1. (15%) A system uses a Multi-Level Feedback Queue (MLFQ) to manage a mix of interactive tasks and long-running compute jobs.
   (a) In a basic MLFQ, a process that yields the CPU before its time slice expires stays at the same priority level. Design a "fair accounting" mechanism that prevents a malicious process from "gaming" the scheduler to maintain high priority. Define how this change affects the total CPU time allocated to a process over a period ***P***.
   (b) Explain why a "Priority Boost" is necessary in an environment with many short-lived interactive tasks. If the boost interval *S* is set to 10 seconds, and a long-running job is at the lowest queue, what is the maximum percentage of CPU time it is guaranteed to lose to higher-priority interactive jobs?
   (c) In the context of a scheduler, define the difference between a mechanism (e.g., context switching) and a policy (e.g., MLFQ). Why is it important for the OS to separate these two concepts?

2. (20%) Consider a 32-bit virtual address space with 4 KB pages (2^{12} bytes) and 4-byte page-table entries (PTEs).
   (a) Instead of a single-level table, the system uses a two-level page table. If the first 10 bits of the address are used for the Page Directory and the next 10 bits for the Page Table, how many entries are in the Page Directory?
   (b) If a process only uses 1 MB of memory at the bottom of its address space, compute the total size of the page tables (Directory + active Tables) in bytes.

3. (15%) A thread is responsible for loading weights from disk into a shared buffer, and the GPU thread consumes them.
   (a) Explain why a "Spinlock" (using a test-and-set instruction) is efficient for very short critical sections but wasteful for long ones.
   (b) Why is a mutex alone insufficient to coordinate these threads?
   (c) Why must the "Consumer" thread use a while loop to check if the buffer is empty after being woken up by a signal, rather than a simple if statement?

4. (16%) Suppose that a scheduling algorithm (at the level of short-term CPU scheduling) favors those processes that have used the least processor time in the recent past.
   (a) (8%) Will this scheduling algorithm favor CPU-bound processes or I/O-bound processes? Why?
   (b) (8%) Please discuss if the starvation situation will permanently occur in this scheduling algorithm?

5. (10 %) Assume you want to implement a web-server for YouTube by using multithreading, where each thread serves one incoming request by loading a video file from the disk. Assume the OS only provides the normal blocking read system call for disk reads, do you think user-level threads or kernel-level threads should be used? Why?

6. (24%) Are the following statements true or false? For each statement, you will get 4 points for correct answer, zero point for blank, or -2 point for incorrect answer.
   (a) Little's Law can not be applied to a system with multiple queues.
   (b) The whois tool allows you to query the DNS to obtain the mapping between domain name and IP address.
   (c) Disabling interrupts while holding a spinlock can prevent deadlock between a thread and an interrupt handler.
   (d) Side-channel attacks are fundamentally designed to exploit mathematical weaknesses or logical flaws within a cryptographic algorithm's design.
   (e) Inverted page table is used to reduce the size of memory committed to page tables.
   (f) GPU internal thread scheduling is physically managed and allocated directly by the OS kernel scheduler.