1. (25 points.) Solve the dining philosophers problem using conditional critical regions. Keep the solution as simple and elegant as possible. Inelegant solutions are worth less points. In particular, using the semaphore-based textbook solution, along with implementing semaphores using conditional critical regions, is worth zero points.

Solution: The solution is particularly simple, since critical regions are such an expressive means of solving synchronization problems. Let fork be a shared logical array of size 5, initially all TRUE. Philosopher $i$ needs access to fork (i) and fork ((i+1) mod 5). These can be tested in the conditional part of a conditional critical region, simultaneously (and with no race condition). If TRUE, they can both be set to FALSE without any interruption. Don’t confuse the word ”fork” (a noun, in this case) with the ”fork” system call (a verb, in Unix).

```c
philosopher (i)
{
   while true do
   {
      think
      get_forks (i)
      eat
      put_forks (i)
   }
}

get_forks (i)
{
   region fork when (fork (i) and fork ((i+1) mod 5)) do
   {
      fork (i) = false
      fork ((i+1) mod 5) = false
   }
}

put_forks (i)
{
   region fork do
   {
      fork (i) = true
      fork ((i+1) mod 5) = true
   }
}
```
2. (24 points.) The read-write head of a magnetic disk is currently satisfying an I/O request for a block at cylinder 10, and is moving towards higher-numbered cylinders. Eight other I/O requests are queued, which require blocks located at the following cylinders (in order): 8, 22, 39, 21, 5, 1, 35, 23. Show the order in which requests are serviced if the following algorithms are used. Note: for this problem, no partial credit can be given, since some of the answers may differ only slightly from each other.

(a) (6 points) First-Come-First-Served
(b) (6 points) Shortest seek time first
(c) (6 points) Look
(d) (6 points) C-Look

Solution: Note to the TA on grading: If the student made a small mistake, such as drawing a correct diagram but then leaving out one of the numbers when they list the sequence, then that’s OK (full credit). Or, if the correct sequence of numbers is given, then the diagram (if any) can be slightly incorrect (full credit). Large deviations between a drawing (if any) and a list of numbers, though, can’t be interpreted accurately - so if it is unclear as to what the student meant, then it’s worth zero. On the other hand, if the drawing is clearly a ”working doodle”, then take the list as the solution and ignore the doodles. If you have any questions, see me.

(a) First-Come-First-Served: 8, 22, 39, 21, 5, 1, 35, 23.
(b) Shortest seek time first: 8, 5, 1, 21, 22, 23, 35, 39.
(c) Look: 21, 22, 23, 35, 39, 8, 5, 1.
(d) C-Look:21, 22, 23, 35, 39, 1, 5, 8.

3. (25 points.) Measurements of a certain system have shown that the average process runs for a time $T$ before blocking on I/O. A process switch requires a time $S$, which is effectively wasted (operating system overhead). For round-robin scheduling with quantum $Q$, give a formula for the CPU efficiency for each of the following. Show your work and describe your derivations. Assume that the ready queue is never empty.

(a) (5 points) $Q$ is infinite.
(b) (5 points) $Q > T$
(c) (5 points) $S < Q < T$
(d) (5 points) $Q = S$
(e) (5 points) $Q$ is nearly zero.

Solution:
The CPU efficiency is the useful CPU time divided by the total CPU time. When $Q \geq T$, the basic cycle is for the process to run for $T$ and undergo a process switch for
Thus (a) and (b) have an efficiency of $T/(S+T)$. When the quantum is shorter than $T$, each run of $T$ will require $T/Q$ process switches, wasting a time $ST/Q$. The efficiency here is then $T/(T+(ST/Q))$ which reduces to $Q/(Q+S)$, which is the answer to (c). For (d), we just substitute $Q$ for $S$ and find that the efficient is 50%. Finally, for (e), as $Q$ goes to zero the efficiency goes to zero.

4. (26 points.) Consider the following snapshot of a system with processes P0, P1, P2, and P3, and with resource types A, B, and C.

<table>
<thead>
<tr>
<th>Process</th>
<th>Current Allocation</th>
<th>Current Request</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A  B  C</td>
<td>A  B  C</td>
</tr>
<tr>
<td>P0</td>
<td>0  1  1</td>
<td>1  0  0</td>
</tr>
<tr>
<td>P1</td>
<td>0  2  2</td>
<td>2  0  0</td>
</tr>
<tr>
<td>P2</td>
<td>0  0  1</td>
<td>0  1  1</td>
</tr>
<tr>
<td>P3</td>
<td>1  0  0</td>
<td>0  0  3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>A  B  C</td>
</tr>
<tr>
<td>1  0  0</td>
</tr>
</tbody>
</table>

(a) (5 points) Draw the resource allocation graph, if applicable to this situation. If it is not applicable, explain why not.

(b) (5 points) Draw the resource trajectory graph, if applicable to this situation. If it is not applicable, explain why not.

(c) (10 points) Is the system deadlocked? If so, which processes are deadlocked? Show your work. You may use any (appropriate) method you wish to answer this question.

(d) (6 points) Answer either question, but not both, depending on your answer to part (b):

i. If the system is deadlocked, what is the smallest amount of “damage” that a careful recovery strategy could do?

ii. If the system is not deadlocked, suppose that the single available instance of resource A becomes broken. Is the system now deadlocked? Show your work. You may use any (appropriate) method you wish to answer this question.

Solution:

(a) The graph is shown below (hand drawn - posted on my door).

(b) The resource trajectory graph cannot be used, for two reasons:

- the process’ order of allocation and release of resources is not given. It is required for the resource trajectory graph.
- it would take a 4-dimensional graph, which is rather difficult.
(c) The system is deadlocked. This cannot be inferred from the graph, though, unless you have good eyes. It's easier to detect this by running the banker's deadlock detection algorithm. If we give (1,0,0) to P0, it can complete. It returns its current allocation, so that the resources available are (1,1,1). Now P2 can complete, by giving it (0,1,1). When it finishes it returns its current allocation. The resources available are now (1,1,2). This is not enough for P1, who wants (2,0,0), and it is not enough for P3, who wants (0,0,3). The system is deadlocked; processes P1 and P3 cannot continue.

(d) i. P1 could continue if one of A is preempted from P3; P3 could continue if one of C is preempted from P1... both P1 and P3 are waiting for an event that only P1 and/or P3 can generate ... which is deadlock.

ii. This question is not applicable.