1. As in the problem set, consider the unit profit job scheduling problem where the $i^{th}$ job has release time $r_i$, processing time $p_i$ and deadline $d_i$. All parameters are positive integers. A schedule is a mapping $\sigma : \{1, \ldots, n\} \rightarrow \{1, \ldots, max_d d_j\} \cup \{\infty\}$. That is, $\sigma$ schedules job $j$ to start at time $\sigma(j)$ if $\sigma(j) \neq \infty$ and does not schedule jobs with $\sigma(j) = \infty$. A feasible schedule $\sigma$ satisfies the properties:
   1) $\sigma(j) \neq \infty$ and $\sigma(k) \neq \infty$ and $j \neq k$ implies $[\sigma(j), \sigma(j) + p_j] \cap [\sigma(k), \sigma(k) + p_k] = \emptyset$
   2) $\sigma(j) \neq \infty$ implies $r_j \leq \sigma(j)$
   3) $\sigma(j) \neq \infty$ implies $\sigma(j) + p_j \leq d_j$.

Define the SPT greedy algorithm for job scheduling on one processor as follows: The algorithm first sorts the jobs so that $p_1 \leq p_2 \ldots \leq p_n$. In each iteration $i$, if the algorithm can schedule the $i^{th}$ job, it schedules it as early as possible consistent with jobs already scheduled.

The objective is to maximize the number of jobs scheduled in a feasible schedule $\sigma$.

(a) Show that there is some input $I$ for which $|OPT(I)| \geq 3 \cdot |SPT(I)|$ where $|A(I)|$ denotes the number of jobs scheduled by $A$. [10 points]

Let $J_1 = (3, 2, 9), J_2 = (1, 3, 4)$ and $J_3 = (4, 3, 7)$. That is, jobs 2 and 3 are intervals each having 3 units of processing time while job 1 has 2 units of processing time and can be scheduled anytime starting at time 3 and ending by time 9. SPT will schedule $J_1$ in the slot $[3, 5]$ thus preventing $J_2$ and $J_3$ from being scheduled. But an OPT schedule will schedule all three jobs with $J_1$ scheduled in slot $[7, 9]$. 
(b) Show that $SPT$ always achieves a 3-approximation of the optimal; that is, $orall I |OPT(I)| \leq 3 \cdot |SPT(I)|$.

Hint: Construct a 2-1 mapping $h : OPT(I) - SPT(I) \rightarrow SPT(I)$ where $OPT(I) - SPT(I) = \{j | j \in OPT(I) \text{ and } j \notin SPT(I)\}$. [15 points]

The mapping $h$ is defined as follows: a job $j \in OPT(I) - SPT(I)$ gets mapped to the leftmost job in $SPT(I)$ with which it intersects. We need to show

- $h$ is defined for every $j \in OPT(I) - SPT(I)$. This follows since $j$ must intersect some job in $SPT(I)$ or else it would have been scheduled by $SPT$.
- $h$ is a 2-1 mapping; that is, for every $k \in SPT(I)$ there are at most two jobs $i$ and $j \in OPT(I) - SPT(I)$ such that $h(i) = h(j) = k$. Suppose $i$ and $j$ are both mapped to $k$. Since $i$ and $j$ are not scheduled by $SPT$, and intersect $k \in SPT(I)$, it must be that $p_i \geq p_k$ and $p_j \geq p_k$ by the SPT rule. This implies that one of these jobs, say $i$, must overlap the scheduling of job $k$ and start to the left of where $k$ starts while the other job $j$ must intersect $k$ and finish to the right of where $k$ finishes. This insures that any other job in $OPT(I) - SPT(I)$ must start after $k$ starts and finish before $k$ finishes and hence have a shorter processing time than $k$ which means that it would have been scheduled before $k$ by $SPT$. 


2. Consider (as in the problem set) the following multiple copy knapsack problem. Given a set of \( n \) items with positive weights \( w_1, \ldots, w_n \) and positive integer values or gains \( g_1, \ldots, g_n \) and a knapsack of capacity \( C \), construct a sequence \( S \) of non-negative integer multiplicities \( m_1, \ldots, m_n \) such that

- \( \sum_{1 \leq i \leq n} m_i \cdot w_i \leq C \) i.e. \( S \) is a feasible sequence
- The sequence \( S \) produces the maximum value \( g(S) \) for all feasible sequences where \( g(S) = \sum_{1 \leq i \leq n} m_i \cdot g_i \).

Outline a dynamic programming solution with complexity \( O(nG) \) for this problem where \( G \) is some known upper bound on the maximum profit achieveable by some feasible sequence \( S \). That is, you are told that \( S = < m_1, \ldots, m_n > \) and \( \sum_{1 \leq i \leq n} m_i \cdot w_i \leq C \) implies \( g(S) \leq G \).

(a) Define an appropriate semantic array \( A \) and show how to compute the optimal value from \( A \). [10 points]

Define \( A(g) = \min \{ w \mid \text{there is a sequence } S = < m_1, \ldots, m_n > \text{ with } \sum_{1 \leq i \leq n} m_i \cdot w_i \leq w \text{ and } g(S) \geq g \} \).

Then the optimal value is \( \max \{ g \mid A(g) \leq C \} \).

(b) Give an equivalent recursively defined computational array \( \tilde{A} \). [10 points]

\[
\tilde{A}(0) = 0 \\
\tilde{A}(g) = \min_i \{ \tilde{A}(g - \min \{ g, g_i \}) + w_i \} \text{ for } g > 0
\]

(c) Intuitively justify why \( A = \tilde{A} \) and briefly say how you would prove this equality. [5 points]

Intuitively, the recurrence is trying to find the last item that has been put into an optimal knapsack with value at least \( g \). If that is item \( i \), then it will require weight \( w_i \) for that item and we will need at least \( g - g_i \) profit amongst the remaining items. If \( g_i \geq g \), then we can obtain profit at least \( g \) just from item \( i \).

We would prove \( A(g) = \tilde{A}(g) \) for \( 0 \leq g \leq G \) by induction on \( g \) with the base case being \( g = 0 \).