a. Exec() - completely replaces the current running process with program passed as an argument
b. Copy on Write – optimization strategy - fundamental idea is that if multiple callers ask for resources which are initially indistinguishable, they can all be given pointers to the same resource
c. Allocator - Allocators handle all the requests for allocation and deallocation of memory for a given container
d. Internal fragmentation – caused when payload is smaller than block size
e. External fragmentation - when there is enough aggregate memory but no block large enough

2. How would a context switch between applications affect the memory subsystem's performance?

By completely flushing the TLB, we can prevent TLB entries from becoming invalid (since virtual to physical mapping is different)

3. Why is "demand paging" generally more efficient than loading an entire application into memory?

Eventually it will keep the pages accessed the most, where as loading an entire application will result in many misses.

4. In our "fast file copy example", how does it avoid copying data into 'user space', and why does this speed things up?

5. Contrast and compare explicit vs. implicit memory allocation.

Explicit – application allocates & frees space (malloc and free)
Implicit – application allocates but does not free (garbage collection in java)

6. What affect would allowing allocators to reorder or buffer requests potentially have?

Pointers directed towards malloc'd space may no longer point to the correct location if the buffer is compacted

7. How would you implement the ability to move blocks once they've been allocated? (Hint – think early MacOS 'handles'.)

Mac OS supports a form of relocatable heap block, which is accessed indirectly via a handle which points into a *master pointer* block (a non-relocatable heap block). Heap blocks can optionally be relocatable, and relocatable blocks can be locked. Relocatable blocks are compacted from time to time to avoid external fragmentation. Relocatable blocks can also be marked purgeable, which means the system may free them during compaction if memory becomes low.

8. Does padding for byte alignment comprise internal or external fragmentation? Why?

Internal – bye padding leaves empty padding on each end.

9. Why can't malloc use the same techniques we use for virtual address spaces?

Because once we malloc something it is a fixed space we cant changes the allocated size

10. Compare and contrast immediate and deferred coalescing. How do these issues compare to Java/.Net garbage collection design issues?

Deferred coalescing is delayed joining of free blocks, immediate joins them as soon as they're free.
Since deferred waits until it has enough it will do them all at once, causing better performance however due to the stipulations of block management delayed coalescing is not useable in a real-time situation.

11. Based on the design discussed in the course notes, what is the limiting factor on the size of request you can make to malloc?

You're limited by how large of a contiguous block you can find.