Question 1. Abstraction and Resource Management. (20 pts.)

In an operating system, how is the concept of abstraction tied to the notion of a privileged state? Explain in general and give one specific example.
Question 2. Resource Management. (20 pts.)

A. List and explain an event that can cause a process to move from the running state to the ready state. If no such event exists, explain why.

B. List and explain an event that can cause a process to move from the running state to the waiting state. If no such event exists, explain why.

C. List and explain an event that can cause a process to move from the waiting state to the ready state. If no such event exists, explain why.

D. List and explain an event that can cause a process to move from the ready state to the waiting state. If no such event exists, explain why.
Question 3. Process Control. (20 pts.)

A. If `fork` is called once and succeeds, how many times does it return? Explain your answer.

B. If `fork` is called once and fails, how many times does it return? Explain your answer.

C. If `exit` is called once, how many times does it return? Explain your answer.

D. If an `exec` family function is called once and succeeds, how many times does it return? Explain your answer.
E. If an `exec` family function is called once and fails, how many times does it return? Explain your answer.

Question 4. Standard I/O. (20 pts.)

A. The ANSI C Standard I/O library allows application programmers to use the same high-level function calls on any system for which an ANSI-C-compliant compiler exists. Is this an example of horizontal abstraction, vertical abstraction, neither, or both? Explain your answer.

B. Explain, in broad outline, what it means for the Standard I/O library to buffer input and output and how this is accomplished.
C. List and explain two benefits of the I/O buffering done by the Standard I/O library functions.

D. List and explain one risk involved in using I/O buffering, such as that provided by the Standard I/O library.
Question 5. Process Control. (20 pts.)

A. A fork system call creates a nearly identical child copy of the parent process that executed the fork. List two immediate differences between the parent and the child process and explain why these differences must exist at the time the child process is created.

B. In general, if we do a fork follows by an exec family function, the exec takes place in the child process, rather than the parent. List and explain one reason that this is the normal way of doing things if process synchronization is desired.