## OS Quiz 1

* **Q1. What is an advantage and disadvantage of many-to-many mappings of logical (user) to physical(kernel) threads?** a. High Concurrency and mapping is very simple many kernel threads would be required b. High Concurrency and not many kernel threads required, mapping is very complex c. Mapping is very simple and not many kernel threads required, concurrency is low d. High Concurrency, mapping is very complex and many kernel threads would be required

**b. High Concurrency and not many kernel threads required, mapping is very complex**

* **Q2. What is the difference between shared memory and message passing based Inter-Process Communication(IPC)** a. They are two different names for the same communication mechanism which uses kernel memory space b. In message passing, the communication is handed by the OS. In shared memory, the communication is handed directly by the processes through a common memory space c. In shared memory, there is memory in the kernel space that is shared between the processes. In message passing, no memory is required in the kernel memory space d. They are two different names for the same communication mechanism which uses user memory space

**b. In message passing, the communication is handed by the OS. In shared memory, the communication is handed directly by the processes through a common memory space**

* **Q3. Describe one advantage and one disadvantage of batch systems** a. Memory management is simple (1 user job). CPU utilization is no efficient (scheduling of jobs is complex) b. Memory management is simple (1 user job). CPU utilization is not efficient (idling during I/O) c. CPU utilization is very efficient ( 1 user job to schedule). Memory management is complex (deciding which user job to keep in memory)

**b. Memory management is simple (1 user job). CPU utilization is not efficient (idling during I/O)**

* **Q4. DMA does not use interrupts** False: DMA uses interrupts once the block transfer is complete (between memory and I/O device) to notify the CPU True: DMA bypasses the CPU and hence does not require interrupts

**False: DMA uses interrupts once the block transfer is complete (between memory and I/O device) to notify the CPU**

* **Q5. Kernel or monitor or supervisor mode is the same as "superuser" in Linux** a. False: Kernel mode is a hardware mode of operation, whereas "superuser" is the "root" user account b. True: Both, the kernel mode as well as "superuser" allows one to execute privileged instructions in hardware c. False: "superuser" is more privileged than kernel mode d. False: "superuser" is the same as supervisor mode, whereas kernel or monitor mode is more privileged

**a. False: Kernel mode is a hardware mode of operation, whereas "superuser" is the "root" user account**

* **Q6. In the process state transition diagram, what is a timer interrupt?** a. It is an hardware interrupt generated by a hardware/software clock that tracks progress of physical time b. It is an interrupt generated by the OS to denote process completion c. It is a trap interrupt generated by the CPU d. None of the options are correct

**a. It is an hardware interrupt generated by a hardware/software clock that tracks progress of physical time**

* **Q7. There is one base and one limit register in the hardware for each user program** a. True: This is necessary to ensure memory protection for each user program b. False: There is only one base and one limit register overall in the hardware, which are loaded with appropriate values by the **OS** before executing the user program c. False: There is only one base and one limit register overall in the hardware, which are loaded with appropriate values by the **CPU** before executing the user program d. False: The registers are in the OS, which also checks whether each memory access is within range

**b. False: There is only one base and one limit register overall in the hardware, which are loaded with appropriate values by the OS before executing the user program**

* **Q8. What comprises a process context and where is it stored?** a. All registers required to execute/resume the process (program counter, stack pointer, memory protection registers, etc) and it is stored in the PCB b. The process memory is defined by the base and limit register values and it is stored in the user memory space c. All registers required to resume the process (program counter, stack pointer, memory protection registers, etc) and it is stored in the user memory space d. Only base and limit register values for a process and it is stored in the PCB

**a. All registers required to execute/resume the process (program counter, stack pointer, memory protection registers, etc) and it is stored in the PCB (which is stored in the kernel memory space)**

* **Q9. How do you distinguish between direct and indirect message passing for inter-process communication?** a. In direct message passing, communication occurs between specific processes; In indirect message passing, any process with access to a mailbox can send/receive messages to/from that mailbox. b. In direct message passing, communication occurs between specific processes through shared user space memory; In indirect message passing, communication occurs through mailboxes. c. Direct message passing is the same shared memory; Indirect message passing uses the kernel memory space.

**a. In direct message passing, communication occurs between specific processes; In indirect message passing, any process with access to a mailbox can send/receive messages to/from that mailbox.**

* **Q10. What part of the process memory space are shared between threads in the same process** a. Everything is shared between threads in a single process. b. Only Text and OS resources (files, etc.) are shared; all other parts are private to the threads. c. Text, Data, Heap and OS resources (files, etc.) are shared between the threads; Stack and Registers are not. d. Text, Data and Heap are shared between the threads; OS resources (files, etc.), Stack and Registers are not.

**c. Text, Data, Heap and OS resources (files, etc.) are shared between the threads; Stack and Registers are not.**

* **Q11. Which of the following precisely describes the sequence of steps to process an interrupt** a. Disable interrupts, locate interrupt service routine (ISR) in interrupt vector table, execute ISR, enable interrupts. b. Switch to kernel mode, save program context, locate interrupt service routine (ISR) in interrupt vector table, execute ISR, restore program context, switch to user mode. c. Switch to kernel mode, locate interrupt service routine (ISR) in interrupt vector table, execute ISR, switch to user mode. d. Disable interrupts, save program context, locate interrupt service routine (ISR) in interrupt vector table, execute ISR, restore program context, enable interrupts.

**b. Switch to kernel mode, save program context, locate interrupt service routine (ISR) in interrupt vector table, execute ISR, restore program context, switch to user mode.**

* **Q12. What is the difference between multi-threading, multi-programming and multi-processing?** a. They are all different terms for the same concept of time-multiplexed execution. b. multi-threading and multi-programming are identical; multi-processing means multiple CPU cores for execution. c. multi-threading means multiple threads of execution within a process; multi-programming and multi-processing are identical. d. multiple threads of execution within a process; multiple processes ready for execution in main memory; multiple CPU cores for execution.

**d. multiple threads of execution within a process; multiple processes ready for execution in main memory; multiple CPU cores for execution.**

* **Q13. What is the difference between a process and a program?** a. They are identical b. Every user program can have at most one active process in the system. c. They are two different and unrelated concepts. d. Process is an instance of a program that is currently active in the system

**d. Process is an instance of a program that is currently active in the system**

* **Q14. What is the difference between a long-term and medium-term scheduler?** a. Long-term scheduler schedules processes in main memory on the CPU; Medium-term scheduler decides which active processes to keep in main memory and which ones to swap to disk when load is heavy. b. They are identical. c. Long-term scheduler decides which new processes should be brought to main memory from disk; Medium-term scheduler schedules processes in main memory on the CPU. d. Long-term scheduler decides which new processes should be brought to main memory from disk; Medium-term scheduler decides which active processes to keep in main memory and which ones to swap to disk when load is heavy.

**d. Long-term scheduler decides which new processes should be brought to main memory from disk; Medium-term scheduler decides which active processes to keep in main memory and which ones to swap to disk when load is heavy.**

* **Q15. What is an advantage and a disadvantage of one-to-one mapping of logical (user) and physical (kernel) threads?** a. More concurrency than many-to-one mappings; mapping is very complex. b. More concurrency than many-to-many mappings; requires creation of several kernel threads (one per user thread). c. More concurrency than many-to-one mappings; requires creation of several kernel threads (one per user thread). d. Mapping is very simple; less concurrency than many-to-one mappings.

**c. More concurrency than many-to-one mappings; requires creation of several kernel threads (one per user thread).**

* **Q16. What is an Operating System?** a. Software program that controls other user programs. b. Allocates and manages hardware resources. c. Always ready to accept new commands from users and hardware. d. All the other options are correct.

**d. All the other options are correct.**

* **Q17. Trap is an interrupt generated only due to software exception** a. True: CPU transfers control to the OS to process the exception b. False: CPU generates trap either due to software exception or system call, and transfers control to the OS c. False: OS generates trap due to software exceptions and transfers control to the CPU d. False: OS generates trap and processes it

**b. False: CPU generates trap either due to software exception or system call, and transfers control to the OS**

* **Q18. Why is there no transition from the waiting state to the running state?** a. Because a process in the waiting state is unable to execute even if it is given access to the CPU b. Because once a process in the waiting state is ready to execute, it would first move to the ready state and then to the running state when schduled for execution by the OS c. Because this allows the OS to methodically select the process to execute from among all processes that are ready to execute on the CPU d. All the other options are correct

**d. All the other options are correct**

* **Q19. What are the two typical execution orders in fork() system call** a. Forked child processes wait for parent process to terminate; Parent process can wait for the forked child processes to terminate before proceeding b. Parent process can execute in parallel to the forked child processes; Parent process can wait for the forked child processes to terminate before proceeding c. Parent process executes before forked child processes; Forked child processes execute before parent process can continue

**b. Parent process can execute in parallel to the forked child processes; Parent process can wait for the forked child processes to terminate before proceeding**

* **Q20. System calls are identical to library function calls in a high-level programming language such as C** a. True: For example, fopen() in C programming is a system call to open a file b. False: User programs can access services in the OS by invoking software functions implemented in the OS through system calls

**b. False: User programs can access services in the OS by invoking software functions implemented in the OS through system calls**

* **Q21. What are the four memory regions of a process and how they are used** a. "Text" for documentation about the process; "Data" for all variables; "Stack" for managing function calls; "Heap" for garbage collection b. "Code" for instruction; "Global" for static and global variable; "Function" for managing function calls and local function variables; "Dynamic" for dynamically created variables c. "Text" for instructions; "Data" for static and global variables; "Stack" for managing function calls and local function variables; "Heap" for dynamically created variables d. All the other options are correct

**c. "Text" for instructions; "Data" for static and global variables; "Stack" for managing function calls and local function variables; "Heap" for dynamically created variables**

* **Q22. What is a Process Control Block stored and who can access it?** a. In the user memory space and can be accessed by the OS as well as the process b. In the kernel memory space and can be accessible by the OS as well as the process c. In the kernel memory space and can be accessed only by the OS d. In the user memory space and can be accessed by only the OS

**c. In the kernel memory space and can be accessed only by the OS**

* **Q23. What is the disadvantage of many-to-one mapping between logical (user) and physical (kernel) threads?** a. The mapping process is very complex b. A blocked user thread will block all the other user threads that are mapped to the same kernel thread c. There is no disadvantage d. A blocked user thread will also block all the other kernel threads in the system

**b. A blocked user thread will block all the other user threads that are mapped to the same**

* **Q24. Multiprocessing without multiprogramming is efficient?** a. Yes, they are different techniques that can applied independently and all combinations are efficient b. Yes, even without multiprogramming, multiprocessing will ensure high CPU utilization c. No, without multiprogramming there will be only one user job to execute and hence CPU utilization will be low

**c. No, without multiprogramming there will be only one user job to execute and hence CPU utilization will be low**

* **Q25. In real-time systems, average response time of user programs should be minimized** a. True: This will ensure that the required deadlines are met b. True: This will ensure that the system is responsive c. False: Real-time systems require guaranteed satisfaction of deadlines, irrespective of average response time d. False: Real-time systems require average response time to be maximized

**c. False: Real-time systems require guaranteed satisfaction of deadlines, irrespective of average response time**

* **Q26. The following sequence of steps precisely defines the processing of a system call** a. OS generates a trap, OS switches to kernel mode, OS identifies the appropriate function for the system call and executes it, upon function completion OS switches the mode back to user mode and transfers control to the user program b. CPU generates a trap, CPU switches to kernel mode, CPU identifies the appropriate function for the system call and executes it, upon function completion CPU switches the mode back to user mode and transfers control to the user program c. CPU generates a trap, CPU switches to kernel mode and transfers control to the OS, OS identifies the appropriate function for the system call and executes it, upon function completion OS switches the mode back to user mode and transfers control to the user program d. CPU generates a trap, CPU switches to kernel mode and transfers control to the OS, OS identifies the appropriate function for the system call and executes it, upon function completion CPU switches the mode back to user mode and transfers control to the user program

**c. CPU generates a trap, CPU switches to kernel mode and transfers control to the OS, OS identifies the appropriate function for the system call and executes it, upon function completion OS switches the mode back to user mode and transfers control to the user program**

* **Q27. When a user program tries to execute an I/O instruction directly** a. the CPU executes the instruction b. the CPU checks if the request is within range of the base and limit registers c. the CPU checks for its validity and legality before executing it d. the CPU generates a trap and transfers control to the OS

**d. the CPU generates a trap and transfers control to the OS**

## ****OS Quiz 2****

* **Q1. Describe one advantage and one disadvantage of Shortest-Job First (SJF) scheduling** a. efficient way to handle convoy effect; not implementable due to lack of information on CPU burst lengths b. optimal in terms of maximizing average response time of all processes; not implementable due to lack of information on CPU burst lengths c. optimal in terms of minimizing average response time of all processes; not implementable due to lack of information on I/O burst lengths d. optimal in terms of maximizing average response time of all processes not implementable due to lack of information on I/O burst lengths

**a. efficient way to handle convoy effect; not implementable due to lack of information on CPU burst lengths**

* **Q2. Under Round-Robin scheduling, if quantum size is q, average CPU burst length is B, average number of CPU bursts per process is N, and average number of processes in the ready queue is R, then the average response time for a process is?** a. (R-1) x q b. q x (B x N x R) c. (B x N x R)/q d. (N-1) x q

**a. (R-1) x q**

* **Q3. The difference between non-preemptive Shortest-Job First (SJF) and Shortest Remaining Time First (SRTF) is that** a. SJF is unaffected by newly admitted processes when a process is "running"; In SRTF if the current CPU burst of the newly admitted process is shorter than the remaining CPU burst of the running process, then a context-switch is triggered b. SRTF is unaffected by newly admitted processes when a process is "running"; In SJF if the total CPU duration of the newly admitted process is shorter than the remaining CPU duration of the running process, then a context-switch is triggered c. SJF is unaffected by newly admitted processes when a process is "running"; In SRTF if the total CPU and I/O duration of the newly admitted process is shorter than the remaining total CPU and I/O duration of the running process, then a context-switch is triggered d. SRTF allows random preemption of processes, whereas SJF does not

**a. SJF is unaffected by newly admitted processes when a process is "running"; In SRTF if the current CPU burst of the newly admitted process is shorter than the remaining CPU burst of the running process, then a context-switch is triggered**

* **Q4. What is an I/O burst?** a. Time taken by all the I/O system calls executed by a process b. Time taken by a single I/O system call executed by a process c. Time taken by the I/O device controller to process a request

**b. Time taken by a single I/O system call executed by a process**

* **Q5.** **In a preemptive CPU scheduler, when does scheduling happen?** a. Upon any of the five transitions (in the process state transition diagram) b. Upon any of the five transitions (in the process state transition diagram) and even at other time instants c. Upon transitions 1 and 5 and ocassionally upon transition 4 (in the process state transition diagram) d. Upon transitions 2, 3 and 4 (in the process state transition diagram)

**b. Upon any of the five transitions (in the process state transition diagram) and even at other time instants**

* **Q6. What is the need for multi-level queue scheduling?** a. Processes with different requirements can be mapped to different queues and each queue can have a different scheduling policy (e.g., RR for interactive processes and FCFS for background) b. This is necessary to minimize the average waiting time of all processes c. Processes with different requirements can be mapped to different queues and each queue can have a different scheduling policy (e.g., FCFS for interactive processes and RR for background) d. It is only useful for multiprocessing (i.e, CPU with multiple cores)

**a. Processes with different requirements can be mapped to different queues and each queue can have a different scheduling policy (e.g., RR for interactive processes and FCFS for background)**

* **Q7. What is a CPU burst?** a. Time taken by a single instruction executed by a process b. Time taken by a set of instruction executed by a process between two successive I/O requests c. Time taken by all the instructions executed by a process

**b. Time taken by a set of instruction executed by a process between two successove I/O requests**

* **Q8. Under global multiprocessor scheduling, implementation overhead is high because:** a. The overhead incurred by an OS to copy the CPU registers from one core to another when a process migrates is very high b. The overhead incurred by a process when it executes on one core, is then preempted, and subsequently continues to execute on the same core, is very high. The overhead is due to information in the private core-specific cache that is no longer accessible c. The overhead incurred by a process when it partially executes on one core and is subsequently migrated to another core is very high. The overhead is due to information in the private core-specific cache that is no longer accessible d. None of the other answers are correct

**c. The overhead incurred by a process when it partially executes on one core and is subsequently migrated to another core is very high. The overhead is due to information in the private core-specific cache that is no longer accessible**

* **Q9. Turnaround Time is defined as:** a. Time taken between transition 4 and transition 5(in the process state transition diagram) b. Time taken between transition 2 and transition 5(in the process state transition diagram) c. Time taken between transition 4 and the first occurence of the transition from "ready" to "running" (in the process state transition diagram) d. Time taken between transition 2 and the next occurence of the transtition from "ready" to "running"(in the process state transition diagram)

**a. Time taken between transition 4 and transition 5(in the process state transition diagram)**

transition 4 is from new to ready and transition 5 is from running to terminated

* **Q10. Shortest-Job-First(SJF) is optimal in the sense that it** a. Minimizes the average waiting time for all processes b. Maximizes the average waiting time for all processes c. Minimizes the average response time for all processes d. Maximizes CPU utilization

**a. Minimizes the average waiting time for all processes**

* **Q11. Round-Robin (RR) scheduling has lower average waiting time for processes when compared to Shortest Remaining Time First (SRTF)** a. False, SRTF has the lowest average waiting time among all preemptive scheduling algorithms b. False, but it has lower average turnaround time than SRTF c. True, because RR has the lowest average waiting time among all preemptive scheduling algorithms d. False, RR has the lowest average response time among all preemptive scheduling algorithms

**a. False, SRTF has the lowest average waiting time among all preemptive scheduling algorithms**

* **Q12. Under First-Come-First-Served (FCFS) scheduling what is convoy effect?** a. A long process is in the "running" state, while several short processes are waiting in the "ready" state b. A long process is in the "running" state, while several short processes are waiting in the "waiting" state c. A long process is in the "ready" state, while several short processes are waiting in the "ready" state behing this long process

**a. A long process is in the "running" state, while several short processes are waiting in the "ready" state**

* **Q13. In a nonpreemptive CPU scheduler, when does scheduling happen?** a. Upon transition 1 and 5 and occassionally upon transition 5 (in the process state transition diagram) b. Upon transition 2, 3, 4 (in the process state transition diagram) c. Upon any of the five transitions (in the process state transition diagram) d. Upon any of the five transition (in the process state transition diagram) and even at other time instants

**a. Upon transition 1 and 5 and occassionally upon transition 5 (in the process state transition diagram)**

* **Q14. Under global multiprocessor scheduling, a process may execute on two cores at the same time** a. False, a process can only execute on one core at a time. Under global scheduling, it may migrate between cores over time b. False, a process can only execute on one core at any time. Under global scheduling, a process is assigned to a core when it arrives in the system and is always executed only on that core c. True, under global scheduling, a process may migrate between cores over time and execute on them in parallel d. True, under global scheduling, a process may execute on two cores in parallel, but never on three or more cores in parallel

**a. False, a process can only execute on one core at a time. Under global scheduling, it may migrate between cores over time**

* **Q15. The key challenge under partitioned multiprocessor scheduling is?** a. How to schedule processes on individual CPU cores? b. How to do time-synchronization across CPU cores? c. How to map and partition the processes to CPU cores? d. All of the others are key challenges

**c. How to map and partition the processes to CPU cores?**

* **Q16. In First-Come-First-Served (FCFS) scheduling, the CPU scheduler is executed whenever a new process is "admitted" in the system** a. False, it is a nonpreemptive scheduler and hence will only execute when either a "running" process completes or moves to the "waiting" state, or there is no running process when the new process is "admitted" b. False, it is a nonpreemptive scheduler and hence will only execute when either a "running" process completes or moves to the "waiting" or "ready" states c. True, this enables the scheduler to check whether the new process must be given access to the CPU d. True, this enables the scheduler to setup the memory region for the new process

**a. False, it is a nonpreemptive scheduler and hence will only execute when either a "running" process completes or moves to the "waiting" state, or there is no running process when the new process is "admitted"**

* **Q17. Waiting Time is defined as:** a. Time spent by a process in the "ready" state b. Time spent by a process in the "waiting" state c. Time taken between transition 4 and transition 5 minus the time spent in the "running" state (in the process transition diagram) d. Time spent in the "ready" and "waiting" states combined

**a. Time spent by a process in the "ready" state**

* **Q18. Multi-level queue scheduling can be either preemptive or non-preemptive, but not both** a. False, it can be both (e.g. one queue can use RR while the other can use FCFS) b. False, it can be both (e.g. one queue can use RR while the other can use SRTF) c. True, e.g. if one queue use RR, the others can use SRTF but not SJF or FCFS

**a. False, it can be both (e.g. one queue can use RR while the other can use FCFS)**

* **Q19. Turnaround time of processes decreases as the quantum size in Round-Robin (RR) scheduling increases** a. False, there is no direct correlation between turnaround time and quantum size b. False, it increases as the quantum size increases c. False, it is always fixed independent of the quantum sizes d. True, and further it remains fixed once the quantum size is larger than the maximum CPU burst length

**a. False, there is no direct correlation between turnaround time and quantum size**

* **Q20. What is Aging?** a. A technique in which the priority of processes that are unable to execute is slowly increased over time to avoid starvation b. A technique in which the priority of processes that are unable to execute is slowly decreased over time to avoid starvation c. A technique in which the priority of all processes is slowly increased over time to avoid starvation

**a. A technique in which the priority of processes that are unable to execute is slowly increased over time to avoid starvation**

* **Q21. Response Time is defined as:** a. Time taken for a process between transition 4 and first occurrence of transition from "ready" to "running" (in the process state transition diagram) b. Time taken for a process between 2 and the next occurrence of transition from "ready" to "running" (in the process state transition diagram) c. Time taken for a process between transition 4 and transition 5 (in the process state transition diagram) d. Time taken for a process between transition 3 and the next occurrence of transition from "ready" to "running" (in the process state transition diagram)

**a. Time taken for a process between transition 4 and first occurrence of transition from "ready" to "running" (in the process state transition diagram)**

## OS Quiz 3

* **Q1. What is meant by the mutual exclusion property?** a) No two processes can be in the critical section at the same time b) At most two processes can be in the critical section at the same time c) No process can context switch in the critical section d) The OS cannot context switch from one process in its critical section to another process in its critical section

**a) No two processes can be in the critical section at the same time**

* **Q2. What is meant by the Progress property?** a) It is liveness; when no process is in the critical section, then a process cannot wait indefinitely to get access to the critical section. b) It is fairness; when a process wants access to the critical section, another process cannot repeatedly get access to the critical section indefinitely. c) It is liveness; when one process is in the critical section, another process cannot wait indefinitely to get access to the critical section.

**a) It is liveness; when no process is in the critical section, then a process cannot wait indefinitely to get access to the critical section.**

* **Q3. What is meant by the property of Bounded Waiting?** a) It is fairness; if a process wants to get access to a critical section, another process cannot get access to the critical section repeatedly infinite number of times. b) It is liveness; if no process is in the critical section, then a process wanting to get access to the critical section should not wait indefinitely. c) It is fairness; processes should make progress collectively through their critical sections.

**a) It is fairness; if a process wants to get access to a critical section, another process cannot get access to the critical section repeatedly infinite number of times.**

* **Q4. In Algorithm 1 presented in the lecture, progress is violated because** a) Process P0 executes the critical section once, enters a long remainder section, context switch occurs to process P1, process P1 is waiting to get access to the critical section indefinitely. b) All of the other options are correct. c) Process P0 executes the critical section once, enters a short remainder section, context switch occurs to process P1, process P1 is waiting to get access to the critical section indefinitely. d) Process P1 executes the critical section once, enters a long remainder section, context switch occurs to process P0, process P0 executes the critical section once and then waits indefinitely to get access to the critical section again.

**d) Process P1 executes the critical section once, enters a long remainder section, context switch occurs to process P0, process P0 executes the critical section once and then waits indefinitely to get access to the critical section again.**

* **Q5. In Algorithm 2 in the lecture, Progress property is violated because** a) Process P0 updates flag to true, context switch to Process P1, Process P1 updates flag to true, both the processes are now stuck in the entry while loop. b) Process P0 updates flag to false, context switch to Process P1, Process P1 updates flag to false, both the processes are now stuck in the entry while loop. c) Process P0 updates flag to true and enters critical section, context switch to Process P1, Process P1 updates flag to true and is now stuck indefinitely.

**a) Process P0 updates flag to true, context switch to Process P1, Process P1 updates flag to true, both the processes are now stuck in the entry while loop.**

* **Q6. In Algorithm 2 in the lecture, Bounded Waiting is satisfied because** a) A process sets its flag to true at the beginning of entry section; once the flag is set to true another process cannot subsequently enter its critical section. b) Progress is not satisfied and this implies that Bounded Waiting will be satisfied. c) Process P0 starts and enters critical section, context switch to Process P1, Process P1 sets flag to true and gets stuck in the entry while loop, context switch to Process P0, Process P0 completes critical and remainder sections and tries to enter the critical section again, Process P0 gets stuck.

**a) A process sets its flag to true at the beginning of entry section; once the flag is set to true another process cannot subsequently enter its critical section.**

* **Q7. Algorithm 3 in the lecture satisfies Progress property because** a) Suppose Process P0 wants access to the critical section; then flag for Process P0 is true; if Process P1 is in the entry section then either P0 or P1 will get access to the critical section; if Process P1 is in the critical section then Progress is vacuously satisfied; if Process P1 is in the remainder section then Process P0 can get access to the critical section. b) Progress property is satisfied because Bounded Waiting is satisfied. c) Process P0 sets its flag to true, turn to 1, and waits at the entry while loop; context switch to Process P1; Process P1 sets its flag to true, turn to 0, and waits at the entry while loop; context switch to Process P0; Process P0 enters the critical section.

**a) Suppose Process P0 wants access to the critical section; then flag for Process P0 is true; if Process P1 is in the entry section then either P0 or P1 will get access to the critical section; if Process P1 is in the critical section then Progress is vacuously satisfied; if Process P1 is in the remainder section then Process P0 can get access to the critical section.**

* **Q8. Algorithm 3 in the lecture satisfies the Bounded Waiting property because** a) If Process P0 is waiting in the entry while loop, Process P1's flag is true and turn is 1; when Process P1 tries to enter the critical section subsequently it will set turn to 0; Process P0 is no longer blocked. Identical argument applies to Process P1 because the solution is symmetric. b) Process P0 sets its flag to true, turn to 1 and waits in the entry while loop; context switch to Process P1; Process P1 sets its flag to true, turn to 0 and waits in the entry while loop; context switch to Process P0; Process P0 enters its critical section. Identical argument applies to Process P1 because the solution is symmetric. c) Bounded Waiting is satisfied because Progress property is satisfied.

**a) If Process P0 is waiting in the entry while loop, Process P1's flag is true and turn is 1; when Process P1 tries to enter the critical section subsequently it will set turn to 0; Process P0 is no longer blocked. Identical argument applies to Process P1 because the solution is symmetric.**

* **Q9. What are the sequence of atomic operations in the TestAndSet(lock) function call?** a) fetch the current value of lock in a local variable; store the value true in the lock; return the fetched value to the caller. b) fetch the current value of lock in a local variable; store the value false in the lock; return the fetched value to the caller. c) store the value true in the lock; fetch the current value of lock in a local variable; return the fetched value to the caller. d) store the value false in the lock; fetch the current value of lock in a local variable; return the fetched value to the caller.

**a) fetch the current value of lock in a local variable; store the value true in the lock; return the fetched value to the caller.**

* **Q10. Explain how TestAndSet(lock) works in the Entry section?** a) If the call changed the value of lock from false to true, exit the Entry section and enter the Critical section. b) If the call changed the value of lock from true to false, exit the Entry section and enter the Critical section. c) If the call changed the value of lock from false to true or the value remained true before and after the call, exit the Entry section and enter the Critical section. d) If the call changed the value of lock from false to true or the value remained false before and after the call, exit the Entry section and enter the Critical section.

**a) If the call changed the value of lock from false to true, exit the Entry section and enter the Critical section.**

* **Q11. In TestAndSet(lock) based implementation of the Entry section, explain why Mutual Exclusion is successful.** a) A process enters its critical section only when the lock value is changed from false to true by the TestAndSet() call. The lock remains true while the process is in the Critical section. Since the call to TestAndSet() is atomic, any other process would see the lock as true when executing TestAndSet(). b) A process enters its critical section only when the lock value is changed from true to false by the TestAndSet() call. The lock remains true while the process is in the Critical section. Since the call to TestAndSet() is atomic, any other process would see the lock as false when executing TestAndSet(). c) A process enters its critical section only when the lock value is changed from false to true or the value remains unchanged by the TestAndSet() call. The lock remains true while the process is in the Critical section. Since the call to TestAndSet() is atomic, any other process would see the lock as true when executing TestAndSet().

**a) A process enters its critical section only when the lock value is changed from false to true by the TestAndSet() call. The lock remains true while the process is in the Critical section. Since the call to TestAndSet() is atomic, any other process would see the lock as true when executing TestAndSet().**

* **Q12. Explain why Bounded Waiting property is not satisfied in the TestAndSet(lock) implementation of the Entry section.** a) TestAndSet(lock) does not store information about failed attempts to acquire the lock. Hence, a process that attempts to acquire an available lock will be given the lock, irrespective of the number of failed attempts by other processes. b) It is only true when Round Robin scheduling is used as shown in the lecture slide. c) Both the other options are correct.

**a) TestAndSet(lock) does not store information about failed attempts to acquire the lock. Hence, a process that attempts to acquire an available lock will be given the lock, irrespective of the number of failed attempts by other processes.**

* **Q13. What is the main difference between a binary and a counting semaphore?** a) A binary semaphore allows at most one process in the Critical section at any time (mutual exclusion); A counting semaphore allows at most a finite number of processes in the Critical section at any time. b) A binary semaphore allows at most one process in the Critical section at any time (mutual exclusion); A counting semaphore allows an infinite number of processes in the Critical section at any time. c) Two different names for the same mechanism of protecting Critical sections using semaphores.

**a) A binary semaphore allows at most one process in the Critical section at any time (mutual exclusion); A counting semaphore allows at most a finite number of processes in the Critical section at any time.**

* **Q14. Semaphore implementations do not require any special hardware support.** a) False; Wait() and Signal() system calls must be executed atomically, which requires hardware support (either disabling of interrupts or atomic instructions like TestAndSet()) b) True; Wait() and Signal() system calls must be executed atomically, but they do not require any hardware support. c) False; Wait() system call must be executed atomically, which requires hardware support (either disabling of interrupts or atomic instructions like TestAndSet()). However, Signal() system call does not require hardware support.

**a) False; Wait() and Signal() system calls must be executed atomically, which requires hardware support (either disabling of interrupts or atomic instructions like TestAndSet())**

* **Q15. In the blocking implementation of semaphores, a value of -5 indicates that 4 processes are currently blocked on the semaphore.** a) False; it indicates that 5 processes are currently blocked on the semaphore. b) False; it indicates that 5 processes have currently acquired the semaphore. c) True; 4 processes are currently blocked and one process has acquired the semaphore. d) False; it indicates that 6 processes are currently blocked on the semaphore and one process has acquired the semaphore.

**a) False; it indicates that 5 processes are currently blocked on the semaphore.**

* **Q16. In the blocking implementation of Wait(), why can we not decrement the semaphore value after the if() condition check?** a) This would imply a context-switch between the if() condition check and the decrement, which can lead to mutual exclusion violation. b) This would imply a context-switch between the if() condition check and the decrement, which can lead to a race condition for the semaphore value. c) It is possible, but the implementation is less efficient than when the decrement is before the if() condition check.

**a) This would imply a context-switch between the if() condition check and the decrement, which can lead to mutual exclusion violation.**

* **Q17. Why must Wait() and Signal() system calls with blocking implementation be atomic? Give all reasons.** a) To prevent race condition in the updates to the value of the semaphore; To ensure mutual exclusion by guaranteeing that the update to the semaphore value and its checking in the if() condition happen atomically in each system call. b) To ensure mutual exclusion by guaranteeing that the update to the semaphore value and its checking in the if() condition happen atomically in each system call. c) To prevent race condition in the updates to the value of the semaphore. d) They don't have to be atomic. Only busy waiting implementations require atomic Wait() and Signal() system calls.

**a) To prevent race condition in the updates to the value of the semaphore; To ensure mutual exclusion by guaranteeing that the update to the semaphore value and its checking in the if() condition happen atomically in each system call.**

* **Q18. Present one advantage and one disadvantage of busy waiting implementation of semaphores.** a) reduces context-switch overheads; infeasible on single-core CPUs. b) increases context-switch overheads; infeasible on single-core CPUs. c) reduces context-switch overheads; infeasible on multi-core CPUs. d) increases context-switch overheads; infeasible on multi-core CPUs.

**a) reduces context-switch overheads; infeasible on single-core CPUs.**

* **Q19. In the busy waiting implementation of binary semaphores, explain all the atomic operation requirements.** a) The increment and decrement to the semaphore value must be atomic to prevent a **race condition**; the busy waiting while() loop and the decrement to the semaphore value in Wait() must also be atomic to ensure **mutual exclusion**. b) The busy waiting while() loop and the decrement to the semaphore value in Wait() must be atomic to ensure mutual exclusion. c) The increment and decrement to the semaphore value must be atomic to prevent a race condition. d) The increment and decrement to the semaphore value must be atomic to ensure **mutual exclusion**; the busy waiting while() loop and the decrement to the semaphore value in Wait() must also be atomic to prevent a **race condition**.

**a) The increment and decrement to the semaphore value must be atomic to prevent a race condition; the busy waiting while() loop and the decrement to the semaphore value in Wait() must also be atomic to ensure mutual exclusion.**

* **Q20. The busy waiting implemention of semaphores is infeasible only on single-core CPUs.** a) True; because to release the semaphore the busy waiting must be interrupted on a single-core CPU which then leads to the Wait() system call being non-atomic. b) False; it is infeasible even on multi-core CPUs c) False; it is feasible on both single as well as multi-core CPUs. d) True; because to lock the semaphore the busy waiting must be interrupted on a single-core CPU.

**a) True; because to release the semaphore the busy waiting must be interrupted on a single-core CPU which then leads to the Wait() system call being non-atomic.**

* **Q21. In the Producer-Consumer Bounded-Buffer implementation, why is the Wait() on empty or full semaphore done before the Wait() on mutex?** a) Otherwise, it can lead to a deadlock when the buffer is empty/full, because to add/remove items in the buffer the mutex semaphore must be acquired b) Otherwise, it can lead to a deadlock when the buffer is empty/full, because to add/remove items in the buffer the empty/full semaphore must be acquired c) Otherwise, it can lead to a deadlock when the buffer is empty/full, because to add/remove items in the buffer the full/empty semaphore must be acquired.

**a) Otherwise, it can lead to a deadlock when the buffer is empty/full, because to add/remove items in the buffer the mutex semaphore must be acquired**

* **Q22. In the producer-consumer bounded buffer implementation, explain the role of each semaphore.** a) mutex is a binary semaphore for mutually exclusive access to the shared buffer; full is a counting semaphore tracking the number of empty slots in the buffer; empty is a counting semaphore tracking the number of items in the buffer. b) mutex is a binary semaphore for mutually exclusive access to the shared buffer; full is used by the Producer to check if the buffer is full; empty is used by the Consumer to check if the buffer is empty. c) mutex is a binary semaphore for mutually exclusive access to the shared buffer; full is used by the Consumer to check if the buffer is full; empty is used by the Producer to check if the buffer is empty. d) mutex is a binary semaphore for mutually exclusive access to the shared buffer; empty is used by the Producer to check if the buffer is full; full is used by the Consumer to check if the buffer is empty.

**d) mutex is a binary semaphore for mutually exclusive access to the shared buffer; empty is used by the Producer to check if the buffer is full; full is used by the Consumer to check if the buffer is empty.**

* **Q23. In the Reader-Writer implementation, explain all the functions for each semaphore.** a) mutex is a binary semaphore to ensure mutually exclusive updates to the readcount variable; wrt is a binary semaphore to ensure mutually exclusive access among writers. b) mutex is a binary semaphore to ensure mutually exclusive updates to the readcount variable; wrt is a binary semaphore to ensure mutually exclusive access among writers as well as between readers and writers. c) mutex is a binary semaphore to ensure mutually exclusive updates to the readcount variable; wrt is a counting semaphore to count the number of readers.

**b) mutex is a binary semaphore to ensure mutually exclusive updates to the readcount variable; wrt is a binary semaphore to ensure mutually exclusive access among writers as well as between readers and writers.**

* **Q24. In the Reader-Writer implementation, explain what it means by "the reader is given preference".** a) While a reader is reading, other readers will be allowed access to the database, even if writers are currently waiting to get access. b) If a reader and writer both want access to the database at the same time, then the reader is given preference over the writer irrespective of who is currently accessing the database. c) Readers can always access the database, even if writers are currently accessing it.

**a) While a reader is reading, other readers will be allowed access to the database, even if writers are currently waiting to get access.**

* **Q25. In the Reader-Writer implementation, IF 1) a writer W1 is currently accessing the database, 2) another writer W2 is waiting to get access to the database, and 3) a reader R1 then requests access, THEN R1 will get access before W2.** a) False; R1 will get access after W2 assuming the semaphore blocked list is First-In-First-Out. b) False; R1 will get access after W2 assuming the semaphore blocked list is Last-In-First-Out. c) True; readers have preference over writers.

**a) False; R1 will get access after W2 assuming the semaphore blocked list is First-In-First-Out.**

* **Q26. In the Dining-Philosopher implementation, explain how allowing at most four philosophers to be hungry simultaneously solves the deadlock problem.** a) Assume philosophers first acquire the chopstick on the left and then the one on the right. A philosopher who is sitting to the **left** of the non-hungry philosopher will always get his second chopstick. If he also gets his **first chopstick**, he will eat. Else, his **left** neighbour will eat. b) Assume philosophers first acquire the chopstick on the left and then the one on the right. A philosopher who is sitting to the **right** of the non-hungry philosopher will always get his second chopstick. If he also gets his **first chopstick**, he will eat. Else, his **right** neighbour will eat. c) Assume philosophers first acquire the chopstick on the left and then the one on the right. A philosopher who is sitting to the **left** of the non-hungry philosopher will always get his first chopstick. If he also gets his **second chopstick**, he will eat. Else, his **left** neighbour will eat. d) Assume philosophers first acquire the chopstick on the left and then the one on the right. A philosopher who is sitting to the **right** of the non-hungry philosopher will always get his first chopstick. If he also gets his **second chopstick**, he will eat. Else, his **right** neighbour will eat.

**a) Assume philosophers first acquire the chopstick on the left and then the one on the right. A philosopher who is sitting to the left of the non-hungry philosopher will always get his second chopstick. If he also gets his first chopstick, he will eat. Else, his left neighbour will eat.**

* **Q27. In the Dining-Philosopher problem, explain how the "asymmetric" solution solves deadlock.** a) Odd philosopher first picks left chopstick and then right, whereas even philosopher first picks right chopstick and then left. This ensures that at least one philosopher will not be able to pick any chopstick when all of them try to acquire it simultaneously and hence deadlock is avoided. b) Odd philosopher first picks right chopstick and then left, whereas even philosopher first picks left chopstick and then right. This ensures that at least one philosopher will not be able to pick any chopstick when all of them try to acquire it simultaneously and hence deadlock is avoided. c) Both the other answers are correct.

**c) Both the other answers are correct.**

* **Q28. Race condition is impossible with a single writer, multiple reader process system.** a) False; race condition can occur even without any writers in the system. b) True; in this case the writes and reads can be causally ordered without any issues. c) False; can still cause a race condition between the writer and some of the readers.

**b) True; in this case the writes and reads can be causally ordered without any issues.**

* **Q29. Race condition is not possible when updating a file.** a) True; race condition is only possible with shared data and variables. b) False; when multiple processes are updating the same file concurrently this will cause a race condition. It is however handled in the OS because file updates occur through system call c) False; when multiple processes are updating the same file concurrently this will cause a race condition. Must be handled by the user

**b) False; when multiple processes are updating the same file concurrently this will cause a race condition. It is however handled in the OS because file updates occur through system call**

* **Q30. Race conditions can be prevented by disabling interrupts in critical sections.** a) True; disabling interrupts will prevent context switches which is necessary and sufficient to prevent a race condition. b) True; disabling interrupts will prevent the execution of any other process or even the OS. This will ensure mutual exclusion c) False; interrupts are unrelated to the race condition problem

**b) True; disabling interrupts will prevent the execution of any other process or even the OS. This will ensure mutual exclusion**

## OS Quiz 4

* **Q1. Every completion sequence must be safe, for the system to be in the safe state.** a) False; the system is in the safe state as long as there is at least one safe completion sequence. b) False; the system is in the safe state as long as there is one safe completion sequence, and all the other sequences are not unsafe c) True; this is the definition of a safe state. d) False; the system is in the safe state as long as there is one safe completion sequence, and all the other sequences are not guaranteed to deadlock.

**a) False; the system is in the safe state as long as there is at least one safe completion sequence.**

* **Q2 What is the no preemption condition for a deadlock?** a) The condition requires that a resource locked by a process can only be voluntarily released by the process itself. b) If non-preemptive CPU scheduling is used, then deadlocks will not occur. c) If preemptive CPU scheduling is used, then deadlocks are guaranteed to occur. d) A resource can only be released when a process terminates (exit() system call).

**a) The condition requires that a resource locked by a process can only be voluntarily released by the process itself.**

* **Q3. What do deadlock avoidance algorithms do?** a) They prevent deadlocks through active monitoring of resource requests. b) They avoid all four deadlock conditions from occurring (circular wait, hold and wait, mutual exclusion and no preemption). c) They allow deadlocks to occur, but then recover from deadlocks by breaking one of the four deadlock conditions (circular wait, hold and wait, mutual exclusion and no preemption) d) None of the other options are correct.

**a) They prevent deadlocks through active monitoring of resource requests.**

* **Q4. The ordered locking solution to the dining-philosopher problem (chopsticks must be taken in increasing order of their number), breaks which one of the four deadlock conditions.** a) Circular Wait b) Hold and Wait c) Mutual Exclusion d) No Preemption.

**a) Circular Wait.**

* **Q5. Banker's algorithm ensures that the system never enter.** a) Unsafe state b) Safe state. c) Deadlocked state.

**a) Unsafe state**

* **Q6. What is a safe completion sequence in the Banker's algorithm?** a) If the processes request for remaining resources and complete based on this sequence, then there is guaranteed to be no deadlock. b) If the processes request for remaining resources and complete based on this sequence, then there is a possibility that deadlock will not occur. c) If the processes complete based on this sequence (irrespective of the order in which the request for the remaining resources), then there is guaranteed to be no deadlock.

**a) If the processes request for remaining resources and complete based on this sequence, then there is guaranteed to be no deadlock.**

* **Q7. In Banker's algorithm, what information does the Allocation matrix contain?** a) At each time instant, it denotes the allocated resources for each process. b) At each time instant, it denotes the remaining resource requests (not yet requested) for each process. c) At each time instant, it denotes the remaining resource requests (not yet requested) for each process, over and above its maximum allowed requests. d) At each time instant, it denotes the available resources in the system

**a) At each time instant, it denotes the allocated resources for each process.**

* **Q8. Real-time systems require fast response/turnaround times from processes** a) False; Real-time systems require low waiting times from processes. b) False; Real-time systems require response/turnaround time from processes that are bounded in the worst-case. c) False; Real-time systems require low CPU bursts from processes. d) True; The fast response/turnaround times ensure that the processes are reactive to events in real-time.

**b) False; Real-time systems require response/turnaround time from processes that are bounded in the worst-case.**

* **Q9. The key difference between a periodic and a sporadic real-time process is that:** a) A periodic process is released periodically at fixed time intervals, whereas a sporadic process is released sporadically with a minimum time separation between successive releases. b) They are both identical and specified using the same three parameters, T for period, C for CPU burst length and D for relative deadine. c) A periodic process is released periodically with a minimum time separation between successive releases, whereas a sporadic process is released sporadically at fixed time intervals d) A periodic process is released periodically at fixed time intervals, whereas a sporadic process is released sporadically with a maximum time separation between successive releases.

**a) A periodic process is released periodically at fixed time intervals, whereas a sporadic process is released sporadically with a minimum time separation between successive releases.**

* **Q10. Virtualization denotes a technique in which:** a) A periodic process is released periodically at fixed time intervals, whereas a sporadic process is released sporadically with a minimum time separation between successive releases. b) The underlying computing system is split into its individual components (CPU, memory, I/O devices, etc.). Each component, called a virtual machine, could run its own OS and applications. c) The underlying hardware is abstracted into virtual and independent computing systems called virtual machines, each of which could run its own OS and applications.

**c) The underlying hardware is abstracted into virtual and independent computing systems called virtual machines, each of which could run its own OS and applications.**

* **Q11. Which of the following is NOT a key benefit of virtualization?** a) It reduces the overhead for applications to use hardware resources. b) It provides mitigation against hardware failures. c) It enables flexibility and agility in deployment and use of hardware resources.

**a) It reduces the overhead for applications to use hardware resources.**

* **Q12. Which of the following is NOT true for bare-metal hypervisors?** a) These hypervisors are popular in the industry and deployed extensively. b) They are commonly used for end-user virtualization. c) These hypervisors have low latency because the number of abstraction layers is less when compared to hosted hypervisors. d) These hypervisors are secure because they directly manage the hardware resources.

**b) They are commonly used for end-user virtualization.**

* **Q13. The necessary conditions for a deadlock are** a) circular wait: each process in a cycle requests for a resource held by the next process in the cycle; mutual exclusion: a resource can only be used mutually exclusively; hold and wait: processes are holding resources while requesting for others; no preemption: a locked resource can only be released voluntarily by the process holding it. b) mutual exclusion: a resource can only be used mutually exclusively; hold and wait: processes are holding resources while requesting for others; no preemption: a locked resource can only be released voluntarily by the process holding it. c) circular wait: each process in a cycle holds a resource and requests another resource held by the next process in the cycle; mutual exclusion: a resource can only be used mutually exclusively; no preemption: a locked resource can only be released voluntarily by the process holding it. d) circular wait: each process in a cycle requests for a resource held by the next process in the cycle; mutual exclusion: a resource can only be used mutually exclusively and atomically; hold and wait: processes are holding resources while requesting for others.

**a) circular wait: each process in a cycle requests for a resource held by the next process in the cycle; mutual exclusion: a resource can only be used mutually exclusively; hold and wait: processes are holding resources while requesting for others; no preemption: a locked resource can only be released voluntarily by the process holding it.**

* **Q14. The necessary conditions for a deadlock are** a) False; all four conditions are necessary for a deadlock b) False; all four conditions are sufficient for a deadlock c) True; any one of the four conditions is sufficient for a deadlock. d) True; all four conditions are necessary but not sufficient for a deadlock.

**a) False; all four conditions are necessary for a deadlock**

* **Q15. If a resource allocation graph with multiple instances for each resource type contains a cycle, then there is a guaranteed deadlock** a) False; there is a possibility of a deadlock, but the circular wait and hold and wait conditions could be broken by a process outside the cycle. b) False; it is guaranteed to be not deadlocked. c) False; there is a possibility of a deadlock, but the mutual exclusion condition could be broken by a process outside the cycle. d) True; a cycle in the resource allocation graph always indicates a deadlock because the circular wait and hold and wait conditions are satisfied.

**a) False; there is a possibility of a deadlock, but the circular wait and hold and wait conditions could be broken by a process outside the cycle.**

* **Q16. Deadlocks cannot occur if resources (e.g., semaphores) are not locked in a nested manner.** a) True; without nesting, a process cannot request for a resource while already holding another, this will prevent the hold and wait condition from occurring. b) True; without nesting, a process cannot request for a resource while already holding another, this will prevent the mutual exclusion condition from occurring. c) False; even if there is no nested resource requests, circular wait condition can occur and cause a deadlock. d) False; even if there is no nested resource requests, mutual exclusion condition can occur and cause a deadlock.

**a) True; without nesting, a process cannot request for a resource while already holding another, this will prevent the hold and wait condition from occurring.**

* **Q17. How are deadlocks handled in most popular OS?** a) They are ignored b) Deadlock prevention is used. c) Deadlock avoidance is used d) Deadlock detection is used.

**a) They are ignored**

* **Q18. In Real-time systems, what is a reasonable bound on the response/turnaround time depends on:** a) All of the other answers are correct. b) The timing requirements of the target functionality/application. c) Strategies implemented for detecting and handling of failures in the system. d) The mechanical properties of the sensors, actuators and other physical devices in the system.

**a) All of the other answers are correct.**

* **Q19. The key difference between fixed-priority and dynamic-priority real-time CPU scheduling is that:** a) They are identical, except the fact that priorities are based on different parameters (T under fixed-priority scheduling and D under dynamic-priority scheduling). b) None of the other responses are correct. c) Under fixed-priority real-time scheduling priorities are fixed between recurrent processes (all the instances of one recurrent process will have higher or lower priority than all the instances of another recurrent process), whereas under dynamic-priority real-time scheduling priorities are only fixed between process instances (an instance of a recurrent process will have higher or lower priority than an instance of another recurrent process). d) Under fixed-priority real-time scheduling priorities are fixed between recurrent processes (all the instances of one recurrent process will have higher or lower priority than all the instances of another recurrent process), whereas under dynamic-priority real-time scheduling no such restrictions exist.

**d) Under fixed-priority real-time scheduling priorities are fixed between recurrent processes (all the instances of one recurrent process will have higher or lower priority than all the instances of another recurrent process), whereas under dynamic-priority real-time scheduling no such restrictions exist.**

* **Q20. In Banker's algorithm, what is the information stored in the Need matrix?** a) At each time instant, it denotes the remaining resource requests (not yet requested) for each process. b) At each time instant, it denotes the allocated resources for each process. c) At each time instant, it denotes the available resources in the system. d) At each time instant, it denotes the remaining resource requests (not yet requested) for each process, over and above its maximum allowed requests.

**a) At each time instant, it denotes the remaining resource requests (not yet requested) for each process.**

* **Q21. In a resource allocation graph, an assignment edge is** a) An edge from a resource to a process denoting a granted resource request. b) An edge from a resource to a process denoting a pending resource request. c) An edge from a process to a resource denoting a pending resource request. d) An edge from a process to a resource denoting a granted resource request.

**a) An edge from a resource to a process denoting a granted resource request.**

* **Q22. The odd-even solution to the dining philosopher problem (odd philosophers first take the left chopstick and then the right; even philosophers first take the right chopstick and then the left) breaks which one of the four deadlock conditions?** a) Circular Wait b) Hold and Wait c) Mutual Exclusion. d) No Preemption.

**a) Circular Wait**

* **Q23. When a system is in the unsafe state, then deadlock is guaranteed.** a) False; deadlock can still be avoided if processes release resources that are currently held before they take more resources. b) False; deadlock can still be avoided if processes release resources when they terminate (in the exit() system call). c) True; an unsafe state implies an imminent and unavoidable deadlock. d) False; deadlock can still be avoided if processes complete based on any existing safe completion sequence.

**a) False; deadlock can still be avoided if processes release resources that are currently held before they take more resources.**

* **Q24. If a system satisfies any two of the four deadlock conditions (mutual exclusion, no preemption, hold and wait, circular wait), then a deadlock has occurred.** a) False; all four conditions are necessary for a deadlock. b) False; all four conditions are sufficient for a deadlock. c) True; any one of the four conditions is sufficient for a deadlock. d) True; all four conditions are necessary but not sufficient for a deadlock.

**a) False; all four conditions are necessary for a deadlock.**

* **Q25. Fixed-priority real-time CPU scheduling is easier to implement than dynamic-priority real-time CPU scheduling.** a) False; they both have the same implementation complexity because they require sorting of the process instances based on priorities. b) True; A separate queue can be maintained for each priority level under fixed-priority scheduling leading to a O(1) complexity implementation. Whereas under dynamic-priority scheduling, online sorting of the queue is required whenever a new process instance is released in the system (cannot be done in O(log n) complexity, but can be done in O(n log n) complexity using a sorting algorithm where n is the number of process instances). c) True; A separate queue can be maintained for each priority level under fixed-priority scheduling leading to a O(1) complexity implementation. Whereas under dynamic-priority scheduling, online sorting of the queue is required whenever a new process instance is released in the system (can be done in O(log n) complexity using tree data-structure where n is the number of process instances). d) None of the other responses are correct.

**c) True; A separate queue can be maintained for each priority level under fixed-priority scheduling leading to a O(1) complexity implementation. Whereas under dynamic-priority scheduling, online sorting of the queue is required whenever a new process instance is released in the system (can be done in O(log n) complexity using tree data-structure where n is the number of process instances).**

* **Q26. If a resource allocation graph with one instance for each resource type contains a cycle, then there is a guaranteed deadlock.** a) False; the cycle denotes a hold and wait condition, but it could be broken by some process outside the cycle. b) True; the cycle denotes a mutual exclusion condition which cannot be broken by any process outside the cycle. c) True; the cycle denotes circular wait and hold and wait conditions which cannot be broken by any process outside the cycle. d) False; the cycle denotes a circular wait condition, but it could be broken by some process outside the cycle.

**c) True; the cycle denotes circular wait and hold and wait conditions which cannot be broken by any process outside the cycle.**

* **Q27. What do deadlock prevention algorithms do?** a) They prevent one of the four deadlock conditions from occurring (circular wait, hold and wait, mutual exclusion and no preemption). b) They prevent all four deadlock conditions from occurring (circular wait, hold and wait, mutual exclusion and no preemption). c) They prevent deadlocks through active monitoring of resource requests. d) They prevent deadlocks by preventing locking of resources altogether.

**a) They prevent one of the four deadlock conditions from occurring (circular wait, hold and wait, mutual exclusion and no preemption).**

* **Q28. The key functions of a Hypervisor or Virtual Machine Manager include:** a) All of the others are correct. b) Virtual Machine management (creation, resource allocation, deletion, etc.). c) Migration of Virtual Machines from one hardware to another, almost instantaneously. d) Communication between Virtual Machines.

**a) All of the others are correct.**

* **Q29. In Banker's algorithm, what information is stored in the Work vector** a) At each time instant, it denotes the available resources in the system. b) At each time instant, it denotes the remaining resource requests (not yet requested) for each process. c) At each time instant, it denotes the allocated resources for each process. d) At each time instant, it denotes the remaining resource requests (not yet requested) for each process, over and above its maximum allowed requests.

**a) At each time instant, it denotes the available resources in the system.**

* **Q30. In a resource allocation graph, the dots inside a resource denote:** a) The number of instances of that resource available in the system. b) The number of processes who have been allocated that resource. c) The number of processes who have requested that resource.

**a) The number of instances of that resource available in the system.**

## OS Quiz 5

* **Q1. Given the logical address 0xAEF9 (in hexadecimal) with a page size of 256 bytes, what is the page offset/page number?** a. 0xAF9 b. 0xAE c. 0XF9 d. 0X9

**b. 0xF9(page offset) / 0xAE (page number)**

* **Q2. Logical address and physical address will be the same if address binding is performed in** a. compile-time b. load-time c. execution-time d. both compile-time and load-time

**d. both compile-time and load-time**

* **Q3. Which dynamic storage-allocation policy results in the smallest leftover hole in memory?** a. First fit b. Best fit c. Worst fits

**b. Best fit**

* **Q4. Which of the following memory allocation methods may result in internal fragmentation?** a. Dynamic partitioning b. Paging c. Segmentation

**b. Paging**

* **Q5. Memory compaction can be performed if address binding is done in** a. execution-time b. compile-time c. load-time

**a. execution-time**

* **Q6. The mapping of a logical address to a physical address is done dynamically in** a. load-time binding b. execution-time binding c. compile-time binding

**b. execution-time binding**

* **Q7. Which of the following memory allocation approaches allocates contiguous memory space for a process?** a. Dynamic partitioning b. Paging c. Segmentation d. All of the Above

**a. Dynamic partitioning**

* **Q8. Which dynamic storage-allocation policy results in the largest leftover hole in memory?** a. First fit b. Best fit c. Worst fit

**c. Worst fit**

* **Q9. Which of the following memory allocation approaches assumes logical address space of a process is contiguous?** a. Dynamic partitioning b. Paging c. Segmentation d. All of the above

**d. All of the above**

* **Q10. Considering a logical address with a page size of 8 KB, how many bits must be used to represent the page offset in the logical address?** a. 13 b. 10 c. 12 d. 8

**a. 13**

* **Q11. Re-locatable code will be generated for** a. compile-time binding b. execution time binding c. load time binding

**c. load time binding**

* **Q12. Which dynamic storage-allocation policy has least overhead?** a. Best fit b. Worst fit c. First fit

**c. First fit**

* **Q13. Which of the following statement about memory compaction is true?** a. It does not shuffle memory contents b. It is used to solve the problem of internal fragmentation c. It is possible only if address binding is dynamic and done at execution time d. It can be done at compile, load, or execution time

**c. It is possible only if address binding is dynamic and done at execution time**

* **Q14. Which of the following memory allocation methods may result in external fragmentation?** a. Both paging and Segmentation b. Paging c. Both dynamic partitioning & paging d. Both dynamic partitioning & segmentation

**d. Both dynamic partitioning & segmentation**