Introduction to Algorithms	**Problem Set 5**
CS 482 Spring 2000	**Due March 27, 2000**

Directions: Each of the three questions below specifies a particular computational problem. For each one, you should show that it is at least as hard as one of the problems introduced in Section 6.1 of the course packet.

Specifically, for each question, choose a problem X from among

> *Set Packing*, *Independent Set*, *Set Cover*, *Vertex Cover*, *3-Dimensional Matching*, *Traveling Salesman*, *Hamiltonian Cycle*, *SAT*, *3-SAT*, and *Subset Sum.*

Then, show that X is polynomial-time reducible to the problem in question, in the sense of Section 6.2. (I.e. $X \leq_P \langle$the given problem$\rangle$).

(1) *Meal Planning*

A number of your friends live at that popular off-campus co-operative living unit, The Upson Collective. Taking into account the restrictions on eating meat, eggs, and cheese, people's allergies to seitan, and the rising cost of organic produce, the planning of meals at the U.C. has become quite a computational challenge.

At the moment, they're musing about the following problem. They currently have on hand a list of n ingredients, labeled $I_1, I_2, \ldots I_n$. They also have a list of t recipes that they're willing to make; the i^{th} recipe is specified by the set $R_i \subseteq \{I_1, I_2, \ldots, I_n\}$ of the ingredients that it requires.

Over the next k days, they want to use up all the ingredients by making a sequence of k recipes from the list $R_1, R_2, \ldots, R_t$. You can assume that if a recipe calls for some ingredient I_j, it completely uses up their supply of this ingredient.

Given k, $\{I_1, I_2, \ldots, I_n\}$, and $\{R_1, R_2, \ldots, R_t\}$, the *Meal Planning* problem asks: can you select a sequence of k recipes so as to consume all the ingredients?

(2) *Multiple Interval Scheduling*

We've seen the Interval Scheduling problem in class; here we consider a computationally much harder version of it that we'll call *Multiple Interval Scheduling.* As before, you have a processor that is available to run jobs over some period of time. (E.g. 9 AM to 5 PM.)

People submit jobs to run on the processor; the processor can only work on one job at any single point in time. Jobs in this model, however, are more complicated than we've seen in the past: each job requires a *set* of intervals of time during which it needs to use the processor. Thus, for example, a single job could require the processor from 10 AM to 11 AM, and again from 2 PM to 3 PM. If you accept this job, it ties up your processor during those two hours, but you could still accept jobs that need any other time periods (including the hours from 11 to 2).

Now, given a set of n jobs, each specified by a set of time intervals, and a target number k, the *Multiple Interval Scheduling* problem asks: Is it possible to accept at least k of the jobs so that no two of the accepted jobs have any overlap in time?

(3) *Monotone Satisfiability with Few True Variables*

Consider an instance of the *Satisfiability* problem, specified by clauses $C_1, \ldots, C_k$ over a set of Boolean variables $x_1, \ldots, x_n$. We say that the instance is *monotone* if each term in each clause consists of a non-negated variable; that is, each term is equal to x_i, for some i, rather than $\overline{x_i}$.

Monotone instances of *Satisfiability* are very easy to solve: they are always satisfiable, by setting each variable equal to 1.

For example, suppose we have the three clauses

$$(x_1 \vee x_2), (x_1 \vee x_3), (x_2 \vee x_3).$$

This is monotone, and indeed the assignment that sets all three variables to 1 satisfies all the clauses. But we can observe that this is not the only satisfying assignment; we could also have set x_1 and x_2 to 1, and x_3 to 0. Indeed, for any monotone instance, it is natural to ask how few variables we need to set to 1 in order to satisfy it.

Given a monotone instance of *Satisfiability*, together with a number k, the problem of *Monotone Satisfiability with Few True Variables* asks: is there a satisfying assignment for the instance in which at most k variables are set to 1?