Name: **BannerID:**

ShutterTalk Series Review Assignment

Imagine...

You've graduated from a top US computer engineering program and are ready to embark on an esteemed career in computing. You've been hired by a software startup called ShutterTalk, which specializes in sharing ephemeral photos between groups of friends via smartphones. This is no run of the mill startup. They've got plans. And you're here to implement them.

Problem 1: Kitchen Patrol [20 points]

ShutterTalk's office space has a shared kitchen facility. Unfortunately, it's early days, and there is not yet sufficient funding to hire staff responsible for keeping the kitchen cleaned up. Therefore, the first task for the newbie (that's you!) is to design an algorithm to assign each of the n members of ShutterTalk's technical staff into a kitchen patrol rotation. Specifically, your job is to create an n day rotation for the days $\{d_1, d_2, \ldots, d_n\}$ in which each staff member is responsible for kitchen patrol exactly once. There's a wrinkle, of course; a given staff member is not available every day, either because of vacation or because of important technical deadlines. That is, for each member of the staff, $\{s_1, s_2, \ldots, s_n\}$, there is a subset of the days when the staff member is *not* available for kitchen patrol. You have these "vacation" days accessible to you before starting your algorithm.

A *feasible schedule* is an assignment of each member of the staff to perform kitchen patrol on one of the n days such that the staff member is available on the scheduled day.

(a) **[15 points]** Describe a polynomial time algorithm that determines whether such a feasible schedule is possible and, if it is, outputs the schedule.

(b) **[5 points]** Give an asymptotically tight upper bound on the runtime of your algorithm. Briefly explain.

Problem 2: Saving Photos [25 points]
You've managed to prove yourself through the interview process and your first in-office assignment; the ShutterTalk development team is ready to trust you with a piece of their application's implementation. The goal is to define an intelligent, automated way to store a small history of the user's photos. ShutterTalk and its users assign some values to the photos that help in designing an automated algorithm. First, assume each photo has a size b_i in bytes. ShutterTalk has promised users that the application will not use more than a fixed M amount of bytes for storing photo data. Each photo is also give a *value* (v_i) that is defined as a function of the *quality* (q_i) of the photo, it's *age* (a_i), and it's estimated *social impact* (s_i) (e.g., based on the number of captured faces in the photo or the closeness of the identified friends in the photo). That is, each photo has a value $v_i = value(q_i, a_i, s_i)$. Your goal is to compute the optimal set of photos to keep in ShutterTalk's allocated storage. Specifically, you should keep a subset P of all available photos such that the total size of the photos in P is less than M and the total value of the photos in P is maximized.

(a) **[15 points]** Write a recursive definition of your solution. Explain your variables and notations.

(b) **[5 points]** Explain the order in which you'd need to populate the look-up table.

(c) **[5 points]** Derive the running time of your algorithm. Is the algorithm's running time polynomial in the size of the input?

Problem 3: NP or not NP? [25 points]

Your next task for ShutterTalk is more sophisticated. You've been asked to lead the development of a completely new feature: synchronous photo broadcast, in which a single photo is simultaneously delivered to a pre-specified set of receivers.

For simplicity, assume a particular ShutterTalk user can be in only one of two states at any time: each user is either a *sender* or a *receiver*. At any time, there are n senders and m receivers. When a sender wants to perform a synchronous broadcast, he selects a subset of the receivers who should all simultaneously receive the photo. At the instant the photo is delivered, all of the designated receivers should be "assigned" to the given sender such that none of them are receiving any other photos and are instead all guaranteed to be focused on this single, shared experience.

Given that there may be many requests coming at the same time, and the sets of targeted receivers may be overlapping, we phrase the general *Synchronous Sender Broadcast Scheduling* problem thusly: Given sets of senders and receivers, the set of requested receivers for each sender, and a number k, is it possible to assign receivers to senders so that (1) each receiver is assigned to not more than one sender, (2) at least k senders will be active, and (3) every active sender is assigned all of its requested receivers.

The Independent Set Problem. *Given a graph G and a number k, does G contain an independent set of at least k? In a graph $G = (V, E)$, a set of nodes $S \subseteq V$ is* independent *if no two nodes in S are joined by an edge.*

(a) **[5 points]** Because of his work in algorithms, your co-workers quickly realizes that this looks like an NP-Complete algorithm. When he mentions that, you vaguely remember hearing something about that concept in a class he dozed through. Your co-worker eagerly says the following:

> *We can prove that* **Synchronous Sender Broadcast Scheduling** *is an NP-Complete problem by showing that* **Synchronous Sender Broadcast Scheduling** *can be reduced to an existing NP-Complete problem in polynomial time!*

Is your co-worker right? Why or why not? (Hint: what are the implications of this reduction?)

(b) **[5 points]** Explain in a few sentences what is meant by the expression **Independent Sets $\leq_P$ Synchronous Sender Broadcast Scheduling**; in particular, we are looking for the explanation of the symbol $\leq_P$.

(c) **[5 points]** Prove that your decision version of **Synchronous Sender Broadcast Scheduling** is NP.

(d) **[10 points]** Prove that your decision version of **Synchronous Sender Broadcast Scheduling** is an NP-Complete problem.