Problem 1 (0 Points)

Consider the following page reference sequence:

\[1, 3, 3, 3, 3, 3, 1, 0, 3, 3, 3, 2, 4, 2, 3, 3, 2, 2, 2, 2\]

Suppose that the process is allocated 3 frames and that all frames are initially empty.

a) Which page references will cause a page fault if the LRU page replacement algorithm is used?

b) What will be the page fault rate if an LRU page replacement algorithm is used?

Problem 2 (8 Points)

Suppose that a process has been allocated 4 page frames (numbered 0 to 3) and initially (time = 0), frame k contains virtual page k (e.g., frame 2 contains page 2). Furthermore, suppose that the following is the reference string (RS) where the memory operation is indicated by a 'r' (read) or 'w' (write):

\[[0r 1r 2r 3r] 2r 0w 3r 1w 4r 1r 0w 1r 2r 3r\]

where the bracketed references were due to the initial loading phase. For the page replacement algorithms below, fill out the table below which shows the evolution of the page table:

<table>
<thead>
<tr>
<th>Time T</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>RS</td>
<td>2r</td>
<td>0w</td>
<td>3r</td>
<td>1w</td>
<td>4r</td>
<td>1r</td>
<td>0w</td>
<td>1r</td>
<td>2r</td>
<td>3r</td>
<td></td>
</tr>
</tbody>
</table>

Assume that initially all R-bits and M-bits are 0 and that the clock cursor is pointing at frame 0. Also assume that

a) Optimal; b) FIFO; c) LRU; and d) Enhanced Clock (M-bit version).

In the case of the Optimal algorithm, assume also that the reference sequence for the time period from T=11 to T=14 is 0r, 1r, 2r, 3r.
Problem 3 (0 Points)
Problem 26, Chapter 4, Tanenbaum.

Problem 4 (0 Points)
Problem 27, Chapter 4, Tanenbaum.

Problem 5 (8 Points)
Consider the working set model with a window size of $\Delta = 3$ and the following page reference sequence for a process:

$$3r, 2r, 2r, 2r, 2r; 2r, 2r, 3r, 3w, 0r; 2r, 2r, 3r, 4r; 3w, 4r, 1r, 1r, 4w; 2r, 2r, 1r, 1r, 1r$$

a) What is the working set for $t = 5, 10, 15, 20$ and $25$? Note that these times correspond to the semicolons in the page reference sequence. Assume that the working set is computed after each page reference.

b) It is impractical to compute the working set after each page reference. Section 4.4.8 describes a more practical (although still cumbersome) algorithm. Give an example of how that algorithm would work here if the periodic clock interrupt occurred every 5 page references above; i.e., at the location of the semicolons in the page reference sequence. Assume the following:

- $tau$ is 2 ticks.
- Initially, pages 2 and 3 are the only pages in memory ($V$-bit = 1) and their R- and M-bits are 0.
- You can assume that the clock starts at $T=0$ and is incremented at each clock interrupt.

To demonstrate your answer, you will need to show how the $V$- and R-bits evolve at each page fault.
Problem 6 (8 Points)

This problem concerns the design of a homework problem involving CPU scheduling, POSIX-like threads, and discrete-event simulation. In a CPU-scheduling simulation, there is a variable \texttt{Clock} which is a non-decreasing, non-negative integer tick counter that represents the current time. In one form of the simulation, \texttt{Clock} can change in jumps in response to events (e.g., job arrives, job departs, job begins service). Each job is represented by a thread that calls the simulation functions \texttt{Acquire()}, \texttt{Delay(nticks)}, \texttt{Release}():

- \textbf{Acquire}(): The job wants to use the CPU. If some other job is using the CPU, place the job on the ready queue.
- \textbf{Release}(): The job is done using the CPU and is giving the next job the opportunity to use the CPU.
- \textbf{Delay(nticks)}: When the job acquires the CPU, it will use the CPU for \texttt{nticks}; i.e., advance \texttt{Clock} by \texttt{nticks} past the time it starts using the CPU.

a) Suppose that you are using a threads packet that has the two mutual exclusion functions \texttt{Mutex\_unlock(lock)} and \texttt{Mutex\_Unlock(lock)} and the two conditional variable signalling functions \texttt{Cond\_Wait(sig,lock)} and \texttt{Cond\_Signal(sig)}. Give an algorithm for each of the three simulation functions. You can assume that there is an existing queue class.

b) What other functions would be required to write a complete FCFS simulator?

c) Write an interesting homework problem based on the simulator and discuss what are the learning objectives of this problem.

d) Discuss whether the Banker's Algorithm has any relevance in designing an $N$-resource simulator in which a job can acquire more than one FCFS resource before it becomes active.