1. Chapter 1, problem 1.6
   The main differences between operating systems for mainframe computers and operating systems for personal computers are: mainframe computers have more concern with security, and also with fairness for multiple users' processes. Personal computer operating systems assume one user, who will probably not attempt to start as many concurrent processes and doesn't need to be protected quite as much from herself.

2. Chapter 1, problem 1.9
   A user would be better off using a time-sharing system, rather than a personal computer or single-user workstation when the user's task needs a lot of resources or a specialized processor that cannot be found in the single-user workstation or personal computer. A user is also better off in a time-sharing system if they are using big expensive peripherals that are only affordable if shared, or if they are using information that is stored only on a disk attached to the time-sharing system and cannot be accessed over the network.

3. Chapter 2, problem 2.5 - explain why each instruction should (or should not) be privileged.
   a. set value of timer - should be privileged, otherwise a process could hog the machine by resetting the timer while it runs
   b. read the clock - no need to be privileged; doesn't change anything
   c. clear memory - should be privileged; would affect other running processes (unless it were restricted to memory assigned to this process)
   d. turn off interrupts - must be privileged, otherwise a process could hog the machine by not letting waiting processes know their requests were completed
   e. switch from user to monitor mode - should be privileged because it allows access to all other privileged operations.

4. Chapter 2, problem 2.6
   It is not possible to construct a secure operating system for computers without a privileged mode of instruction, because a malicious program could always use instructions such as "turn off interrupts", or access arbitrary memory locations to disrupt other processes.

   It is possible to construct a secure operating system for computers without a privileged mode of instruction if the "virtual machine" model is used, and all instructions are intercepted by the VM (which has virtual privileged mode) before being executed (or not) on the real machine.

5. Chapter 3, problem 3.6
   Five services provided by an OS
   process and job control - users believe that their job is always running; see a machine "just for them" A user-level program would not be able to interrupt other running processes to share the CPU.
file manipulation - users can easily access their files without knowing details of their locations on the disk, system also figures out where to put new information - requires direct access to shared i/o resources that would be dangerous if done by a user-level program.
device management - users believe that devices are available when needed, details of queues are often hidden from the users, though very slow devices, such as printers, have a way for users to check the queue.
memory management - users believe that they have their own memory in which their program runs, but the OS keeps track of placing those programs in non-overlapping areas of memory and swapping processes out of memory to disk when necessary.
 communications - users can access information locally or across the network, OS can make this transparent - user-level processes can't access the network directly or they will cause problems as in any peripheral device.

6. Chapter 3, problem 3.10
For an OS designer, the virtual machine architecture provides a device-independent platform upon which to develop the OS. Developing OS for different machine architectures is simplified because only the VM needs to change. Also development and testing is possible without getting a private bare machine -- it can be done on one VM while others are serving other people. From the user's point of view the VM provides additional safety and more uptime, even during testing.

7. Consider the following processes. Simulate the operation of the machine under each of two policies for 10 cycles. Be sure to indicate the state for each process in each cycle (use the 5-state model), as well as the contents of the ready and I/O queues (assume only one I/O device).

For your first simulation, consider that each process is interrupted after every cycle of CPU time to allow another process to run. For your second, consider that a process is allowed to run until it is blocked by an I/O wait or until completion. Which policy do you think is better, and why?

Process A: starts at time step 0, uses the CPU for 2 cycles, waits for I/O (will take 3 cycles) uses the CPU for 3 cycles.

Process B: starts at time step 0, uses CPU for 1 cycle, waits for I/O (2 cycles worth), uses CPU for 3 cycles

Process C: starts at time step 2, uses CPU for 4 cycles