Write one of the words or terms from the following list into the blank appearing to the left of the appropriate definition. Note that there are more words and terms than definitions. (1 pt. each)

affinity scheduling    false sharing    MCS lock    preemption    semaphore
busy waiting            fine-grain locking    mechanism    priority aging    spin lock
currency                gang scheduling    memory barrier    priority boosting    starvation
CPU burst               Hoare semantics    Mesa semantics    publish    swapper
critical section        Little's Law    oblivious scheduling    race condition    time quantum
dispatcher              lock ordering    policy    reader/writer lock    work conserving

1. ________________ A sequence of code that operates on shared state.

2. ________________ The short-term scheduler that typically runs after each interrupt.

3. ________________ The length of time that a task is scheduled before being preempted.

4. ________________ When a scheduler takes the processor away from one task and gives it to another.

5. ________________ When the priority of a task decreases as the task runs and increases as the task waits.

6. ________________ When the behavior of a program relies on the interleaving of operations of different threads.

7. ________________ An efficient spinlock implementation where each thread spins on a separate memory location.

8. ________________ A widely used approach to prevent deadlock, where locks are acquired in a predetermined order.

9. ________________ A way to increase concurrency by partitioning an object’s state into different subsets each protected by a different lock.

10. ________________ A scheduling policy for multiprocessors that performs all the runnable tasks for a particular process at the same time.

11. ________________ A scheduling policy where tasks are preferentially scheduled onto the same processor that they had previously been assigned, to improve cache reuse.

12. ________________ A general relationship between the average throughput, response time, and number of tasks in a stable system in which the arrival rate matches the departure rate.

13. ________________ A single, atomic memory write that updates a shared object protected by an RCU lock. The write allows new reader threads to observe the new version of the object.

14. ________________ When a thread calls signal(), execution of the signaling thread is suspended, the ownership of the lock is immediately transferred to one of the waiting threads, and execution of that thread is immediately resumed.

15. ________________ A type of synchronization variable with only two atomic operations, P() and V(). P waits for the value of the semaphore to be positive, and then atomically decrements it. V atomically increments the value, and if any threads are waiting in P, triggers the completion of the P operation.
16. Can the lost update problem happen on a uniprocessor, or is it a potential problem only for multiprocessors? If it can happen on a uniprocessor, then explain how it can happen. If it can only happen on multiprocessors, then explain why the problem cannot happen on a uniprocessor. (4 pts.)

For questions 17 to 19, fill in the blanks with numeric values (i.e., 0, 1, 2, ...) or with the symbol “n” for an arbitrary amount, and then provide an explanation as to why you listed those numbers.

17. The bzero() function writes zero in each memory word in a specified memory region. To make the function multithreaded, you must use _____ lock(s) and _____ condition variable(s). (5 pts.)

18. Mutual exclusion requires _____ lock(s) and _____ condition variable(s). (5 pts.)

19. The bounded blocking queue (BBQ) in the textbook uses _____ lock(s) and _____ condition variable(s). (5 pts.)
20. Below is a list of methods for the Queue class, the declaration of the Scheduler class from Figure 5.17 in the textbook, the declaration of the CV class from Figure 5.18, and the code for the wait method for the CV class from Figure 5.18. Please fill in the code for the signal method. If you do not know C++, then please use C-like pseudo code for your answer. (5 pts.)

```cpp
// FILL IN THE NECESSARY CODE:

// Monitor lock is held by the current thread, which is identified by Thread *myTCB
void CV::signal() {
    // Monitor lock is held by the current thread
    void CV::signal() {

    }
```

21. Show the implementation of the SpinLock acquire() method from Figure 5.16 in the textbook. Each spin lock has a private variable named “value” to hold the status of the lock: 0 => FREE, 1 => BUSY. (4 pts.)

```cpp
void acquire()
```
22. Under what condition does the textbook say that spin locks make sense? (3 pts.)

23. Give the four necessary conditions for deadlock. (8 pts.)

24. Identify at least one deadlock prevention approach. (2 pts.)

25. What is the rationale for a detect-and-recover approach to deadlock versus prevention or avoidance? (3 pts.)

True/False. Circle only one of T or F. (2 pts. each)
Use this table for questions 26 to 28 for deadlock avoidance questions based on the Banker’s algorithm. Treat each question independently, starting each time from the values shown in the table.

<table>
<thead>
<tr>
<th>process</th>
<th>max need</th>
<th>allocated</th>
<th>remaining need</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>8</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>7</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>C</td>
<td>5</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

Unused units = 3

26. T / F Can safely grant request of process A for 2 units.
27. T / F Can safely grant request of process B for 2 units.
28. T / F Can safely grant request of process C for 2 units.
29. Given the following list of tasks, arrival times, and service times, calculate the completion (i.e., departure) time and response (i.e., turnaround) time of each task for FIFO scheduling and again for SJF-preemptive scheduling. (16 pts.)

<table>
<thead>
<tr>
<th>Task</th>
<th>Arrival Time</th>
<th>Service Time</th>
<th>FIFO Completion Time</th>
<th>FIFO Response Time</th>
<th>SJF-preemptive Completion Time</th>
<th>SJF-preemptive Response Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>4</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>6</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>12</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**FIFO/RR/MFQ/SJF-preemptive.** Circle one or more of F, R, M, S, as applies. (2 pts. each)

30. F / R / M / S Uses time slices.
31. F / R / M / S Is non-preemptive.
32. F / R / M / S Requires future knowledge.
33. F / R / M / S Is the simplest to implement.
34. Fully describe the workload in which RR results in optimal (i.e., best case) scheduling. (3 pts.)

35. Let the three threads shown below run with static scheduling priority on a uniprocessor. The lock being used is a queueing lock. Explain the sequence of events and actions that results in priority inversion. You can use the statement identifiers to help explain the sequencing, but a list of the statement identifiers by themselves is not sufficient. (6 pts.)

<table>
<thead>
<tr>
<th>low priority thread A</th>
<th>medium-priority thread B</th>
<th>high-priority thread C</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1: lock.acquire();</td>
<td>B1: work</td>
<td>C1: lock.acquire();</td>
</tr>
<tr>
<td>A2: critical section</td>
<td>B2:</td>
<td>C2: critical section</td>
</tr>
<tr>
<td>A3: lock.release();</td>
<td></td>
<td>C3: lock.release();</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

36. Identify and explain at least one reason why a system might have to do more work per request as the load increases. (2 pts.)