Question 1: Buffer Management (Part 2) . . . . [270 points]

(i) **[10 points] Bit Manipulation:**
How do you get the k most significant bits of an unsigned integer i?

(ii) **[10 points] Bit Manipulation:**
How do you get the k least significant bits of an unsigned integer i?

(iii) **[10 points] Bit Manipulation:**
How do you print an unsigned integer i as a sequence of bits?

(iv) **[10 points] Threads vs Process:**
Define a thread of execution. How is it related to an OS process?

(v) **[10 points] Threads:**
Do threads share a virtual address space or not?

(vi) **[10 points] Threads:**
Explain how a thread may wait for another thread to complete its execution in C++.

(vii) **[10 points] Thread Safety:**
Define thread safety.

(viii) **[10 points] Multi-core Processors:**
Why do we have multi-core processors as opposed to 10 GHz single-core processors?

(ix) **[20 points] Thread Safety:**
Distinguish between shared and exclusive access. Why is it important to distinguish between these types of accesses in practice?

(x) **[10 points] Atomic Operations:**
Justify the term – "atomic".

(xi) **[10 points] Atomic Operations:**
How is atomic operations implemented using assembly instructions?

(xii) **[10 points] Atomic Operations:**
How do atomic operations enable thread safety?

(xiii) **[10 points] Atomic Operations:**
Why is a thread **not** able to load the latest value of a non-atomic counter?

(xiv) **[10 points] Atomic Operations:**
Why is a thread able to load the latest value of an atomic counter?

(xv) **[10 points] Thread-Local Storage:**
Define thread-local storage.

(xvi) **[10 points] Thread-Local Storage:**
Will the main thread be able to see the thread-local value of the worker thread?

(xvii) **[10 points] Mutual Exclusion:**
Why is `std::mutex` more general than `std::atomic`?

Question 1 continues…

(xviii) **[10 points] Mutual Exclusion:**
How does `std::mutex` use the `futex` system call?

(xix) **[10 points] Mutual Exclusion:**
How does `std::mutex` compare in speed against `std::atomic`?

(xx) **[10 points] Lock Guard:**
Why is `lock_guard` an RAII-style mechanism?

(xxi) **[10 points] Shared Mutex:**
Distinguish between `std::mutex` and `std::shared_mutex`.

(xxii) **[10 points] Shared Mutex:**
Distinguish between `lock` and `try_lock`.

(xxiii) **[20 points] Copy Constructor:**
Define copy constructor and copy assignment operators. When are they used?

(xxiv) **[20 points] 2Q:**
Explain the 2Q policy.