1.	**Explain the distinction between a demand-paging system and a paging system with swapping.**	A demand-paging system is similar to a paging system with swapping where processes reside in secondary memory. With demand paging, when a process is executed, it is swapped into memory. Rather than swapping the entire process into memory, however, a lazy swapper is used. A lazy swapper never swaps a page into memory unless that page will be needed. Thus, a paging system with swapping manipulates entire processes, whereas a demand pager is concerned with the individual pages of a process.
2.	**Explain the sequence of events that happens when a page-fault occurs.**	When the operating system cannot load the desired page into memory, a page-fault occurs. First, the memory reference is checked for validity. In the case of an invalid request, the program will be terminated. If the request was valid, a free frame is located. A disk operation is then scheduled to read the page into the frame just found, update the page table, restart the instruction that was interrupted because of the page fault, and use the page accordingly.
3.	**How is the effective access time computed for a demand-paged memory system?**	In order to compute the effective access time, it is necessary to know the average memory access time of the system, the probability of a page fault, and the time necessary to service a page fault. The effective access time can then be computed using the formula: effective access time = (1 - probability of page fault) * memory access time + probability of page fault * page fault time.
4.	**How does the second-chance algorithm for page replacement differ from the FIFO page replacement algorithm?**	The second-chance algorithm is based on the FIFO replacement algorithm and even degenerates to FIFO in its worst-case scenario. In the second-chance algorithm, a FIFO replacement is implemented along with a reference bit. If the reference bit is set, then it is cleared, the page's arrival time is set to the current time, and the program moves along in a similar fashion through the pages until a page with a cleared reference bit is found and subsequently replaced.

5.	**Explain the concept behind prepaging.**	Paging schemes, such as pure demand paging, result in large amounts of initial page faults as the process is started. Prepaging is an attempt to prevent this high level of initial paging by bringing into memory, at one time, all of the pages that will be needed by the process.
6.	**Why doesn't a local replacement algorithm solve the problem of thrashing entirely?**	With local replacement, if one process starts thrashing, it cannot steal frames from another process and cause the latter to thrash as well. However, if processes are thrashing, they will be in the queue for the paging device most of the time. The average service time for a page fault will increase because of the longer average queue for the paging device. Thus, the effective access time will increase, even for a process that is not thrashing.
7.	**Explain the difference between programmed I/O (PIO) and interrupt driven I/O.**	To send out a long string of bytes through a memory-mapped serial port, the CPU writes one data byte to the data register to signal that it is ready for the next byte. If the CPU uses polling to watch the control bit, constantly looping to see whether the device is ready, this method of operation is called programmer I/O. If the CPU does not poll the control bit, but instead receives an interrupt when the device is ready for the next byte, the data transfer is said to be interrupt driven.
8.	**What are the benefits of using slab allocation to allocate kernel memory?**	The slab allocator provides two main benefits. First, no memory is wasted due to fragmentation. When the kernel requests memory for an object, the slab allocator returns the exact amount of memory required to represent the object. Second, memory requests can be satisfied quickly. Objects are created in advance and can be quickly allocated. Also, released objects are returned to the cache and marked as free, thus making them immediately available for subsequent requests.
9.	**How are lock bits useful in I/O requests?**	A lock bit is associated with every frame. If a frame is locked, it cannot be selected for replacement. To write a block on tape, we lock into memory the pages containing the block. The system then continues as usual with other processes if the I/O request is in a queue for that I/O

		device. This avoids the replacement of the pages for other processes and the possible unavailability of those pages when the I/O request advances to the head of the device queue. When the I/O is complete, the pages are unlocked.
10.	**Explain how copy-on-write operates.**	Copy-on-write (COW) initially allows a parent and child process to share the same pages. As long as either process is only reading—and not modifying—the shared pages, both processes can share the same pages, thus increasing system efficiency. However, as soon as either process modifies a shared page, a copy of that shared page is created, thus providing each process with its own private page. For example, assume an integer X whose value is 5 is in a shared page marked as COW. The parent process then proceeds to modify X, changing its value to 10. Since this page is marked as COW, a copy of the page is created for the parent process, which changes the value of X to 10. The value of X remains at 5 for the child process.
11.	**Explain the distinction between global allocation versus local allocation.**	When a process incurs a page fault, it must be allocated a new frame for bringing the faulting page into memory. The two general strategies for allocating a new frame are global and local allocation policies. In a global allocation scheme, a frame is allocated from any process in the system. Thus, if process A incurs a page fault, it may be allocated a page from process B. The page that is selected from process B may be based upon any of the page replacement algorithms such as LRU. Alternatively, a local allocation policy dictates that when a process incurs a page fault, it must select one of its own pages for replacement when allocating a new page.
12.	**Discuss two strategies for increasing TLB reach.**	TLB reach refers to the amount of memory accessible from the TLB and is the page size multiplied by the number of entries in the TLB. Two possible approaches for increasing TLB reach are (1) increasing the number of entries in the TLB, and (2) increasing the page size. Increasing the number of entries in the TLB is a costly strategy as the TLB consists of associative memory, which is both costly

		and power hungry. For example, by doubling the number of entries in the TLB, the TLB reach is doubled. However, increasing the page size (or providing multiple page sizes) allows system designers to maintain the size of the TLB, and yet significantly increase the TLB reach. For this reason, recent trends have moved towards increasing page sizes for increasing TLB reach.
13.	**What is the benefit of using sparse addresses in virtual memory?**	Virtual address spaces that include holes between the heap and stack are known as sparse address spaces. Using a sparse address space is beneficial because the holes can be filled as the stack or heap segments grow, or when we wish to dynamically link libraries (or possibly other shared objects) during program execution.
14.	**Explain the usefulness of a modify bit.**	A modify bit is associated with each page frame. If a frame is modified (i.e. written), the modify bit is then set. The modify bit is useful when a page is selected for replacement. If the bit is not set (the page was not modified), the page does not need to be written to disk. If the modify bit is set, the page needs to be written to disk when selected for replacement.