Problem 1 (0 Points)
Tanenbaum, Problem 40 (Chapter 2).

Problem 2 (6 Points)
Consider an I/O system in which input process I, user process P, and output process O are connected by two buffers. The processes exchange data in equal sized blocks. Furthermore, process P produces 1 output block for each input block and consumes input blocks one at a time. These blocks are buffered on disk using a floating boundary between the input and output buffers. The communication primitives ensure that the following resource constraint is satisfied:

\[ n_I + n_O \leq M \]  \hspace{1cm} (1)

where \( M \) is the maximum number of blocks on disk, \( n_I \) is the number of input blocks on disk, and \( n_O \) is the number of output blocks on disk.

- a) If all processes will eventually continue execution if there is sufficient disk buffers, under what circumstances will the system deadlock if buffers are never preempted?
- b) What additional constraint(s) is (are) required to prevent deadlock but still permit the boundary between input and output buffers to vary in accordance to the present needs of the processes?
- c) Suppose that the disk space must also hold buffers for intermediate processing by process P as it computes the contents of the corresponding output buffer. Furthermore, suppose that there are only 4 input message types, and a type \( i \) message requires 1 input buffer and \( p_i \), \( i = 1, 2, 3, 4 \), processing buffers. What additional constraint(s) is (are) required to avoid deadlock but still permit the boundary between input, output, and processing buffers to vary in accordance to the present needs of the processes?
Problem 3 (2 Points)

• Consider the case of two tasks ($N = 2$) where the service demands are $t(1) = 5$ and $t(2) = 3$. What is the average response time for the two possible service orderings (i.e., job 1 then job 2 and job 2 then job 1)?

• Consider the more general case of $N$ jobs with service demands of $t(i)$, $i=1:N$ where the service demands are strictly ordered; i.e., $t(1) < t(2) < ... < t(N)$. Prove that SJF will result in the smallest average response time. HINT: Consider the average response time when Job $i$ swaps service position with Job $i-1$.

Problem 4 (2 Points)

Suppose that a user’s last four requests have used 10, 20, 30 and 40 milliseconds of CPU time respectively. What is the predicted CPU demand if we use an aging algorithm with $a = 1/2$? Assume that the first estimate is equal to the first observation 10.

Problem 5 (4 Points)

We wish to derive the equations of motion for a workload that consists of $N + 1$ jobs consisting of $N$ type A jobs and one type B job being serviced by the simple scheduling scheme described below. Job type A has a CPU demand of $a$ seconds, and job type B has a demand of $b$ seconds. Furthermore, $b$ is a large integer multiple of $a$, and type A jobs arrive at fixed time points $X$, $2X$, $3X$, etc. where $X$ is a positive integer multiple of $a$; i.e., $X = ka$ where $k$ is a positive integer.

In this system, type A jobs have a higher priority than the one type B job and will preempt (with 0 overhead) any type B job from the CPU.

a) Draw the space-time diagram (time runs to the right) for the case when $a = 1$, $b = 10$, $k = 2$, and $N = 4$.

b) Derive an expression for the turnaround time and the queuing time of the type B job and each type A job. Here, you need to handle the general case of arbitrary $N$ and $k$ although subject to the constraints specified earlier; i.e., $k$ is a positive integer, and $b$ is a large integer multiple of $a$. 

2
Problem 6 (8 Points)

This is a warm-up for Project B, the extension to Project A. Write a program called \texttt{npipe} that has the following optional flags:

\begin{verbatim}
npipe [-n N] [-v]
\end{verbatim}

where the default value for \texttt{N} is 3. \texttt{N} indicates the number of instances of \texttt{npipe}. Basically, you are implementing the pipeline "\texttt{npipe | npipe | ... | npipe}" where the \texttt{k}th process should create the pipe to the \texttt{(k + 1)th} process and fork the \texttt{(k + 1)th} process. The \texttt{-v} flag indicates verbose output where each pipe file descriptor creation and close is accompanied by a message indicating the action (create or close), the process id and the affected file descriptor. When reading/writing from/to a pipe, use unbuffered I/O (i.e., \texttt{read(2)} and \texttt{write(2)}). As a verification that each process outputs the same bytes, each process should compute the sum of all bytes treating each byte as an unsigned integer value and print this sum one second after it detects EOF on stdin.

Submit the following:

- A listing of the source code.
- The output and an explanation of a test run for the case \texttt{npipe -n 3 -v} that indicates that your program is functioning properly. A test input file will be provided.