1. Consider figure 3.2 on p. 103 of the text (also on the Process States in the OS slide from class). Give at least 2 examples of events that would cause a process to transition:
   a. from running to ready.
   b. from running to waiting.

2. Notice that figure 3.2 does not contain a transition from the ready state to the waiting state. Explain why this transition is not included.

3. Some operating systems will suspend processes that have been in the waiting state for a long time by copying their memory contents to disk.
   a. Explain why it might make sense for an operating system to suspend such processes.
   b. Redraw figure 3.2 to include a suspended state.

4. From the perspective of a program, making a sleep system call causes the program to pause for a specified amount of time and then resume right where it left off. Describe in detail how a sleep system call could be implemented from the operating system perspective.

5. Operating System Concepts Chapter 3, #3.9

6. Operating System Concepts Chapter 3, #3.10

7. Operating System Concepts Chapter 3, #3.13

8. Create a C program that computes A! * fib(B). A! is the factorial of A and fib(B) is the B\textsuperscript{th} Fibonacci number. The values of A and B can be defined in the main method. Your program must use two child processes to perform this computation. One child process will compute A! and the other will compute fib(B). The parent process should use fork() to create the child processes and then wait for them to exit. The child processes should perform their computations and then return the result to the parent process through the use of shared memory IPC. You do not need to use execv for this problem. The code for both child processes can be in the same file as the code for the parent process, similar to the examples from class.