1. (25 points)

   Consider the following scheduling problem (which can be viewed as a generalization of the knapsack problem). The input consists of n jobs $J(1), \ldots, J(n)$ where $J(i) = [p_i, d_i, v_i]$ where (respectively) p_i is the processing time (the deadline, the value) of job $J(i)$. Jobs that complete their processing by their deadline contribute their value to the total profit of the schedule and as in interval scheduling, scheduled jobs cannot intersect. The goal is to schedule jobs so as to maximize the total profit. (Late jobs have no value and hence need not be scheduled.) Note that the knapsack problem is the special case when $d_i = W$ for all i.

   (a) (10 points)
   Show that if a subset S of jobs can be scheduled, then these jobs can be scheduled so that for all $J(i), J(j) \in S$, if $J(i)$ is scheduled before $J(j)$ then $d_i \leq d_j$.

   SOLUTION: This is established by an exhange argument.

   (b) (15 points)
   Suppose $V = \sum_i v_i \leq n^2$ and that all v_i are integers. Provide a polynomial time dynamic programming algorithm that will compute the optimal value of a feasible solution. Provide appropriate semantic and computational (i.e. recursively defined) arrays and indicate the time complexity of your algorithm.

   SOLUTION: Having established part (a), we can assume that in any solution (including an optimal solution), the jobs are moved as early as possoble without causing a conflict. We now assume that $i \leq j$ implies $d(i) \leq d(j)$. As stated in class, we can define the following semantic array:
   $W[i, v]$ = earliest finishing time t such that there is a feasible set $S \subseteq \{J(1), \ldots, J(i)\}$ that can be scheduled so as to finish at time t and obtain profit at least v without any job in S violatng its deadline. The array is defined for $0 \leq i \leq n$ and $0 \leq v \leq V$. (For convenience we will also define $W[i, v] = 0$ for $v \leq 0$.)

   NOTE: I have changed the previous solution to give a more efficient solution.

   We will define an equivalent computational array as follows:
   $W[i, v] = 0$ for all $v \leq 0$

   $W[0, v] = \infty$ for all $v > 0$

   $W[i, v] = \min\{A, B\}$ where $A = W[i-1, v]$ and

$B = W[i-1, v-v_i] + p_i$ if $W[i-1, v-v_i] + p_i \leq d_i$, else ∞.
The desired result is the maximum v for which $W[n, v] < \infty$.

The complexity is $O(n^3)$ since there are $n+1$ choices for i, n^2+1 choices for v and each entry takes time $O(1)$ given the previous entries.

2. (20 points)

Consider the following "more than a fourth frequency problem". We are given a set A of $n \geq 4$ elements which cannot be sorted (e.g. the elements are polynomials or matrices) but there is a test for equality of any two elements.

Describe a divide and conquer algorithm to find all the elements x (if any) such that x occurs $n_x \geq \lceil n/4 \rceil$ times in A.

Give a recurrence that describes the number of pairwise equality tests made by your algorithm. Solve the recurrence and derive a bound on the number of equality tests using your algorithm.
Hint: How many such elements can A have? Also note that a frequently occurring element must occur frequently in at least one half of the input. You may want to generalize the problem so as to make it more amenable to the divide and conquer paradigm.

SOLUTION: The main observation (as stated above) is that any element that occurs frequently must occur frequently (at the same at least one fourth density) in at least one half of the input. So we can let $S[i,j] = \{a \in A : a \text{ occurs at least } \lceil (j-i+1)/4 \rceil \text{ times in } A[i..j]\}$ where $A[i..j]$ is the subset of elements $\{a_i, \ldots a_j\} \subseteq A$. Then we wish to compute $S[i,j]$ for all $i \leq j$.

The base case(s) are that $S[i,j] = A[i..j]$ if $j - i \leq 3$.

For $j > i+3$,
Let $k = \lfloor j - i + 1 \rfloor$
$S[i,j] := \emptyset$
Recursively compute $S[i,k]$ and $S[k+1,j]$
For each $a \in S[i,k] \cup S[k+1,j]$
    Count the number n_a of times a occurs in A.
    If $n_a \geq \lceil (j-i+1)/4 \rceil$
        then $S[i,j] := S[i,j] \cup \{a\}$
    End If
End For

The complexity recurrence is $T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + O(n)$.

The complexity is the same as if we had $n = 2^r$ and $T(n) = 2T(n/2) + O(n)$; that is, we get $O(n \log n)$.

NOTE: In this problem it is not without loss of generality that we can assume $n = 2^r$ as rounding n up to the next power of 2 will impact the relative frequency.