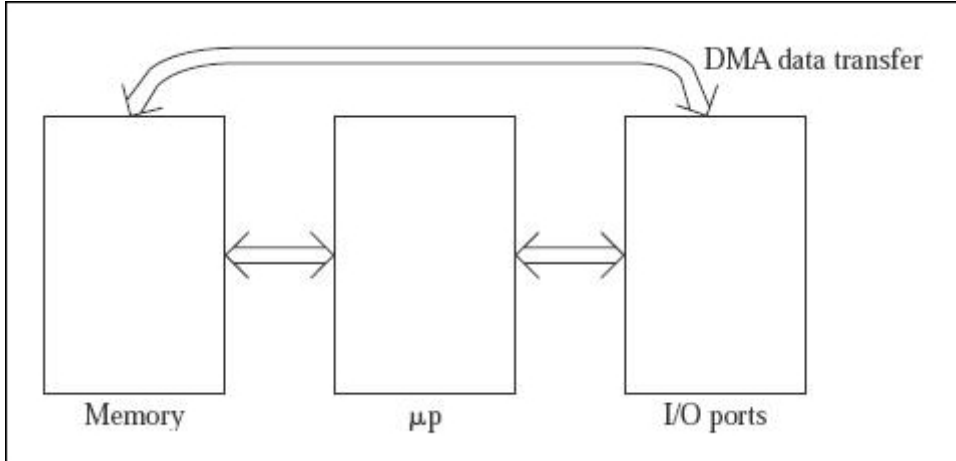


Q	ANSWERS																
1	a)	What is the content of page table? Explain.															
	Ans	<ul style="list-style-type: none"> The content of the page table is the data structure that stores the mapping between virtual addresses and physical addresses in memory. The page table allows the operating system to manage memory efficiently and protect the system from unauthorized or erroneous access to memory. The content of the page table can be explained as follows: <ol style="list-style-type: none"> Page Table Entry (PTE): <ul style="list-style-type: none"> An entry in the page table that contains information about a specific page of memory, such as its physical address, presence, protection, and access permissions. The size and format of a PTE can vary depending on the system architecture and the operating system used. Translation Lookaside Buffer (TLB): <ul style="list-style-type: none"> A cache of recently used mappings from the page table that is stored in the memory management unit (MMU) inside the CPU. The TLB is searched first when a virtual address needs to be translated into a physical address. If a match is found, the translation is fast, and the memory access can continue. If not, the page table is looked up, which is slower and may cause a page fault. Page Fault: <ul style="list-style-type: none"> A situation where the page table lookup fails, either because the virtual address is invalid or because the corresponding page is not in memory. The operating system must handle the page fault by either terminating the process, allocating a new page, or bringing the page from secondary storage to memory. Hierarchical Page Table: <ul style="list-style-type: none"> A technique to reduce the size and improve the performance of the page table by dividing it into smaller tables, each pointing to a larger table. <p>This allows for more efficient memory management and faster access to page table entries.</p> 															
	b)	Compare process scheduling and process switching.															
	Ans	<table border="1"> <thead> <tr> <th>Aspect</th><th>Process Scheduling</th><th>Process Switching</th></tr> </thead> <tbody> <tr> <td>Definition</td><td>Technique to determine the order of process execution.</td><td>Mechanism to transition from one process to another.</td></tr> <tr> <td>Objective</td><td>Optimize CPU utilization, improve performance, fairness.</td><td>Save and restore process context for seamless execution.</td></tr> <tr> <td>Frequency</td><td>Occurs less frequently (process completion or blocking).</td><td>Occurs more frequently (event-driven or preemptive).</td></tr> <tr> <td>Decision Criteria</td><td>Based on scheduling algorithms, priorities, and waiting time.</td><td>Event-driven, e.g., interrupts, time slices, priorities.</td></tr> </tbody> </table>	Aspect	Process Scheduling	Process Switching	Definition	Technique to determine the order of process execution.	Mechanism to transition from one process to another.	Objective	Optimize CPU utilization, improve performance, fairness.	Save and restore process context for seamless execution.	Frequency	Occurs less frequently (process completion or blocking).	Occurs more frequently (event-driven or preemptive).	Decision Criteria	Based on scheduling algorithms, priorities, and waiting time.	Event-driven, e.g., interrupts, time slices, priorities.
Aspect	Process Scheduling	Process Switching															
Definition	Technique to determine the order of process execution.	Mechanism to transition from one process to another.															
Objective	Optimize CPU utilization, improve performance, fairness.	Save and restore process context for seamless execution.															
Frequency	Occurs less frequently (process completion or blocking).	Occurs more frequently (event-driven or preemptive).															
Decision Criteria	Based on scheduling algorithms, priorities, and waiting time.	Event-driven, e.g., interrupts, time slices, priorities.															

		Context Switching	May or may not involve context switching.	Always involves context switching.
		Resource Allocation	Deals with CPU time allocation in a multiprogramming system.	Involves temporarily halting one process to start another.
		Process Scheduling Vs Process Switching		
	c)	What is Semaphore? What is its significance?		
	Ans	<ul style="list-style-type: none"> • A semaphore is a synchronization object that controls access by multiple processes to a common resource in a parallel programming environment. • Semaphores are used to enforce mutual exclusion, avoid race conditions, and implement synchronization between processes. • The process of using Semaphores provides two operations: wait (P) and signal (V). • The wait operation decrements the value of the semaphore, and the signal operation increments the value of the semaphore. • When the value of the semaphore is zero, any process that performs a wait operation will be blocked until another process performs a signal operation. <p>Here are some advantages of semaphores:</p> <ul style="list-style-type: none"> - A simple and effective mechanism for process synchronization. - Supports coordination between multiple processes. - Provides a flexible and robust way to manage shared resources. - It can be used to implement critical sections in a program. - It can be used to avoid race conditions. <p>Here are some disadvantages of semaphores:</p> <ul style="list-style-type: none"> - Implementing semaphore can lead to priority inversion where the two processes get into spinlock condition. - It can cause performance issues in a program if not used properly. <p>It can be difficult to debug and maintain.</p>		
	d)	Explain UNIX OS kernel.		
	Ans	<ul style="list-style-type: none"> • The kernel is the central component of an operating system that manages communication between hardware and software components. • It provides an interface to the hardware devices as well as to process, memory, and I/O management. • The UNIX kernel consists of many kernel subsystems like process management, scheduling, file management, device management, network management, memory management, and dealing with interrupts from hardware devices. <p>1. Process management:</p> <ul style="list-style-type: none"> - The kernel is responsible for creating, scheduling, and terminating processes. - It also manages the process's memory allocation and deallocation. 		

		<ul style="list-style-type: none"> - The kernel creates a process when a program is executed and assigns it a unique process ID. - It also schedules the processes to run on the CPU and allocates resources such as memory and I/O devices to the processes. <p>2. Memory management:</p> <ul style="list-style-type: none"> - The kernel is responsible for managing the memory of the system. - It allocates memory to processes and deallocates memory when it is no longer needed. - The kernel uses virtual memory to manage the memory of the system. - It divides the memory into pages and maps the virtual addresses used by the processes to the physical addresses of the memory. <p>3. File System:</p> <ul style="list-style-type: none"> - The kernel is responsible for managing the file system. - It provides an interface to the file system and manages the creation, deletion, and modification of files and directories. - The kernel provides a virtual file system that abstracts the physical file system and provides a uniform interface to the processes. <p>4. Device Drivers:</p> <ul style="list-style-type: none"> - The kernel is responsible for managing the device drivers. - It provides an interface to the device drivers and manages the communication between the device drivers and the hardware devices. - The kernel provides a device driver framework that allows the device drivers to be loaded and unloaded dynamically. <p>5. Handling interrupts:</p> <ul style="list-style-type: none"> - The kernel is responsible for handling interrupts from hardware devices. - It manages the communication between the hardware devices and the software components of the system. - When a hardware device generates an interrupt, the kernel suspends the current process and executes the interrupt handler. <p>The interrupt handler communicates with the device driver to handle the interrupt.</p>
	e)	Explain Direct Memory Access (DMA) in detail.
	Ans	<ul style="list-style-type: none"> • DMA is a feature of computer systems that allows certain hardware subsystems to access main system memory independently of the central processing unit (CPU). • DMA is used by many hardware systems, including disk drive controllers, graphics cards, network cards, and sound cards. • DMA can offload expensive memory operations, such as large copies or scatter-gather operations, from the CPU to a dedicated DMA engine. • DMA can also be used for "memory to memory" copying or moving of data within memory. • DMA reduces the overhead involved in data transfers, freeing up the CPU to

		<p>continue processing other operations instead.</p> <ul style="list-style-type: none">• DMA transfers data directly between devices, while non-DMA transfers require CPU processing.• DMA requires very few clock cycles while transferring data, and it distributes workload very appropriately.• DMA helps the CPU in decreasing its load.• Cache coherence problems can occur while using DMA controllers. <div><p style="text-align: center;">Fig. Direct Memory Access (DMA)</p></div>																								
2	a)	<p>Consider the following snapshot of the processes:</p> <table border="1" data-bbox="339 1164 1246 1426"><thead><tr><th>Process</th><th>Burst time</th><th>Arrival time</th><th>Priority</th></tr></thead><tbody><tr><td>P1</td><td>8</td><td>0</td><td>1</td></tr><tr><td>P2</td><td>20</td><td>1</td><td>3</td></tr><tr><td>P3</td><td>3</td><td>2</td><td>2</td></tr><tr><td>P4</td><td>6</td><td>3</td><td>5</td></tr><tr><td>P5</td><td>12</td><td>4</td><td>4</td></tr></tbody></table> <p>i. Draw the Gantt chart for the execution of the processes, showing their start time and end time using FCFS, SJF (without considering the priority), priority scheduling (pre-emptive), RR (with time quantum=5), Calculate turnaround time, and average waiting time and average turnaround time for the system.</p>	Process	Burst time	Arrival time	Priority	P1	8	0	1	P2	20	1	3	P3	3	2	2	P4	6	3	5	P5	12	4	4
Process	Burst time	Arrival time	Priority																							
P1	8	0	1																							
P2	20	1	3																							
P3	3	2	2																							
P4	6	3	5																							
P5	12	4	4																							
	Ans	<p>i. Here is the Gantt chart for the execution of the processes using different scheduling algorithms:</p> <ul style="list-style-type: none">• First Come First Serve (FCFS):																								

Process	Burst Time	Arrival Time	Start Time	End Time
P1	8	0	0	8
P2	20	1	8	28
P3	3	2	28	31
P4	6	3	31	37
P5	12	4	37	49

• **Shortest Job First (SJF):**

Process	Burst Time	Arrival Time	Start Time	End Time
P1	8	0	0	8
P3	3	2	8	11
P4	6	3	11	17
P5	12	4	17	29
P2	20	1	29	49

• **Priority Scheduling (Pre-emptive):**

Process	Burst Time	Arrival Time	Priority	Start Time	End Time
P1	8	0	1	0	8
P2	20	1	3	8	28
P3	3	2	2	28	31
P4	6	3	5	31	37
P5	12	4	4	37	49

- **Round Robin (RR) with time quantum = 5:**

Process	Burst Time	Arrival Time	Start Time	End Time
P1	8	0	0	8
P2	20	1	8	13
P3	3	2	13	16
P4	6	3	16	22
P5	12	4	22	34
P2	20	1	34	49

- ii. The turnaround time, average waiting time, and average turnaround time for the system are as follows:

		<ul style="list-style-type: none"> • First Come First Serve (FCFS): <ul style="list-style-type: none"> ◦ Turnaround Time: P1=8, P2=27, P3=29, P4=34, P5=45 ◦ Average Waiting Time: 16.2 ◦ Average Turnaround Time: 28.6 • Shortest Job First (SJF): <ul style="list-style-type: none"> ◦ Turnaround Time: P1=8, P2=29, P3=11, P4=17, P5=29 ◦ Average Waiting Time: 9.2 ◦ Average Turnaround Time: 18.8 • Priority Scheduling (Pre-emptive): <ul style="list-style-type: none"> ◦ Turnaround Time: P1=8, P2=28, P3=29, P4=34, P5=45 ◦ Average Waiting Time: 10.6 ◦ Average Turnaround Time: 28.8 • Round Robin (RR) with time quantum = 5: <ul style="list-style-type: none"> ◦ Turnaround Time: P1=8, P2=48, P3=14, P4=19, P5=30 ◦ Average Waiting Time: 11.8 ◦ Average Turnaround Time: 23.8 <p>Based on the above results, the SJF algorithm has the lowest average waiting time and is the most efficient algorithm for the given processes.</p>						
	b)	Explain with suitable example, how virtual address is converted to physical address?						
	Ans	<ul style="list-style-type: none"> • The process of converting a virtual address to a physical address is called address translation. • The operating system uses a data structure called the page table to store the mapping between virtual addresses and physical addresses in memory. • The most significant bits in the virtual address denote the page number being referred, and the remaining bits will be the page offset. • To get the frame address in the main memory, the first bits of the virtual address are used. • The page table stores the corresponding frame accommodating the page. • The last bits of the virtual address remain the same in the physical address as that of the virtual address. • Example: <p>Here is an example table for the given question:</p> <table border="1"> <thead> <tr> <th>Page</th><th>Page Frame</th><th>Reference Bit</th></tr> </thead> <tbody> <tr> <td>0</td><td>1</td><td>0</td></tr> </tbody> </table>	Page	Page Frame	Reference Bit	0	1	0
Page	Page Frame	Reference Bit						
0	1	0						

		1		0
		2		0
		3	9	0
		4		0
		5	14	0
		6		0
		7		0
		8		0
		9	10	0
		10	13	0
		11	15	0
		12	8	0
		13	5	0
		14	4	0
		15	2	0
	convert the following virtual addresses (in hexadecimal) to the equivalent physical addresses. Also I want to set the reference bit for the appropriate entry in the page table.			

		<ul style="list-style-type: none">• 0xE12C• 0x3A9D• 0xA9D9• 0x7001• 0xACA1 <p>Here are the physical addresses for the given virtual addresses:</p> <p>0xE12C → 0x312C 0x3A9D → 0xAA9D 0xA9D9 → 0x59D9 0x7001 → 0xF001 0xACA1 → 0x5CA1</p>																																								
3	a)	<p>Consider the following state of a system with four processes, P1, P2, P3, and P4, and five types of resources, RS1, RS2, RS3, RS4, and RS5:</p> <div><div>C=<table><tr><td>0</td><td>1</td><td>1</td><td>1</td><td>2</td></tr><tr><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td></tr><tr><td>2</td><td>1</td><td>0</td><td>0</td><td>0</td></tr></table><p>E = (24144) A = (01021)</p></div><div>G=<table><tr><td>1</td><td>1</td><td>0</td><td>2</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td><td>2</td><td>1</td></tr><tr><td>0</td><td>2</td><td>0</td><td>3</td><td>1</td></tr><tr><td>0</td><td>2</td><td>1</td><td>1</td><td>0</td></tr></table></div></div> <p>Using the deadlock detection algorithm check deadlock is there or not? If deadlock is there, then identify the processes that are deadlocked.</p>	0	1	1	1	2	0	1	0	1	0	0	0	0	0	1	2	1	0	0	0	1	1	0	2	1	0	1	0	2	1	0	2	0	3	1	0	2	1	1	0
0	1	1	1	2																																						
0	1	0	1	0																																						
0	0	0	0	1																																						
2	1	0	0	0																																						
1	1	0	2	1																																						
0	1	0	2	1																																						
0	2	0	3	1																																						
0	2	1	1	0																																						
	Ans	<p>The given system state can be represented as follows:</p> <table><tr><th>Process</th><th>RS1</th><th>RS2</th><th>RS3</th><th>RS4</th><th>RS5</th></tr><tr><td>P1</td><td>0</td><td>1</td><td>1</td><td>1</td><td>2</td></tr><tr><td>P2</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td></tr><tr><td>P3</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>P4</td><td>1</td><td>2</td><td>1</td><td>0</td><td>0</td></tr></table>	Process	RS1	RS2	RS3	RS4	RS5	P1	0	1	1	1	2	P2	0	1	0	1	0	P3	0	0	0	0	0	P4	1	2	1	0	0										
Process	RS1	RS2	RS3	RS4	RS5																																					
P1	0	1	1	1	2																																					
P2	0	1	0	1	0																																					
P3	0	0	0	0	0																																					
P4	1	2	1	0	0																																					

Resource	RS1	RS2	RS3	RS4	RS5
Available	2	4	2	2	2

Process	RS1	RS2	RS3	RS4	RS5
P1	0	1	1	1	2
P2	0	1	0	1	0
P3	2	1	1	2	0
P4	1	2	1	0	0

Process	RS1	RS2	RS3	RS4	RS5
P1	0	2	2	2	4
P2	0	0	0	0	0
P3	2	2	1	2	0
P4	2	3	2	0	0

		<p>The system state can be represented as follows:</p> <table><tr><th>Process</th><th>RS1</th><th>RS2</th><th>RS3</th><th>RS4</th><th>RS5</th></tr><tr><td>P1</td><td>0</td><td>2</td><td>2</td><td>2</td><td>4</td></tr><tr><td>P2</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>P3</td><td>2</td><td>2</td><td>1</td><td>2</td><td>0</td></tr><tr><td>P4</td><td>2</td><td>3</td><td>2</td><td>0</td><td>0</td></tr></table> <p>The system is in a deadlocked state. The processes that are deadlocked are P1, P3, and P4.</p>	Process	RS1	RS2	RS3	RS4	RS5	P1	0	2	2	2	4	P2	0	0	0	0	0	P3	2	2	1	2	0	P4	2	3	2	0	0
Process	RS1	RS2	RS3	RS4	RS5																											
P1	0	2	2	2	4																											
P2	0	0	0	0	0																											
P3	2	2	1	2	0																											
P4	2	3	2	0	0																											
	b)	What is virtual memory technique? Discuss segmentation with example.																														
Ans		<ul style="list-style-type: none">Virtual memory is a memory management technique that allows a computer to temporarily increase the capacity of its main memory by using secondary memory such as a hard drive or solid-state drive (SSD).Segmentation is a memory management technique that divides a process into segments, which are not necessarily of the same size, and maps them to physical memory.Here are some points to help you understand segmentation:<ul style="list-style-type: none">Segmentation is a memory management technique that divides a process into segments, which are not necessarily of the same size, and maps them to physical memory.Segmentation provides the user's view of the process, which paging does not provide.Segmentation can be of two types: virtual memory segmentation and simple segmentation.In virtual memory segmentation, each process is divided into a number of segments, but the segmentation is not done all at once. This segmentation may or may not take place at the run time of the program.In simple segmentation, each process is divided into a number of segments, all of which are loaded into memory at run time, though not necessarily contiguously.A table stores the information about all such segments and is called a segment table. It maps a two-dimensional logical address into a one-dimensional physical address. Each table entry has a base address and a segment limit.Segmentation provides a higher degree of flexibility than paging. Segments can be of variable size, and processes can be designed to have multiple																														

		<div>segments, allowing for more fine-grained memory allocation.</div> <div><div><div>- Segmentation allows for sharing of memory segments between processes. This can be useful for inter-process communication or for sharing code libraries.</div><div>- Segmentation provides a level of protection between segments, preventing one process from accessing or modifying another process's memory segment. This can help increase the security and stability of the system.</div></div></div>																																																																																																																																																																																
4	a)	<div>Consider the following reference string: 1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6.</div> <div>Find the number of page faults with FIFO,</div> <div>Optimal Page Replacement and LRU with frame size=4,</div>																																																																																																																																																																																
	Ans	<div>i. FIFO algorithm</div> <div><div><div>• Total frames: 4</div><div>• Algorithm: FIFO</div><div>• Reference string length: 20 references</div><div>• String: 1 2 3 4 2 1 5 6 2 1 2 3 7 6 3 2 1 2 3 6</div></div></div> <div>Solution visualization</div> <table><tr><td>t</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td></tr><tr><td>ref</td><td></td><td>1</td><td>2</td><td>3</td><td>4</td><td>2</td><td>1</td><td>5</td><td>6</td><td>2</td><td>1</td><td>2</td><td>3</td><td>7</td><td>6</td><td>3</td><td>2</td><td>1</td><td>2</td><td>3</td><td>6</td></tr><tr><td>f</td><td></td><td>1</td><td>2</td><td>3</td><td>4</td><td>4</td><td>4</td><td>5</td><td>6</td><td>2</td><td>1</td><td>1</td><td>3</td><td>7</td><td>6</td><td>6</td><td>2</td><td>1</td><td>1</td><td>3</td><td>3</td></tr><tr><td>f</td><td></td><td></td><td>1</td><td>2</td><td>3</td><td>3</td><td>3</td><td>4</td><td>5</td><td>6</td><td>2</td><td>2</td><td>1</td><td>3</td><td>7</td><td>7</td><td>6</td><td>2</td><td>2</td><td>1</td><td>1</td></tr><tr><td>f</td><td></td><td></td><td></td><td>1</td><td>2</td><td>2</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>6</td><td>2</td><td>1</td><td>3</td><td>3</td><td>7</td><td>6</td><td>6</td><td>2</td><td>2</td></tr><tr><td>f</td><td></td><td></td><td></td><td></td><td>1</td><td>1</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>5</td><td>6</td><td>2</td><td>1</td><td>1</td><td>3</td><td>7</td><td>7</td><td>6</td><td>6</td></tr><tr><td>hit</td><td></td><td>X</td><td>X</td><td>X</td><td>X</td><td>✓</td><td>✓</td><td>X</td><td>X</td><td>X</td><td>X</td><td>✓</td><td>X</td><td>X</td><td>X</td><td>✓</td><td>X</td><td>X</td><td>✓</td><td>X</td><td>✓</td></tr><tr><td>v</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>1</td><td>2</td><td>3</td><td>4</td><td></td><td>5</td><td>6</td><td>2</td><td></td><td>1</td><td>3</td><td></td><td>7</td><td></td></tr></table> <div><div><div>• Total references: 20</div><div>• Total distinct references: 7</div><div>• Hits: 6</div><div>• Faults: 14</div><div>• Hit rate: $6/20 = 30\%$</div><div>• Fault rate: $14/20 = 70\%$</div></div></div> <div>ii. Optimal Page Replacement algorithm</div> <div><div><div>• Total frames: 4</div><div>• Algorithm: OPT</div><div>• Reference string length: 20 references</div><div>• String: 1 2 3 4 2 1 5 6 2 1 2 3 7 6 3 2 1 2 3 6</div></div></div> <div>Solution visualization</div>	t	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	ref		1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6	f		1	2	3	4	4	4	5	6	2	1	1	3	7	6	6	2	1	1	3	3	f			1	2	3	3	3	4	5	6	2	2	1	3	7	7	6	2	2	1	1	f				1	2	2	2	3	4	5	6	6	2	1	3	3	7	6	6	2	2	f					1	1	1	2	3	4	5	5	6	2	1	1	3	7	7	6	6	hit		X	X	X	X	✓	✓	X	X	X	X	✓	X	X	X	✓	X	X	✓	X	✓	v								1	2	3	4		5	6	2		1	3		7	
t	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20																																																																																																																																																													
ref		1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6																																																																																																																																																													
f		1	2	3	4	4	4	5	6	2	1	1	3	7	6	6	2	1	1	3	3																																																																																																																																																													
f			1	2	3	3	3	4	5	6	2	2	1	3	7	7	6	2	2	1	1																																																																																																																																																													
f				1	2	2	2	3	4	5	6	6	2	1	3	3	7	6	6	2	2																																																																																																																																																													
f					1	1	1	2	3	4	5	5	6	2	1	1	3	7	7	6	6																																																																																																																																																													
hit		X	X	X	X	✓	✓	X	X	X	X	✓	X	X	X	✓	X	X	✓	X	✓																																																																																																																																																													
v								1	2	3	4		5	6	2		1	3		7																																																																																																																																																														

t	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
ref		1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
f		1	2	3	4	4	4	5	6	6	6	6	6	7	7	7	7	1	1	1	1
f			1	2	3	3	3	3	3	3	3	3	3	6	6	6	6	6	6	6	6
f				1	2	2	2	2	2	2	2	2	2	3	3	3	3	3	3	3	3
f					1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2
hit		X	X	X	X	✓	✓	X	X	✓	✓	✓	✓	X	✓	✓	✓	X	✓	✓	✓
v								4	5					1				7			

- Total references: 20
- Total distinct references: 7
- Hits: 12
- Faults: 8
- **Hit rate:** $12/20 = 60\%$
- **Fault rate:** $8/20 = 40\%$

iii. LRU algorithm

- Total frames: 4
- Algorithm: LRU
- Reference string length: 20 references
- String: 1 2 3 4 2 1 5 6 2 1 2 3 7 6 3 2 1 2 3 6

Solution visualization

t	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
ref		1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
f		1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
f			1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3
f				1	2	3	4	2	1	5	6	6	1	2	3	7	6	3	3	1	2
f					1	1	3	4	2	1	5	5	6	1	2	2	7	6	6	6	1
hit		X	X	X	X	✓	✓	X	X	✓	✓	✓	X	X	X	✓	✓	X	✓	✓	✓
v								3	4					5	6	1			7		

- Total references: 20
- Total distinct references: 7
- Hits: 10
- Faults: 10
- **Hit rate:** $10/20 = 50\%$
- **Fault rate:** $10/20 = 50\%$

b)	State features of Cloud OS. Enlist its advantages and disadvantages.
Ans	Cloud OS is an operating system that runs on cloud computing architecture. It is designed to provide a platform for running applications and storing data in the cloud. Some of the features of Cloud OS are:

		<ul style="list-style-type: none"> • Scalability: Cloud OS can scale up or down based on the needs of the user. • Flexibility: Cloud OS can be accessed from anywhere with an internet connection. • Cost-effective: Cloud OS eliminates the need for expensive hardware and software. • Reliability: Cloud OS is designed to be highly available and fault-tolerant. • Security: Cloud OS provides secure access to data and applications. <p>Advantages of Cloud OS include:</p> <ul style="list-style-type: none"> • Lower costs: Cloud OS eliminates the need for expensive hardware and software. • Scalability: Cloud OS can scale up or down based on the needs of the user. • Flexibility: Cloud OS can be accessed from anywhere with an internet connection. • Reliability: Cloud OS is designed to be highly available and fault-tolerant. • Automatic updates: Cloud OS automatically updates itself, which means that users always have access to the latest features and security updates. <p>Disadvantages of Cloud OS include:</p> <ul style="list-style-type: none"> • Internet dependency: Cloud OS requires an internet connection to function. • Security concerns: Cloud OS is vulnerable to security breaches and data loss. • Limited control: Users have limited control over the infrastructure and software used by Cloud OS. • Vendor lock-in: Users may be locked into a specific vendor's cloud platform, which can make it difficult to switch to another platform.
5	a)	What is demand paging? Discuss the hardware support required to support demand paging.
	Ans	<ul style="list-style-type: none"> • Demand paging is a memory management technique used in operating systems to improve memory usage and system performance. • It is a technique used in virtual memory systems where pages enter main memory only when requested or needed by the CPU. • In demand paging, the operating system loads only the necessary pages of a program into memory at runtime, instead of loading the entire program into memory at the start. • A page fault occurs when the program needs to access a page that is not currently in memory. • The operating system then loads the required pages from the disk into memory and updates the page tables accordingly. • This process is transparent to the running program, and it continues to run as if the page had always been in memory.

		<ul style="list-style-type: none"> • Demand paging requires several types of hardware support: <ul style="list-style-type: none"> - A Translation Buffer (TB): It is a small, fast lookup table that maps virtual addresses to physical addresses. It is used to speed up the address translation process. - An address translation mechanism: It is used to translate virtual addresses to physical addresses. It is usually implemented using a page table. - Page table entries with disk addresses: These entries contain the disk address of the page when it is not in memory. - The ability to detect a page fault: When a page fault occurs, the operating system must be able to detect it and respond appropriately. <p>Hardware support for demand paging is essential for efficient memory management and system performance.</p>
	b)	What is Threading and Multithreading? Explain importance of Multithreading.
	Ans	<ul style="list-style-type: none"> • Threading is a technique used in computer science to allow a program to perform multiple tasks concurrently. • A thread is a lightweight process that can run concurrently with other threads within the same process. • Multithreading is the ability of an operating system to support multiple threads of execution within a single process. • Multithreading is important because it can improve the performance and efficiency of a program by utilizing the available CPU resources more effectively. • By executing multiple threads concurrently, it can take advantage of parallelism and reduce overall execution time. • Multithreading can enhance responsiveness in applications that involve user interaction. • It can also improve the scalability of a program by allowing it to take advantage of multiprocessor architectures. • The benefits of multithreading can be broken down into four major categories : <ol style="list-style-type: none"> i. Responsiveness: Multithreading in an interactive application may allow a program to continue running even if a part of it is blocked or is performing a lengthy operation, thereby increasing responsiveness to the user. ii. Resource Sharing: Threads share the memory and the resources of the process to which they belong by default. The benefit of sharing code and data is that it allows an application to have several threads of activity within the same address space. iii. Economy: Allocating memory and resources for process creation is a costly job in terms of time and space. Since threads share memory with the process it belongs, it is more economical to create and context switch threads. iv. Scalability: The benefits of multi-programming greatly increase in case of multiprocessor architecture, where threads may be running parallel on multiple processors.
6	a)	Necessary conditions for deadlock
	Ans	<ul style="list-style-type: none"> • Deadlock is a situation in which two or more processes are blocked because each process is holding a resource and waiting for another resource acquired by some other process.

		<ul style="list-style-type: none"> The four necessary conditions for a deadlock situation are: <ul style="list-style-type: none"> Mutual Exclusion: At least one resource must be held in a non-shareable mode, which means that only one process can use the resource at any given time. Hold and Wait: A process must be holding at least one resource while waiting for other processes to release resources that are currently held by other processes. No Preemption: Resources cannot be preempted, which means that a resource cannot be taken away from a process until the process has completed its task. Circular Wait: A circular chain of processes exists, where each process is waiting for a resource held by the next process in the chain. If all four conditions are present simultaneously, a deadlock can occur. Deadlocks are undesirable in operating systems because they waste resources and have a negative impact on overall system performance and responsiveness. To prevent deadlocks, various resource allocation and process scheduling algorithms, such as deadlock detection and avoidance, are employed.
	b)	RAID levels
	Ans	<ul style="list-style-type: none"> RAID (Redundant Array of Independent Disks) is a technique that makes use of a combination of multiple disks instead of using a single disk for increased performance, data redundancy, or both. RAID levels are used to describe the different structural approaches and functions of the RAID. The most commonly used RAID levels are RAID 0, RAID 1, RAID 5, RAID 6, and RAID 10. <p>Here is a brief description of each RAID level:</p> <ul style="list-style-type: none"> i. RAID 0 (Striping): <ul style="list-style-type: none"> Blocks are "stripped" across disks. It is easy to implement and utilizes the storage capacity in a better way. However, it is not a good choice for a critical system as a single drive loss can result in the complete failure of the system. ii. RAID 1 (Mirroring): <ul style="list-style-type: none"> More than one copy of each block is stored in a separate disk. Thus, every block has two (or more) copies, lying on different disks. It is capable of reliability, but it is not an economical option as it requires twice the number of disks. iii. RAID 5 (Block-Level Striping with Distributed Parity): <ul style="list-style-type: none"> It distributes parity among all the disks in the array. It is a good choice for systems that require high read performance and moderate write performance. It is also cost-effective as it requires only one disk for parity. iv. RAID 6 (Block-Level Striping with two Parity Bits):

		<ul style="list-style-type: none"> ○ It is similar to RAID 5, but it uses two parity bits instead of one. ○ It can tolerate the failure of two disks simultaneously, making it more reliable than RAID 5. However, it requires more disk space for parity information. <p>v. RAID 10 (Striping of Mirrors):</p> <ul style="list-style-type: none"> ○ It is a combination of RAID 0 and RAID 1. ○ It provides both performance and redundancy by striping data across mirrored pairs. ○ It requires at least four disks and is more expensive than other RAID levels.
	c)	Disk Scheduling
	Ans	<ul style="list-style-type: none"> • Disk scheduling is a process used by operating systems to schedule I/O requests arriving for the disk. • The objective of disk scheduling is to minimize the time it takes to access data on the disk and to minimize the time it takes to complete a disk access request. • There are several disk scheduling algorithms, each with its own advantages and disadvantages. • Some of the commonly used disk scheduling algorithms are: <ul style="list-style-type: none"> i. First-Come, First-Serve (FCFS): <ul style="list-style-type: none"> ○ It is the simplest disk scheduling algorithm where the requests are addressed in the order they arrive in the disk queue. ii. Shortest Seek Time First (SSTF): <ul style="list-style-type: none"> ○ It selects the request that requires the least movement of the disk arm from its current position. iii. SCAN (Elevator Algorithm): <ul style="list-style-type: none"> ○ It moves the disk arm from one end of the disk to the other, servicing requests in between in a first-come, first-served manner. iv. C-SCAN (Circular SCAN): <ul style="list-style-type: none"> ○ It is similar to SCAN, but the disk arm moves only in one direction, servicing requests in that direction only. ○ It is similar to SCAN, but the disk arm moves only to the last request in each direction, instead of moving to the end of the disk. v. C-LOOK: <ul style="list-style-type: none"> ○ It is similar to LOOK, but the disk arm moves only in one direction, servicing requests in that direction only. <p>Each algorithm has its own strengths and weaknesses, and the choice of algorithm depends on the specific requirements of the system.</p>
	d)	Real Time Operating System
	Ans	<ul style="list-style-type: none"> • An RTOS is a type of operating system that is designed to handle tasks that have strict time constraints or deadlines. • An RTOS can process data and events quickly and reliably, without delays or interruptions. • An RTOS is used for applications that require high accuracy, precision, and responsiveness, such as industrial control systems, flight control systems,

		<p>medical devices, and autonomous vehicles.</p> <ul style="list-style-type: none"> • An RTOS can be classified into different types based on the following criteria: <ul style="list-style-type: none"> ○ Hard vs. Soft vs. Firm RTOS: <ul style="list-style-type: none"> - A hard RTOS guarantees that critical tasks are completed within a fixed time limit, otherwise the system may fail. - A soft RTOS tries to meet the deadlines but can tolerate some delays without serious consequences. - A firm RTOS also has deadlines but missing them may only reduce the quality of the service. ○ Deterministic vs. Non-deterministic RTOS: <ul style="list-style-type: none"> - A deterministic RTOS ensures that all tasks and processes execute with predictable timing, which is important for applications that require high accuracy and precision. - A non-deterministic RTOS may have variable timing, depending on the workload and the scheduling algorithm. ○ Event-driven vs. Time-sharing RTOS: <ul style="list-style-type: none"> - An event-driven RTOS switches tasks only when a higher-priority event occurs, which minimizes the overhead and the response time. - A time-sharing RTOS switches tasks on a regular clock interrupt, which allows for smoother multitasking and fairer resource allocation. • An RTOS uses a scheduling algorithm to decide which task to execute next, based on the priority, the deadline, the arrival time, and the execution time of each task. • Some common scheduling algorithms are First-In-First-Out (FIFO), Round Robin, Earliest Deadline First (EDF), Least Laxity First (LLF), and Rate Monotonic (RM).
	e)	Deadlock avoidance
	Ans	<ul style="list-style-type: none"> • Deadlock avoidance is a technique used in operating systems to prevent the situation where two or more processes are unable to proceed because each is waiting for one of the others to release a resource. • Deadlock avoidance requires the operating system to have some knowledge of the resource requirements and availability of each process, and to allocate resources in a way that ensures that no deadlock can occur. <p>Some of the methods used for deadlock avoidance are:</p> <ol style="list-style-type: none"> Resource Allocation Graph: <ul style="list-style-type: none"> ○ It is a graphical representation of the processes, resources, and their allocation status. ○ A deadlock can be avoided by ensuring that the graph does not have any cycles, which indicate a circular dependency among processes. Banker's Algorithm: <ul style="list-style-type: none"> ○ It is a mathematical model that simulates a bank that grants loans to customers. ○ The bank keeps track of the total resources, the allocated resources,

		<p>and the maximum resources needed by each customer.</p> <ul style="list-style-type: none"> ○ The bank only grants a loan request if it can ensure that the system will remain in a safe state, where all customers can finish their tasks without causing a deadlock. <p>iii. Safe State and Unsafe State:</p> <ul style="list-style-type: none"> ○ A safe state is a state where the system can allocate resources to each process in some order and still avoid a deadlock. ○ An unsafe state is a state where the system cannot guarantee the avoidance of deadlock. ○ A deadlock avoidance algorithm must ensure that the system never enters an unsafe state.
	f)	Process Control Block
Ans		<p>Here are some points about Process Control Block (PCB) in operating system:</p> <ul style="list-style-type: none"> ○ A PCB is a data structure that is used by an operating system to store all the information about a process. ○ A process is a program that is being executed by the CPU. ○ A PCB contains information such as the process ID, the process state, the program counter, the CPU registers, the memory limits, the open files list, the scheduling information, and the accounting information. ○ A PCB is created when a process is created and deleted when a process is terminated. ○ A PCB helps the operating system to manage and control the processes efficiently and effectively. ○ A PCB is also used for context switching, which is the process of saving the state of the current process and restoring the state of the next process to be executed. ○ A PCB is also used for inter-process communication, which is the process of exchanging data and messages between processes. ○ A PCB is also used for fault tolerance, which is the ability of a system to recover from errors and failures. <div data-bbox="371 1391 1364 1944" data-label="Diagram"> <p style="text-align: center;">PROCESS CONTROL BLOCK (PCB)</p> <pre> graph TD subgraph PCB [PCB Diagram] direction TB P1[Pointer] P2[Process state] P3[Process ID] P4[Program counter] P5[Register's] P6[Memory Limits] P7[Accounting] P8[List of open file] end FPF[Facilities for process] --> P1 UID[Unique ID] --> P3 NPT[Next Program that run] --> P4 LI[Log INFO about Process] --> P7 </pre> <p style="text-align: center;">PCB Diagram</p> </div>