

JK Lakshmipat University Jaipur
INSTITUTE OF ENGINEERING AND TECHNOLOGY

End Term Examination, May 2026
 B. Tech. (CSE), Semester IV, 2025-26

Roll No.....

CS1108: Operating Systems

Time: 2.5 hours

Max. Marks: 40

Instructions to students:

1. Do not write anything other than your roll number on question paper.
2. Mention all the assumptions for your answers clearly.
3. **Scientific calculators are allowed.**

Q.1.1	Given three page frames (initially empty) are available for storing data, determine the number of page faults in each of the following cases using the LRU page replacement algorithm. (i) Consider the code below. Assume that the array is stored in a row major order and each row is stored in exactly one page. Justify your answer <pre>int x[200][200]; for (i = 0; i<200; i++){ for (j = 0; j<200; j++) x[j][i] = 0; }</pre> (ii) Consider a complete graph $G(V,E)$ with $V = 128$ vertices. Two programmers represent the graph using different data structures: Programmer 1 uses an adjacency matrix. Programmer 2 uses an adjacency list representation. Assume the following: - Each row of the adjacency matrix is stored in exactly one page. - Each adjacency list corresponding to a vertex is stored in one page. - The graph is traversed vertex by vertex, accessing all entries of one vertex before moving to the next.	1 (CO2) 2 (CO2)
Q.1.2	A specific program accesses a sequence of pages as follows: 7, 0, 1, 2, 0, 3, 0, 4, 2. Initially, all page frames are empty. Using FIFO page replacement algorithm, the number of page faults when the system has 3 page-frames are 8. 1) Repeat the process with 4 page-frames. Document each step, showing which pages are in the frames after each page access. 2) Discuss whether Belady's Anomaly occurs in this case and explain your answer	2 (CO2)
Q.2.1	In a system with 16 MB of physical memory and a 24-bit virtual address space, using pages of size 4 KB, determine the size of the page table, assuming each entry in the page table occupies 4 bytes. Given a virtual address 0xABCD E F, calculate the page number, and offset. Without performing any arithmetic calculations, write the corresponding physical address. Clearly show how the address is formed.	4 (CO2)

Q.2.2	Considering a three-level paging system where the virtual address space is 64 GB and the physical memory is 128 GB. The system uses a page size of 4 KB. a. Calculate how many bits are needed for the page number and offset at each level of the page table. Assume there are an additional 7 bits as control bits in a PTE. b. If a virtual address 0x123456789 is given, break it down into its constituent parts according to the multi-level page table. Assume each page table can take a maximum of one page worth of space.	4 (CO2)															
Q.3.1	Consider the memory requests from processes in the following order: 180K, 90K. The available memory blocks are: 100K, 200K, 300K. Apply Worst Fit and Best Fit. Clearly show the final free blocks and the total external fragmentation in each case.	2 (CO2)															
Q.3.2	With a physical memory size of 256 MB and a page size of 2 KB, given an inverted page table is used, with a page table entry size of 4 bytes, estimate the total size of the inverted page table. Given the following IPT, for Process P2 and virtual page number 5, what is the frame number and displacement? <table border="1"> <thead> <tr> <th>Process</th><th>Page</th><th>Valid</th></tr> </thead> <tbody> <tr> <td>P1</td><td>2</td><td>V</td></tr> <tr> <td>P2</td><td>5</td><td>V</td></tr> <tr> <td>P0</td><td>5</td><td>V</td></tr> <tr> <td>P2</td><td>1</td><td>V</td></tr> </tbody> </table>	Process	Page	Valid	P1	2	V	P2	5	V	P0	5	V	P2	1	V	3 (CO2)
Process	Page	Valid															
P1	2	V															
P2	5	V															
P0	5	V															
P2	1	V															
Q.4.1	You may assume the following: 1. The location of the first block of the file is already in memory 2. There is room for file to grow at the end but not at the beginning 3. The middle block is 25th block. 4. Do not include any I/O operations needed to add a removed block to the free list. 5. The block of information to be added is already in memory 6. In calculating the number of I/O operations, if the location of the first block of a file changes, do not include the output operations needed to rewrite that revised information to the directory entry on the disk. How many read and write operations would be required to perform the following operations from a 50-block file under the conditions specified above? <i>Assume each question is independent:</i> a) A system uses contiguous allocation and add a block at the beginning b) A system uses contiguous allocation and remove the middle block c) A system uses linked allocation and add a block after the middle block d) A system uses linked allocation and remove the last/end block	4 (CO6)															
Q.4.2	On a disk with 1000 cylinders, numbers 0 to 999, compute the number of tracks the disk arm must move to satisfy all requests in the disk queue. Assume the head is at 756. The queue is 811, 348, 153, 968, 407, 500. Perform the computation for SSTF AND C-SCAN (Assume the head is moving towards 0).	2+2 (CO6)															

Q.5	Answer any two of the following questions:	
a)	Explain why a TLB is used in modern computer systems. Describe what happens during a TLB hit and TLB miss when a process accesses a virtual address.	
b)	Compare Segmentation versus Paging using a program of your choice (simple C program). Compare segmentation and paging in terms of external and internal fragmentation.	2+2 (CO2, 5)
c)	Discuss the five-state process life cycle. Draw a neat labeled diagram.	
Q.6.1	Implement two-way communication between parent and child processes. Since a single pipe is unidirectional, you must create two pipes to achieve stable two-way communication without the processes competing for their own data. Create a "Token Exchange" program. The parent sends "Start" to the child; the child receives it and sends back "Acknowledged" to the parent. Display all messages appropriately.	4 (CO1, 5)
Q.6.2	Consider a scenario where a bus picks up waiting passengers from a bus stop periodically. The bus has a capacity of K. The bus arrives at the bus stop, allows up to K waiting passengers (fewer if less than K are waiting) to board, and then departs. Passengers have to wait for the bus to arrive and then board it. Passengers who arrive at the bus stop after the bus has arrived should not be allowed to board, and should wait for the next time the bus arrives. The bus and passengers are represented by processes in a program. The passenger process should call the function board() after the passenger has boarded and the bus should invoke depart() when it has boarded the desired number of passengers and is ready to depart. The processes share the following variables, none of which are implicitly updated by functions like board() or depart(). <pre>Semaphore mutex = 1 Semaphore bus_arrived = 0 Semaphore passenger_boarded = 0 int waiting_count = 0</pre> Below is given synchronized code for the passenger process. You should not modify this in any way. <pre>wait(mutex); waiting_count++; signal(mutex); wait(bus_arrived); board(); signal(passenger_boarded);</pre> Write down the corresponding synchronized code for the bus process that achieves the correct behavior specified above. The bus should board the correct number of passengers, based on its capacity and the number of those waiting. The bus should correctly board these passengers by calling wait/signal on the semaphores suitably. The bus code should also update waiting count as required. Once boarding completes, the bus process should call depart(). <i>You can use any extra local variables in the code of the bus process, like integers, loop indices and so on. However, you must NOT use any other extra synchronization primitives.</i>	6 (CO5)

