There are two written questions on this homework. **No late days allowed for this homework** due to the upcoming prelim. We will post solutions as soon as the homeworks are in, and will release grades before the prelim. Your response to each written question needs to be at most two pages long.

Your homework submissions for written questions need to be typeset (hand-drawn figures are OK). See the course web page for suggestions on typing formulas. The solution to each written question needs to be uploaded to CMS as a separate pdf file. To help provide anonymity in your grading, do not write your name on the homework (CMS will know it's your submission).

For a proof that a problem is NP-complete, you must prove that the problem is both in NP and NP-hard. The proof of the first may often be very short (a few sentences) to motivate the certificate and certifier. The NP-hardness proof requires selecting a problem X known to be NP-complete, and showing that the new problem is at least as hard as problem X. Showing this requires three parts: a reduction, an argument this reduction takes polynomial time in the size of the problem, and the proof of correctness of the reduction. To prove the problems below NP-complete you can use any problem that was showing in class to be NP-complete.

Collaboration is encouraged while solving the problems, but

1. list the names of those with whom you collaborated with (as a separate file submitted on CMS);
2. you must write up the solutions in your own words;
3. you must write your own code.

(1) Monotone SAT (10 points).

A monotone SAT formula is a SAT formula with no negated variables. So, for example,

$$\Phi = (x_1 \vee x_2 \vee x_3) \wedge (x_2 \vee x_3 \vee x_4) \wedge (x_1 \vee x_4)$$

is a monotone formula. A monotone formula is easy to satisfy, we can simply set all variables to be *true*. In the Monotone SAT problem we are given a monotone formula and an integer k, and ask if there is a way to satisfy formula Φ with setting at most k variables true (and the rest of the variables false).

Show that the Monotone SAT problem is NP-complete.

(2) Weighted Multiple Interval Scheduling (10 points). Recall the weighted interval scheduling problem, where we had n possible jobs, each job i had value v_i and required a resource for a fixed interval of time $[s_i, f_i]$. The problem was to find a set of jobs I that request disjoint intervals and have maximum total value $\sum_{i \in I} v_i$. A decision version of the problem would have an additional input value v and ask if there is a set I requesting disjoint intervals with total value at least v, that is $\sum_{i \in I} v_i \geq v$.

In this problem, jobs may require a few disjoint intervals. For example, a single job i may require the resource for [9am,9:10am] and then again [2pm, 2:22pm] and you can only accept the job i if it is possible to assign the resource to job i for all the requested intervals of time.

The input to the problem consist of a target value v, and a set of n jobs, each with a value v_i and a set of intervals of time called R_i, where R_i is the set of times job i needs the resource (may consist of multiple intervals).[1] The problem is to decide if there is a subset of jobs I with $\sum_{i \in I} v_i \geq v$, such that all requests of all the jobs included in I are disjoint, i.e. $\cup_{i \in I} R_i$ consists of disjoint intervals. We call this the Weighted Multiple Interval Scheduling problem.

[1] You may assume that all numbers involved (values and beginning and end of intervals) are integers, but can be arbitrary large numbers.

For example if the minimum total value is $v = 2$ and job 1 has $R_1 = \{[1,3],[4,6]\}, v_1 = 1$, job 2 has $R_2\{[7,8]\}$, $v_2 = 1$, and job 3 has $R_3 = \{[2,4],[7,8]\}$, $v_3 = 1$, then choosing $I = \{1,2\}$ would satisfy the requirements. However $I = \{1,3\}$ would not since $[2,4]$ overlaps with $[1,3]$.

Show that the Weighted Multiple Interval Scheduling problem is NP-complete.