Scheduling Theory **CO 454, Spring 2009**

Homework 3

Due: June 18th, 2009

1. **(6)**
   Run the dynamic programming algorithm done in class for $(1||\sum w_j U_j)$ for the following data.

Jobs	1	2	3	4	5
p_j	2	3	1	2	2
d_j	2	4	3	6	5
w_j	3	4.5	1	2	3

2. **(6)**
   In class, we saw a dynamic program to solve $(1||\sum w_j U_j)$ problem in time $O(n \sum_j p_j)$. Give a dynamic programming algorithm to solve the problem in time $O(n \sum_j w_j)$. *(Hint: Construct a table with entries indexed by items and weight with $T[i, W]$ indicating a feasible subset of weight exactly W having minimum total processing time.)*

3. **(2+2+2)** For each of the statements, write true or false giving reasons.
   - If $X \leq_P Y$ and $Y \leq_P Z$, then $X \leq_P Z$.
   - If $X \leq_P Y$ and Y is NP-hard then X is NP-hard.
   - Let X be a problem in the class NP. If $P \neq NP$, then X cannot be solved in polynomial time.

4. **(3+3)**
   (a) The HPP (Hamiltonian path problem) is the following: given a graph G is there a simple path which contains every vertex of G. Recall that HCP (Hamiltonian cycle problem) was given a graph G, if there is a cycle containing each vertex of G. Show that $HCP \leq_P HPP$. (*Hint: Think what happens in the HPP problem if there is one vertex of degree* 1. *What happens if there are two vertices of degree* 1.)

   (b) In class we saw that $HCP \leq_P TSP$ which showed that TSP was NP-hard. Show that it is NP-hard to obtain a tour of total length at most βC^* for any $\beta > 1$, where C^* is the length of the optimal tour.

5. (a) **(3)**
   Show that the problem $(1|r_j|L_{max})$ is NP-hard by reducing the partition problem done in class to it. *(Hint: Given an instance of the partition problem, construct an instance of jobs with release dates such that if there is a partition no job is late, if there is no partition, at least one job is late)*

   (b) **(3)**
   The above only shows that $(1|r_j|L_{max})$ is weakly NP-hard since partition is only a weakly NP-hard problem. Show that $(1|r_j|L_{max})$ is strongly NP-hard by reducing Bin Packing, which is a strongly NP-hard problem, to it.

   Bin Packing: Given k items with sizes $(a_1, \ldots, a_k)$ and t bins each of capacity B, can one partition the items into the t bins such that the total size of the items in any bin is at most B.

   (Hint: Given an instance of Bin Packing *construct an instance with $k + (t-1)$ jobs, where the first k jobs correspond to the sizes and the last $(t-1)$ jobs "partition" the jobs into bins.)*