for incorrect or ambiguous use of notation and terminology, and for making incorrect, unjustified, ambiguous, or vague claims in your solutions.*

---

1. The *Maximum Flow with Losses* problem is similar to the maximum flow problem, except that each vertex $u \in V - \{s,t\}$ has a real number *loss coefficient* $\varepsilon_u \in [0,1]$ such that the total flow out of u is equal to $(1-\varepsilon_u)$ times the total flow into u. As before, we are looking for an assignment of flow values to every edge that maximizes the total flow out of s.

   (a) Show how to solve the maximum flow with losses problem using linear programming. Give a detailed description of your linear program and justify clearly and carefully that it solves the problem.

   (b) Suppose that you have found a library that implements an efficient algorithm to solve linear programs. The main library function expects to be given only three inputs: a $[1 \times n]$ coefficient vector c, an $[m \times n]$ constraint matrix A, and an $[m \times 1]$ bound vector b (all numbers are real numbers). The function returns an $[n \times 1]$ solution vector x that **minimizes** the objective function $c \cdot x$ subject to the constraints that $Ax \geqslant b$ (all entries of x are also real numbers).

   Explain how you can use this library to solve your linear program from part (a). This will involve modifying your linear program to fit the input requirements of the main library function: explain clearly each change that you need to make and why; then explain how to take the value returned by the library function and turn it back into a solution to the original linear program.

2. Show that for every decision problem $D \in NP$, there is some polynomial $p(n)$ and some algorithm A (both of which depend on D) such that A solves D in worst-case time $\mathcal{O}\left(2^{p(n)}\right)$.

   HINT: This involves mostly "unrolling" the definitions, so write up your answer carefully. In your answer, you must use the formal definition of *NP* in terms of polynomial-time *verifiers*—note that the intuition given by "generate-and-verify" algorithms, as presented in class, is **not** a formal definition of *NP*.

3. A web server has a number of simultaneous "requests" to reply to. The server has to send its replies in "packets," each one of which has a fixed positive integer size limit L. Each packet can contain more than one reply (so multiple requests can be replied to in a single packet), but each individual reply has its own positive integer size s_i. We would like to use as few packets as possible to send all of the replies.

This problem can be formulated as a decision problem FewPackets ("FP" for short), as follows:

Input: Positive integer *packet size limit* L, positive integers *reply sizes* $s_1, \ldots, s_n$, positive integer *number of packets* k—all integers represented in binary.

Output: Is there some partition of $\{1, \ldots, n\}$ into packets $P_1, \ldots, P_k$, where each packet has size at most L—formally: $\exists P_1, \ldots, P_k, P_1 \cup \cdots \cup P_k = \{1, \ldots, n\} \wedge (\forall i, j, P_i \cap P_j = \emptyset) \wedge \forall i, \sum_{j \in P_i} s_j \leqslant L$?

For example, $(8, \{2, 5, 4, 5\}, 3) \in$ FP because we can use packets $P_1 = \{1, 3\}$ (with size $s_1 + s_3 = 2+4 \leqslant 8$), $P_2 = \{2\}$ (with size $s_2 = 5 \leqslant 8$), and $P_3 = \{4\}$ (with size $s_4 = 5 \leqslant 8$). However, $(8, \{2, 5, 4, 5\}, 2) \notin$ FP because there is no way to partition every reply into only 2 packets, each one of size $\leqslant 8$—even though the total size of all replies is only 16.

Write a *detailed* proof that FP is *NP*-complete.

4. Give a precise definition for a MinPackets *optimization* problem related to language FewPackets from the previous question—state *clearly* the input and output for your problem, including the quantity that must be optimized in the output.

   Then, write a detailed argument that MinPackets is polytime self-reducible.