CS365 Operating System Final Exam

Name: Tyler Bartlett

Question Number	Score
1	
2	
3	
4	
5	
6	
Total:	

Question1:
Circle true or false: (total 23 pts, 1pt each)

T	1. Many operating system merge I/O devices and files into a combined file because of the similarity of system calls for each.
F	2. System calls can be run in either user mode or kernel mode.
F	3. The difference between a program and a process is that a program is an active entity while a process is a passive entity.
F	4. Java's RMI is a feature similar to RPCs.
T	5. Each thread has its own register set and stack.
F	6. The single benefit of a thread pool is to control the number of threads.
T	7. Virtually all contemporary operating systems support kernel threads.
F	8. Linux distinguishes between processes and threads.
T	9. Race conditions are prevented by requiring that critical regions be protected by locks.
F	10. The value of a counting semaphore can range only between 0 and 1.
F	11. A deadlock-free solution eliminates the possibility of starvation
F	12. The local variables of a monitor can be accessed by only the local procedures.
F	13. Monitors are a theoretical concept and are not practiced in modern programming languages.
T	14. Mutex locks and binary semaphores are essentially the same thing.
F	15. Mutex locks and counting semaphores are essentially the same thing.
T	16. The circular-wait condition for a deadlock implies the hold-and-wait condition.
F	17. If a resource-allocation graph has a cycle, the system must be in a deadlocked state.
F	18. Protocols to prevent hold-and-wait conditions typically also prevent starvation.

T	19. The wait-for graph scheme is not applicable to a resource allocation system with multiple instances of each resource type.
F	20. Ordering resources and requiring the resources to be acquired in order prevents the circular wait from occurring and therefore prevents deadlock from occurring.
T	21. The banker's algorithm is useful in a system with multiple instances of each resource type.
F	22. A system in an unsafe state will ultimately deadlock.
F	23. Deadlock prevention and deadlock avoidance are essentially the same approaches for handling deadlock.

Question2:
Explain the following terminology: (total 8pts, 1pt each)

1. **Preemptive Scheduling and Non-preemptive Scheduling**
   Preemptive*:* Processes are run based on level of importance. The processor may be forced to give up CPU for higher priority job.

   Nonpreemptive*:* Whatever process is being allocated CPU is allowed to finish. The processing will switch from running to waiting, from running to ready, or from waiting to ready by sleep, yield, or interrupt respectively.

2. **Dispatch latency**
   The time it takes for the dispatcher to stop one process and start running another.

3. **Turnaround time, waiting time, and response time**
   Turn Around Time (TAT): the amount of time to execute a process.Equal to the time of arrival - time of completion.

   Waiting Time: TAT - Burst Time (time taken to execute a process) - time required by a process for execution.

Response Time: time taken from when a request was submitted until the first response is produced.

4. **Spinlock**
   A thread stuck looping to acquire the lock. Results in wasted CPU time but yields better performance. This is mostly not noticeable in modern operating systems.

5. **Mutex**
   Also known as a binary semaphore, unlike spinlock thread will attempt to gain access to lock a couple of times before going to sleep until it's available.

6. **Race Condition**
   Several processes access and manipulate the same data concurrently and the outcome of the execution depends on the particular order in which the access takes place.

7. **Monitor**
   A synchronization tool that allows threads to have mutual exclusion and be able to wait for a condition to be met before continuation.

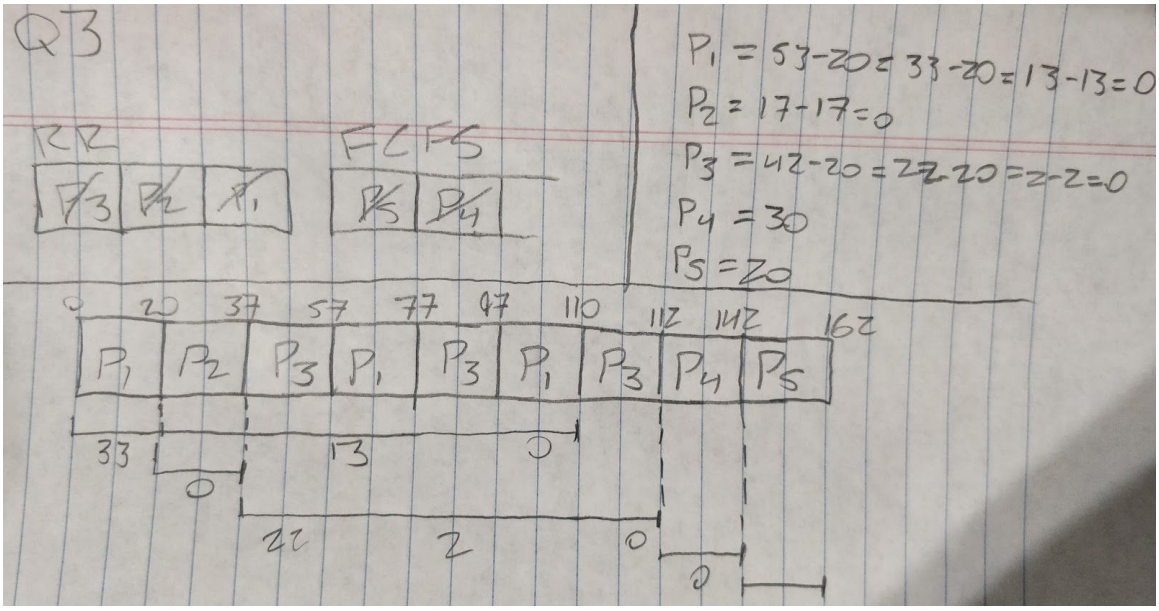
8. **Deadlock**
   Two or more processes are waiting indefinitely for an event that can be caused only by one of the waiting processes.

Question3: (total 10 pts)
By "**Multilevel Queue Fixed priority**" scheduling algorithm, draw the CPU scheduling Gantt chart and complete the table for the give processes information.

	Process	Burst time	Algorithm
■ Foreground	P1	53	(RR interval:20)
	P2	17	
	P3	42	
■ Background	P4	30	(FCFS)
	P5	20	

Gantt Chart:

end time is 162.



	P1	P2	P3	P4	P5
Waiting time	57	20	70	112	142
Turnaround time	110	37	112	142	162
Response time	0	20	37	112	142

Question4: (total 20 pts)
1. A solution to the critical-section problem must satisfy which **three** requirements? (3pts)

Mutual Exclusion: when a thread is executing in its critical section, no other threads can be executing in their critical sections.

Progress: If no thread is executing in its critical section and if there are some threads that wish to enter their critical sections, then one of these threads will get into the critical section.

Bounded waiting: after a thread makes a request to enter its critical section, there is a bound on the number of times that other threads are allowed to enter their critical sections, before the request is granted.

2. We consider a system consisting of two processes, P0 and P1, each accessing two semaphores, S and Q, set to the value 1.

P0	P1
Wait(S)	Wait(S)
Wait(Q)	Wait(Q)
…	…
…	…
…	…
Signal(S)	Signal(Q)
Signal(S)	Signal(S)

What kind of unwanted **situation(s)** will happen? Explain your answer. (3pts)
Both processes are waiting for the "ok-to-go" signal from the other process which results in a deadlock because they're both trying to signal S.

3. We consider a system consisting of two processes, P0 and P1, each accessing two semaphores, S and Q, set to the value 1.

P0	P1
Wait(S)	Wait(S)
Wait(Q)	Wait(Q)
…	…
…	…
…	…
Signal(S)	Signal(Q)
Signal(Q)	Signal(S)

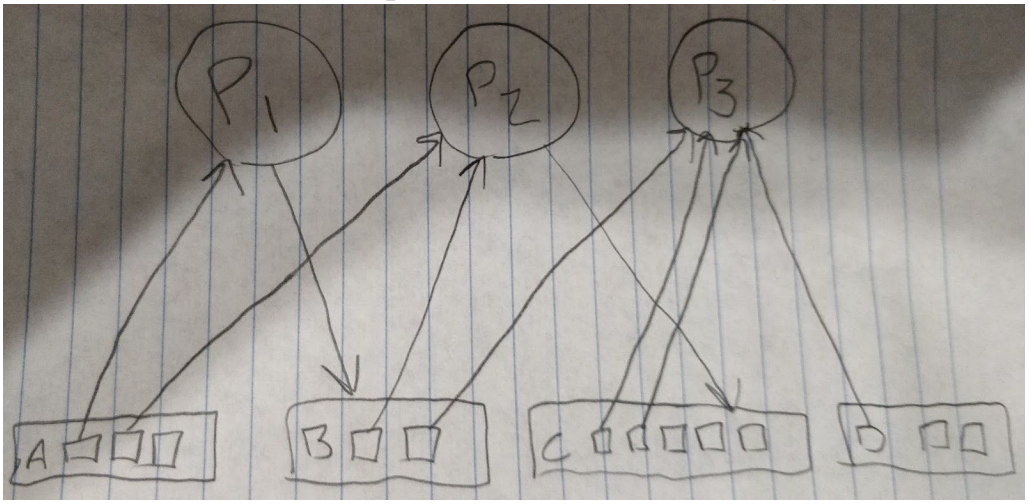
Does this have the chance to occur deadlock situation? Explain your answer? (3pts)

Yes there is a chance but its not likely to occur. There is a circular wait between both processes when one waits while the other goes.

4.

	Allocation				Request				Available			
	A	B	C	D	A	B	C	D	A	B	C	D
P1	1	0	0	0	0	1	0	0	1	0	3	1
P2	1	1	0	0	0	0	1	0				
P3	0	1	2	1	0	0	0	0				

1) Draw the resource-allocation graph under the above situation. (3pts)
2) Is there a cycle? (1pt)
3) Is there a deadlock? (1pt) There is no cycle or deadlock.



5. Assume there are 5 processes P_0 through P_4;
   3 resource types:
   A (10 instances), B (5instances), and C (7 instances)
   Snapshot at time T_0:

	Allocation	*Max*	*Available*
	A B C	*A B C*	*A B C*
P_0	0 1 0	7 5 3	3 3 2
P_1	2 0 0	3 2 2	
P_2	3 0 2	9 0 2	
P_3	2 1 1	2 2 2	
P_4	0 0 2	4 3 3	

Use banker's algorithm to check (1) if the request (1,0,2) by $\boldsymbol{P_1}$ can be granted? (3 pts) (2) Can request for (0,2,0) by $\boldsymbol{P_0}$ be granted? (3 pts) Explain your answer.

1. Yes the request (1,0,2) by $\boldsymbol{P_1}$ can be granted and results in the order: ($\boldsymbol{P_1}, \boldsymbol{P_3}, \boldsymbol{P_4}, \boldsymbol{P_0}, \boldsymbol{P_2}$).

2. No, request for (0,2,0) by $\boldsymbol{P_0}$ cannot be granted because the allocation for $\boldsymbol{P_0}$ is not less than is needed.

Question5:

1. Give two examples of applications for which circuit switching and packet switching would be more suitable respectively. (4 points)

Circuit switching:
SKIP PER YANWEI EMAIL

Packet switching:
SKIP PER YANWEI EMAIL

2. Routers communicate using point-to-point instead of broadcast. That is, the broadcast is stopped at the gateway. Why would it be a bad idea for gateways to pass broadcast packets between networks? What would be the advantages of doing so? (2 points)

SKIP PER YANWEI EMAIL

Question 6:

Circle the best answer. (20 pts, 1 pt for each)

1. A _____ is an example of a systems program.
A) command interpreter
B) Web browser
C) text formatter
D) database system

2. A message-passing model is ____.
A) easier to implement than a shared memory model for intercomputer communication
B) faster than the shared memory model
C) a network protocol, and does not apply to operating systems
D) only useful for small simple operating systems

3. The ____ of a process contains temporary data such as function parameters, return addresses, and local variables.
A) text section
B) data section
C) program counter
D) stack

4. The list of processes waiting for a particular I/O device is called a(n) ____.
A) standby queue
B) device queue
C) ready queue
D) interrupt queue

5. A process may transition to the Ready state by which of the following actions?
A) Completion of an I/O event
B) Awaiting its turn on the CPU
C) Newly-admitted process
D) All of the above

6. Which of the following is true of cooperative scheduling?
A) It requires a timer.
B) A process keeps the CPU until it releases the CPU either by terminating or by switching to the waiting state.
C) It incurs a cost associated with access to shared data.
D) A process switches from the running state to the ready state when an interrupt occurs.

7. ____ is the number of processes that are completed per time unit.
A) CPU utilization
B) Response time
C) Turnaround time
D) Throughput

8. ____ scheduling is approximated by predicting the next CPU burst with an exponential average of the measured lengths of previous CPU bursts.
A) Multilevel queue
B) RR
C) FCFS
D) SJF

9. The ____ scheduling algorithm is designed especially for time-sharing systems.
A) SJF
B) FCFS
C) RR
D) Multilevel queue

10. Which of the following scheduling algorithms must be nonpreemptive?
A) SJF
B) RR
C) FCFS
D) priority algorithms

11. Which of the following is true of multilevel queue scheduling?
A) Processes can move between queues.
B) Each queue has its own scheduling algorithm.
C) A queue cannot have absolute priority over lower-priority queues.
D) It is the most general CPU-scheduling algorithm.

12. _____ allows a thread to run on only one processor.
A) Processor affinity
B) Processor set
C) NUMA
D) Load balancing

13. _____ is a technique for handling critical sections in operating systems.
A) Nonpreemptive kernels
B) Preemptive kernels
C) Spinlocks
D) Peterson's solution

14. A semaphore ____.
A) is essentially an integer variable
B) is accessed through only one standard operation
C) can be modified simultaneously by multiple threads
D) cannot be used to control access to a thread's critical sections

15. A(n) _____ refers to where a process is accessing/updating shared data.
A) critical section
B) entry section
C) mutex
D) test-and-set

16. _____ can be used to prevent busy waiting when implementing a semaphore.
A) Spinlocks
B) Waiting queues
C) Mutex lock
D) Allowing the wait() operation to succeed

17. A deadlocked state occurs whenever ____.
A) a process is waiting for I/O to a device that does not exist
B) the system has no available free resources
C) every process in a set is waiting for an event that can only be caused by another process in the set
D) a process is unable to release its request for a resource after use

18. One necessary condition for deadlock is ____, which states that at least one resource must be held in a nonsharable mode.
A) hold and wait
B) mutual exclusion
C) circular wait
D) no preemption

19. In a system resource-allocation graph, ____.
A) a directed edge from a process to a resource is called an assignment edge
B) a directed edge from a resource to a process is called a request edge
C) a directed edge from a process to a resource is called a request edge
D) None of the above

20. A cycle in a resource-allocation graph is ____.
A) a necessary and sufficient condition for deadlock in the case that each resource has more than one instance
B) a necessary and sufficient condition for a deadlock in the case that each resource has exactly one instance
C) a sufficient condition for a deadlock in the case that each resource has more than once instance
D) is neither necessary nor sufficient for indicating deadlock in the case that each resource has exactly one instance