The homework assignments will not be graded. Please solve these problems this week (by October 8th) as an exercise for self-assessment. You may expect similar questions in quizzes and in the exams. The solutions will be provided next week for your convenience.

**Problem 1:**

Consider a preemptive priority scheduling algorithm based on dynamically changing priorities. Larger priority numbers imply higher priority. When a process is waiting for the CPU (in the ready queue, but not running), its priority changes at a rate $\alpha$; when it is running, its priority changes at a rate $\beta$. All processes are given a priority of 0 when they enter the ready queue. The parameters $\alpha$ and $\beta$ can be set to give many different scheduling algorithms.

a. What is the algorithm that results from $\beta > \alpha > 0$?

b. What is the algorithm that results from $\alpha < \beta < 0$?

**Problem 2:**

Consider a system running ten I/O-bound tasks and one CPU-bound task. Assume that the I/O-bound tasks issue an I/O operation once for every millisecond of CPU computing and that each I/O operation takes 10 milliseconds to complete. Also assume that the context switching overhead is 0.1 millisecond and that all processes are long-running tasks. What is the CPU utilization for a round-robin scheduler when:

a. The time quantum is 1 millisecond

b. The time quantum is 10 milliseconds

**Problem 3:**

Which of the following scheduling algorithms could result in starvation? Explain the reasoning.

a. First-come, first-served
b. Shortest job first
c. Round robin
d. Priority
**Problem 4:**

Write a bounded-buffer monitor in which the buffers (portions) are embedded within the monitor itself.

**Problem 5:**

Consider the deadlock situation that could occur in the dining-philosophers problem when the philosophers obtain the chopsticks one at a time. Discuss how the four necessary conditions for deadlock indeed hold in this setting. Discuss how deadlocks could be avoided by eliminating any one of the four conditions.

**Problem 6:**

Consider the dining-philosophers problem where the chopsticks are placed at the center of the table and any two of them could be used by a philosopher. Assume that requests for chopsticks are made one at a time. Describe a simple rule for determining whether a particular request could be satisfied without causing deadlock given the current allocation of chopsticks to philosophers.

**Problem 7:**

Show that, if the `wait()` and `signal()` semaphore operations are not executed atomically, then mutual exclusion may be violated.

**Problem 8:**

Write a monitor that implements an *alarm clock* that enables a calling program to delay itself for a specified number of time units (*ticks*). You may assume the existence of a real hardware clock that invokes a procedure *tick* in your monitor at regular intervals.