Homework1
(Due on October 14th, 2003)
(The homework is due at the beginning of class. In this fashion, we can discuss solutions in class.)

1. When the goal of scheduling is to meet deadlines, there is no advantage in completing any job sooner than necessary. We may even want to postpone the execution of hard-real-time jobs for some reason (e.g. to enable soft real-time jobs, whose response times are important, to complete earlier). For this reason, we sometimes also use the latest release time (LRT) algorithm. This algorithm treats release times as deadlines and deadlines as release times and schedules the jobs backwards, starting from the latest deadline of all jobs, in priority-driven manner, to current time. In particular, the priorities are based on the release times of jobs: the later the release time, the higher the priority.

Prove the following lemma:

**Lemma:** [Optimality of LRT] When preemption is allowed and jobs do not contend for resources, the LRT algorithm can produce a feasible schedule of a set $J$ of jobs with arbitrary release times and deadlines on a single processor if and only if feasible schedules of $J$ exist.

2. We have seen an example in class of a non-preemptable job set for which EDF is not optimal, i.e. EDF fails to find a feasible schedule although such a schedule exists.

Show that EDF is again optimal for non-preemptable job sets if we restrict ourselves to job sets with identical release times by proving the following theorem:

**Theorem:** [Optimality of EDF for non-preemptable jobs with identical release times] When preemption is not allowed and jobs do not contend for resources, the EDF algorithm can produce a feasible schedule of a set $J$ of jobs with identical release times and arbitrary deadlines on a single processor if and only if feasible schedules of $J$ exist.

3. For some systems, precedence constraints can be rather elaborate. In this problem, we want to analyze a system that has the following job set $J = \{J_i = (J_{Ai}, J_{Bi}, J_{Ai} \rightarrow_d J_{Bi}), i = 1, \ldots, N\}$, that is, each job $J_i$ consists of two jobs (call them two parts, if you wish) $J_{Ai}$ and $J_{Bi}$ with the precedence constraint relation $J_{Ai} \rightarrow_d J_{Bi}$ meaning that $J_{Bi}$ becomes ready to execute exactly $d$ time units after $J_{Ai}$ terminates.

Assume that $J_{Ai}$ and $J_{Bi}$ are unit-length, i.e. $e_{Ai} = 1$ and $e_{Bi} = 1$ for all $i$. The release time $r_i$ is defined as the point in time when $J_{Ai}$ is released; the deadline $d_i$ as the point
in time by which $J_{Bi}$ must be completed. Both $r_i$ and $d_i$ are multiples of a single time unit, i.e. $r_i, d_i \in \mathbb{N}$.

We use the following algorithm EDF$_{-d}$:

(a) For all $i$, define $r_{Ai}$ to be $r_i$.
(b) For all $i$, define (for now) $r_{Bi}$ to be infinity.
(c) For all $i$, define $d_{Bi}$ to be $d_i$.
(d) For all $i$, define $d_{Ai}$ to be $d_i - d - e_i = d_i - d - 1$.
(e) Schedule the jobs in the following way:
   - Whenever the processor is idle, select for execution that job with the earliest deadline among all the ready jobs.
   - If at some time $t$ the processor completes a job of type $J_{Ai}$, set the release time $r_{Bi}$ of the corresponding job $J_{Bi}$ to be $t + d$.

Prove that EDF$_{-d}$ is optimal, that is:

**Theorem:** [Optimality of EDF$_{-d}$] When jobs do not content for resources, the EDF$_{-d}$ algorithm can produce a feasible schedule of a set $J = \{J_{Ai} \rightarrow_d J_{Bi}, i = 1, \ldots, N\}$ of jobs with zero release times and arbitrary integer deadlines and unit execution times on a single processor if and only if feasible schedules of $J$ exist.

**Hint:** As a first step, it may help to show that preemption is not an issue if release times are multiples of the execution time, as we have in this example.

4. We determined in class that the schedulable utilization of FIFO for periodic tasks with relative deadline at the end of the period is 0 for FIFO if we limit ourselves to unit-size execution times and integer-length periods? (By *unit-size* we mean “one time unit long” and by *integer-length* we mean “an integer multiple of a time unit long”.)

5. 5.3 in [Liu].

6. 6.17 in [Liu].

7. 6.23 in [Liu].

8. 6.37 in [Liu].

9. Give a formula for the schedulable utilization of RM for a task sets with given period ratio $\delta$.

10. A class of simplified (sufficient, but not necessary) schedulability test formulae based on time demand have the following form:

$$e_i + B_i + I_i \leq D_i,$$

where $e_i$ denotes the execution time, $B_i$ the blocking time, and $D_i$ the relative deadline for Task $T_i$. The term $I_i$ denotes the amount of time higher-priority tasks interfere
with the execution of Task $T_i$.
The accuracy of the schedulability test then depends on $I_i$. For example, $I_i$ can be defined as follows:

$$I_i = \sum_{k=1}^{i-1} \left\lfloor \frac{D_i}{p_k} \right\rfloor * e_k.$$  \hspace{1cm} (2)

This formula is not accurate. (It rejects tasks that can be feasibly scheduled.) How would you improve this formula?