Operating Systems - Test 1 - Chapter 5 (Concurrency)

1. **Anything that is accessing a critical section first needs to ___**	acquire a lock
2. **Communication of a message between two processes implies**	synchronization between the two
3. **Define barrier synchronization**	Processes cannot proceed past a certain point until all processes have reached that point
4. **Define critical section**	Anything that is being shared
5. **Define deadlock in the context of semaphores**	Two or more processes are waiting on each other in order to get locks, but none of them can proceed due to not being able to get the locks.
6. **Define mutual exclusion**	Only one process can be in the critical section at a time.
7. **Define semaphore.**	A nonnnegative integer variable that can only be changed or tested by two atomic functions: P() and V()
8. **Define starvation in the context of semaphores**	Being indefinitely postponed from entering the CS
9. **Describe the barbershop problem.**	There is a barbershop with a waiting room, a barber's chair, and a barber who sleeps when there are no customers. When the barber is not cutting anyone's hair, he will go to the waiting room to check if there are any customers waiting. If there are, he escorts them to the barber's chair. If there are none, he goes back to his chair and sleeps. When a customer comes in, they check to see what the barber is doing.
10. **Describe the dining philosopher problem.**	5 dining philosophers are sitting around a table with 5 forks. In order to eat the spaghetti, a philosopher must have 2 forks.
11. **Describe the producer/consumer problem.**	Two processes (producer and consumer) share a common, fixed-size buffer used as a queue. The producer is repeatedly inserting data into the buffer. At the same time, the consumer is repeatedly removing data from the buffer. The problem is to ensure the producer won't insert data into a full buffer and that the consumer won't remove data from an empty buffer.
12. **Describe the readers-writer problem**	Two or more threads are attempting to access the same resource at the same time.
13. **Fill out a semaphore/critical section example chart.**	Cool.
14. **Functions used to implement lock (get(lock) and release(lock)) must be ___**	atomic
15. **If a process calls get() / P(s), it should not __**	be postponed forever
16. **In regards to processes, what occurs when an interrupt is raised?**	The states of the processes are saved
17. **In the context of concurrency, what does synchronization do?**	Enforces mutual exclusion.
18. **In the context of mutual exclusion, what does atomic mean?**	Uninterruptible
19. **In the context of synchronization, when a receive primitive is executed in a process, what are the two possibilities?**	If there is no message, the process is blocked until a message arrive or the process continues to execute, abandoning the attempt to receive; or if a message has previously been sent the message is received and execution continues
20. **In the semaphore solution, what is each lock associated with?**	A semaphore
21. **In what form is message passing provided?**	send(destination, message), receive(source, message)
22. **Label a semaphore diagram**	Cool.

23. **Only the processes that are ___must be considered for resolving who enteres the CS next.**	Competing for a CS
24. **The receiver cannot receive a message until**	it has been sent by another process
25. **What are concerns that are related to concurrency?**	Synchronization (support for mutual exclusion), communication (data sharing/message passing), protection of data/resources (access control for sharing), deadlock, resource allocation/deallocation
26. **What are disadvantages of disabling interrupts?**	User process can abuse; could be disabled infinitely long; prevents any process from executing; in a multiprocessor system, disabling interrupts in one processor does not disable them in a different processor
27. **What are other names for get()?**	P(), wait(), pthread_mutex_lock()
28. **What are other names for release()?**	V(), signal(), pthread_mutex_unlock()
29. **What are some methods of managing access?**	Disabling interrupts, locks, semaphores
30. **What are the rules for software locks?**	No assumptions should be made about the relative speeds of processes or the number of competing processes; a process should not be delayed access to a critical section when no other processes are attempting to access it; no deadlock or starvation should occur; only the processes competing for a CS are to be considered for resolving who enters the CS next
31. **What does it mean to write a routine in an atomic way?**	The routine cannot be interrupted in the middle of its running.
32. **What is another method for facilitating concurrency?**	Message passing.
33. **What is the fairest solution to the readers/writers problem?**	FIFO queue
34. **What is the ideal solution to the dining philosophers problem?**	Each fork is assigned a particular value; each philosopher is required to attempt to pick up a set fork each time (i.e. the lower valued fork) before picking up another fork. This leaves one fork open for a philosopher to finish eating with, and then the next philosopher can pick up a fork to eat, and so on.
35. **What results from a lack of mutual exclusion?**	Results of multiple executions are not consistent
36. **What two mechanisms are needed to facilitate concurrency?**	Synchronization and communication
37. **Who originally came up with the semaphore?**	Dijkstra
38. **Why should assumptions not be made about the relative speed of processes or the number of competing processes when coding?**	That information changes.
39. **Write out an example of code that will result in deadlock.**	Cool.
40. **Write out the code associated with the ideal solution to the dining philosophers problem.**	Cool.
41. **Write out the code for the ideal solution to the barbershop problem.**	Cool.

42.	**Write out the code for the semaphore struct, get(semaphore s), and release(semaphore)**	Cool.
43.	**Write out the code necessary for the fairest solution to the readers/writers problem?**	Cool.
44.	**Write out the Compare & swap implementation of semaphore.**	Cool.
45.	**Write the code for the solution to the bounded-buffer producer/consumer problem**	Cool.