VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

REG.NO.:
SLOT: G2+TG2

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
CONTINUOUS ASSESSMENT TEST - I
WINTER SEMESTER 2025-2026

Programme Name & Branch : **B.Tech. CSE**
Course Code and Course Name : **BACSE106 – Operating Systems**
Faculty Name(s) : **Vijayasherly V, Vijaya Kumar K, Sivakumar N, Deepa K, Suresh P, Mohankumar B, Berlin M A, Ezhil Arasi V, Naga Priyadarsini R, Bhawana Tyagi, Raghul J, Saraswathi U, Sivakumar T, Arthi M, Nagaraja Rao A, Satyajit Mohanty Karthika M S**
Class Number(s) : **1587, 1562, 1566, 1576, 2404, 1586, 1570, 6373, 1589, 1591, 1596, 1602, 1617, 1611, 1567, 1635, 1593**
Date of Examination : **02.02.2026**
Exam Duration : **90 minutes** **Maximum Marks: 50**

General instruction(s):

- Answer All Questions
- M - Max mark; CO – Course Outcome; BL – Blooms Taxonomy Level
- Course Outcomes
  - CO1: Understand the structure, functionalities, and foundational concepts of operating systems.
  - CO2: Apply process scheduling algorithms and analyze thread models and deadlock-handling methods.

Q. No	Question	Module	Marks	CO	BL
1.	A multinational technology company is designing a next generation operating system intended to support three different environments using a common core design: • Environment A: A safety critical embedded control system used in medical and industrial equipment, where system crashes are unacceptable and fault isolation is essential. • Environment B: An academic operating system for universities, aimed at helping students clearly understand how operating system components interact internally. • Environment C: A high performance general purpose operating system for servers and desktops that must support frequent updates, device drivers, and third party extensions without requiring a system reboot. The design team is considering the following operating system structures: Monolithic, Layered, Microkernel, and Modular. As a system architect, analyze the above case study and answer the following: a) Explain how each of the four operating system structure approaches organizes kernel services, using neatly labeled diagrams, and discuss the advantages and disadvantages of each method. (7 Marks) b) For each environment (A, B, and C), select the most appropriate operating system structure from the list and justify your choice	1	10	CO1	BL2

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
CONTINUOUS ASSESSMENT TEST - I
WINTER SEMESTER 2025-2026

---

	based on performance, reliability, security, maintainability, and extensibility. (3 Marks)				
2.	A software development company is building a multi user operating system to support applications such as file editors, media players, networked applications, and secure system utilities. During system execution, the following situations occur: • A user starts a new application, terminates it forcibly, and waits for a child process to complete execution. • An application creates a new file, writes data to it, reads the stored content, and then deletes the file. • A printer and a disk drive are accessed by multiple programs, and the operating system must allocate and release these devices safely. • The operating system periodically retrieves system time, process status, and resource usage information for monitoring and logging purposes. • Two applications running on different machines exchange messages and synchronize their execution over a network. • A secure banking application restricts access to sensitive files and ensures that only authorized users can perform critical operations. As an operating system designer, analyze the above case study and answer the following: a) Identify the appropriate type of system call used in each situation listed above. (3 Marks) b) For each identified system call type, explain its importance in operating system functioning with suitable examples. (5 Marks) c) Describe the system call mechanism, explaining how a user program invokes a system call and how control is transferred between user mode and kernel mode. (2 Marks)	1	10	CO1	BL2
3.	a) Discuss the data structure used by the operating system to store and manage information about each process during execution. Explain the contents of this structure and justify how it supports process scheduling and context switching. Illustrate your answer with a neat labeled diagram.	2	5	CO2	BL2
	b) Analyze the various multithreading models in operating systems and compare user level threads with kernel level threads. (3 + 2)	2	5		
4.	Consider a system consisting of six processes P1, P2, P3, P4, P5, and P6 arriving at times 0, 1, 2, 4, 5, and 6 milliseconds respectively. The total execution time of these processes, which includes both CPU burst time and I/O burst time, is 20, 18, 15, 10, 6, and 12 milliseconds respectively. Each process spends 20% of its total execution time performing I/O operations and the remaining 80% performing CPU computation. The CPU burst time for each process is obtained by calculating 80% of the total execution time and rounding the result to the nearest whole	2	10	CO2	BL3

---

	number. The operating system schedules the processes using the following two CPU scheduling algorithms by considering only the CPU burst time of each process: • Shortest Remaining Time First (SRTF) • Round Robin scheduling with a time quantum of 5 milliseconds For each scheduling algorithm: a) Draw the Gantt chart (2 + 2 Marks) b) Calculate the average turnaround time (1 + 1 Mark) c) Calculate the average waiting time (1 + 1 Mark) d) Determine the response time of each process (1 + 1 Mark)				
5.	A computer system uses Multilevel Queue CPU Scheduling with three priority queues. The queues are arranged in descending order of priority, where a lower queue number indicates higher priority. Each queue follows a different CPU scheduling algorithm. Processes are permanently assigned to a queue and cannot move between queues. The scheduler always selects a process from the highest priority non-empty queue, and scheduling is preemptive between queues. The scheduling policies for the queues are as follows: • Queue Q1 uses Preemptive Priority Scheduling • Queue Q2 uses First Come First Served (FCFS) • Queue Q3 uses Round Robin Scheduling with a time quantum of 3 milliseconds The details of the processes are given in the table below.	2	10	CO4	BL3

Process ID	Arrival Time (ms)	Burst Time (ms)	Priority	Queue
P1	0	10	2	Q2
P2	1	6	1	Q1
P3	2	8	3	Q3
P4	3	4	1	Q1
P5	4	5	2	Q2
P6	4	9	2	Q1
P7	5	13	4	Q3
P8	5	4	1	Q2
P9	7	1	5	Q3

a) Draw the Gantt chart for the given set of processes using Multilevel Queue Scheduling. (4 Marks)
b) Calculate the Waiting Time, Turnaround Time, and Response Time for each process. (3 Marks)
c) Determine the number of context switches that occur during execution and explain their causes. (3 Marks)

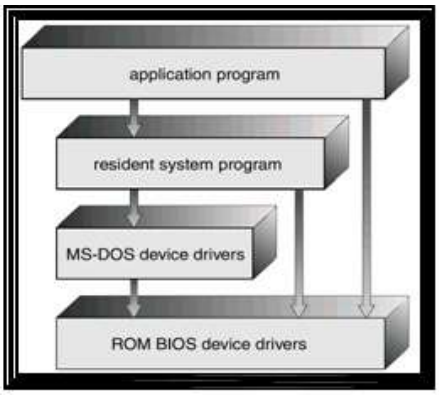
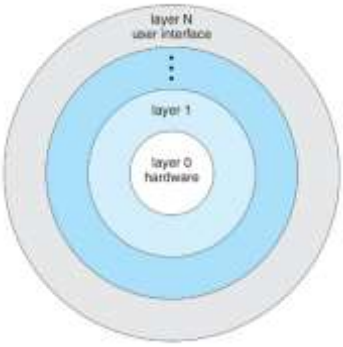
****

REG.NO.:

SLOT: G2+TG2

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
CONTINUOUS ASSESSMENT TEST - I
WINTER SEMESTER 2025-2026

Answer Key

Q. No	Question
1.	A multinational technology company is designing a next generation operating system intended to support three different environments using a common core design: • Environment A: A safety critical embedded control system used in medical and industrial equipment, where system crashes are unacceptable and fault isolation is essential. • Environment B: An academic operating system for universities, aimed at helping students clearly understand how operating system components interact internally. • Environment C: A high performance general purpose operating system for servers and desktops that must support frequent updates, device drivers, and third party extensions without requiring a system reboot. The design team is considering the following operating system structures: Monolithic, Layered, Microkernel, and Modular. As a system architect, analyze the above case study and answer the following: a) Explain how each of the four operating system structure approaches organizes kernel services, using neatly labeled diagrams, and discuss the advantages and disadvantages of each method. (6 Marks) b) For each environment (A, B, and C), select the most appropriate operating system structure from the list and justify your choice based on performance, reliability, security, maintainability, and extensibility. (4 Marks) **Answer** **a) OS Structure Approaches (6 Marks)** ***Monolithic:*** In a monolithic kernel, all operating system services such as process management, memory management, file systems, and device drivers run in kernel mode as a single large program. Communication between services occurs through direct function calls, making the system fast but less reliable.  Pros • High performance due to minimal overhead • Efficient system calls Cons • Poor fault isolation • Difficult to debug and maintain • Kernel crash affects entire system  ***Layered:*** The OS is divided into hierarchical layers, where each layer uses services of the lower layer and provides services to the higher layer. Each layer has a clearly defined function. Pros • Clear modularity • Easy to understand and debug

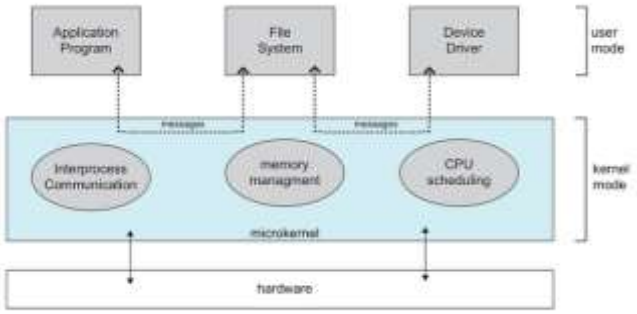
- Good for teaching

Cons

- Performance overhead due to layer traversal
- Difficult to define perfect layers

Microkernel:

The microkernel keeps only essential services in kernel mode, such as inter process communication, scheduling, and basic memory management. Other services run in user space as servers.



Pros

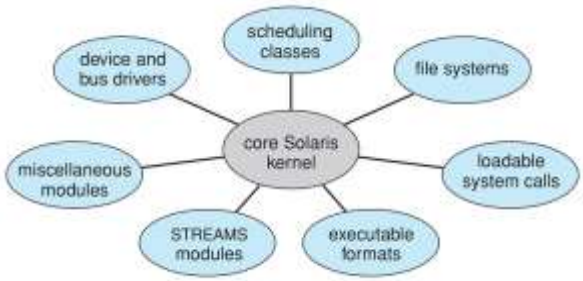
- High reliability and fault tolerance
- Better security
- Easier to extend

Cons

- Performance overhead due to message passing
- Complex IPC mechanisms

Modular:

The modular kernel uses a small core kernel with loadable modules for services such as file systems and device drivers. Modules can be added or removed at runtime.



Pros

- Flexible and extensible
- Performance close to monolithic kernel
- Easier maintenance

Cons

- Bugs in modules can still affect kernel
- More complex than layered systems

b) Best OS Structure Selection for Each Environment (4 Marks)

Environment A: Safety Critical Embedded System

Best Choice: ***Microkernel***

Justification

- Fault isolation ensures failures do not crash entire system
- High reliability and security
- Ideal for medical and industrial applications

Environment B: Academic Teaching OS

Best Choice: ***Layered***

Justification

- Clear separation of responsibilities
- Easy for students to understand internal working

	• Simplifies debugging and learning Environment C: High Performance General Purpose OS Best Choice: ***Modular Kernel*** Justification • Supports dynamic loading of drivers and services • High performance close to monolithic systems • Used in real systems like Linux
2.	A software development company is building a multi user operating system to support applications such as file editors, media players, networked applications, and secure system utilities. During system execution, the following situations occur: • A user starts a new application, terminates it forcibly, and waits for a child process to complete execution. • An application creates a new file, writes data to it, reads the stored content, and then deletes the file. • A printer and a disk drive are accessed by multiple programs, and the operating system must allocate and release these devices safely. • The operating system periodically retrieves system time, process status, and resource usage information for monitoring and logging purposes. • Two applications running on different machines exchange messages and synchronize their execution over a network. • A secure banking application restricts access to sensitive files and ensures that only authorized users can perform critical operations. As an operating system designer, analyze the above case study and answer the following: a) Identify the appropriate type of system call used in each situation listed above. (3 Marks) b) For each identified system call type, explain its importance in operating system functioning with suitable examples. (5 Marks) c) Describe the system call mechanism, explaining how a user program invokes a system call and how control is transferred between user mode and kernel mode. (2 Marks) **Answer:** **a) Identification of Appropriate System Call Types (3 Marks)** Based on the given case study, the operating system uses different system call categories to handle each situation efficiently.

Scenario Description	System Call Type	Justification
Creating a new process, terminating it, and waiting for a child process to complete	**Process Control**	These operations manage the lifecycle of processes including creation, execution, synchronization, and termination
Creating, writing, reading, and deleting files	**File Management**	File related operations involve creating, accessing, modifying, and removing files stored on secondary memory

Allocating and releasing printer and disk devices	**Device Management**	Device allocation, I O control, and release are handled through device specific system calls
Retrieving system time, process status, and resource usage	**Information Maintenance**	These system calls maintain and provide system level information needed for monitoring and accounting
Message exchange and synchronization between applications over a network	**Communication**	Inter process communication and networking services are provided using communication system calls
Restricting access to sensitive files and enforcing user authorization	**Protection**	Protection system calls ensure access control, authentication, and permission enforcement

b) Importance of Each System Call Type (5 Marks)

1. Process Control System Calls

Process control system calls are essential for managing program execution. They allow the operating system to create new processes, execute programs, wait for process completion, and terminate processes safely. Without these system calls, multitasking and synchronization between parent and child processes would not be possible. Examples include fork(), exec(), wait(), and exit().

2. File Management System Calls

File management system calls provide a structured way to store and retrieve data permanently. They ensure data integrity, support file sharing, and allow applications to perform read and write operations securely. These system calls also manage file permissions and directory structures. Examples include open(), read(), write(), close(), and unlink().

3. Device Management System Calls

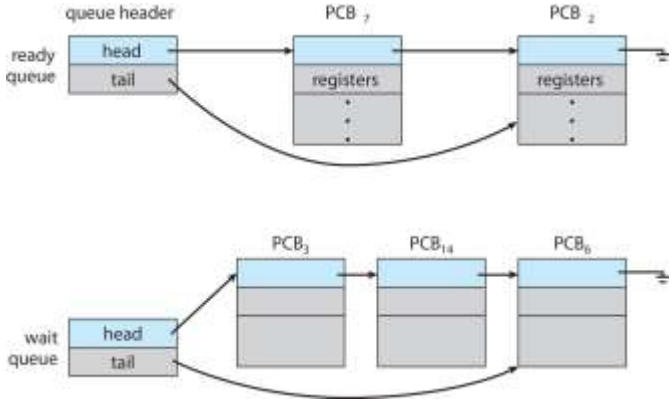
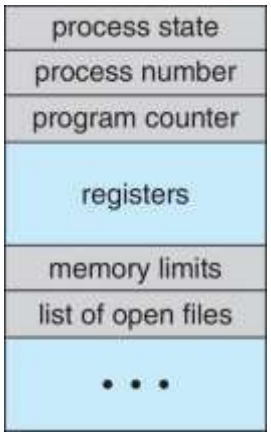
Device management system calls enable controlled access to hardware devices such as printers and disks. They help prevent conflicts when multiple processes request the same device and ensure proper allocation and release. These calls abstract hardware details from applications, improving portability. Examples include ioctl(), read(), and write().

4. Information Maintenance System Calls

Information maintenance system calls allow the operating system to track and report system status. They are used for time management, performance monitoring, and resource accounting. These calls help system administrators and applications make informed decisions. Examples include gettimeofday(), getpid(), and stat().

5. Communication System Calls

Communication system calls support data exchange between processes either on the same system or across networks. They enable synchronization, coordination, and distributed

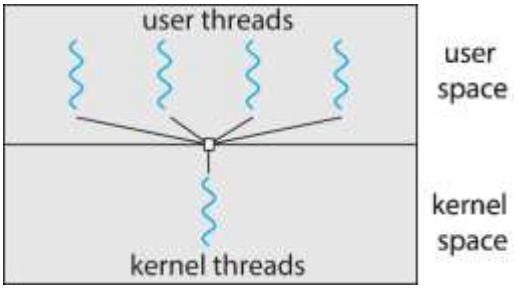
	computing. Without these calls, modern networked applications would not function. Examples include pipe(), socket(), send(), and recv(). **6. Protection System Calls** Protection system calls ensure system security by controlling access to resources. They authenticate users, enforce permissions, and prevent unauthorized operations. These calls are critical for secure applications such as banking systems. Examples include chmod(), chown(), and setuid(). **c) System Call Mechanism (2 Marks)** A system call acts as an interface between a user level program and the operating system kernel. 1. When a user program needs a privileged operation, it invokes a system call using a predefined API function. 2. The call triggers a software interrupt or trap, transferring control from user mode to kernel mode. 3. The operating system kernel verifies the request, checks permissions, and executes the required service. 4. After execution, the kernel returns the result to the user program and switches control back to user mode. This mechanism ensures system protection, since user programs cannot directly access hardware or critical kernel data structures.
3.	a) Discuss the data structure used by the operating system to store and manage information about each process during execution. Explain the contents of this structure and justify how it supports process scheduling and context switching. Illustrate your answer with a neat labeled diagram. **Answer:** **Process Control Block (PCB) (Explanation – 3 marks, Diagram – 2 marks)** Information associated with each process (also called **task control block**)   ▪ Process state – running, waiting, etc. ▪ Program counter – location of instruction to next execute ▪ CPU registers – contents of all process-centric registers ▪ CPU scheduling information- priorities, scheduling queue pointers ▪ Memory-management information – memory allocated to the process

- Accounting information – CPU used, clock time elapsed since start, time limits
- I/O status information – I/O devices allocated to process, list of open files

b) Analyze the various multithreading models in operating systems and compare user level threads with kernel level threads. (3 + 2)

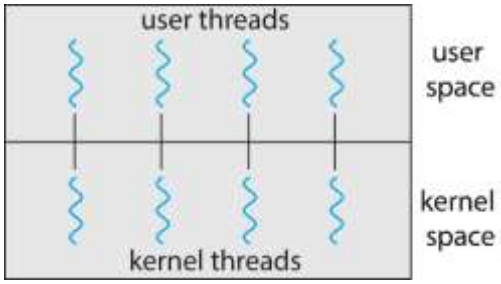
Answer:

1. Many-to-One (1 Mark)



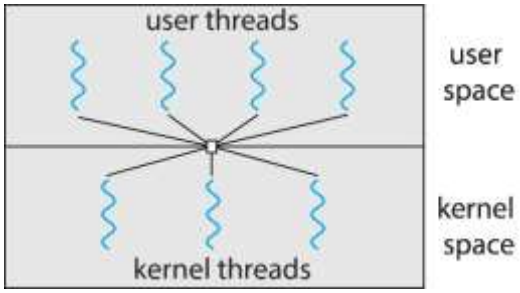
- Many user-level threads mapped to single kernel thread
- One thread blocking causes all to block
- Multiple threads may not run in parallel on multicore system because only one may be in kernel at a time
- Few systems currently use this model
- Examples:
  - **Solaris Green Threads**
  - **GNU Portable Threads**

2. One-to-One (1 Mark)



- Each user-level thread maps to kernel thread
- Creating a user-level thread creates a kernel threa d
- More concurrency than many-to-one
- Number of threads per process sometimes restricted due to overhead
- Examples
  - Windows
  - Linux

3. Many-to-Many Model (1 Mark)



- Allows many user level threads to be mapped to many kernel threads
- Allows the operating system to create a sufficient number of kernel threads
- Windows with the *ThreadFiber* package
- Otherwise not very common

Compare user level threads with kernel level threads. (2 Marks)

No.	Parameters	User Level Thread	Kernel Level Thread
1.	Implemented by	User threads are implemented by users.	Kernel threads are implemented by Operating System (OS).
2.	Recognize	Operating System doesn't recognize user level threads.	Kernel threads are recognized by Operating System.
3.	Implementation	Implementation of User threads is easy.	Implementation of Kernel thread is complicated.
4.	Context switch time	Context switch time is less.	Context switch time is more.
5.	Hardware support	Context switch requires no hardware support.	Hardware support is needed.
6.	Blocking operation	If one user level thread performs blocking operation then entire process will be blocked.	If one kernel thread perform blocking operation then another thread can continue execution.
7.	Multithreading	Multithread applications cannot take advantage of multiprocessing.	Kernels can be multithreaded.
8.	Creation and Management	User level threads can be created and managed more quickly.	Kernel level threads take more time to create and manage.
9.	Operating System	Any operating system can support user-level threads.	Kernel level threads are operating system-specific.

4. Consider a system consisting of six processes P1, P2, P3, P4, P5, and P6 arriving at times 0, 1, 2, 4, 5, and 6 milliseconds respectively. The total execution time of these processes, which includes both CPU burst time and I/O burst time, is 20, 18, 15, 10, 6, and 12 milliseconds respectively. Each process spends 20% of its total execution time performing I/O operations and the remaining 80% performing CPU computation. The CPU burst time for each process is obtained by calculating 80% of the total execution time and rounding the result to the nearest whole number. The operating system schedules the processes using the following two CPU scheduling algorithms by considering only the CPU burst time of each process:

- Shortest Remaining Time First (SRTF)
- Round Robin scheduling with a time quantum of 5 milliseconds

For each scheduling algorithm:
a) Draw the Gantt chart (2 + 2 Marks)
b) Calculate the average turnaround time (1 + 1 Mark)
c) Calculate the average waiting time (1 + 1 Mark)
d) Determine the response time of each process (1 + 1 Mark)

Answer:

Process Details

Process	Arrival Time (ms)	Total Execution Time (ms)
P1	0	20
P2	1	18
P3	2	15
P4	4	10
P5	5	6
P6	6	12

CPU and I/O Time Calculation

Each process:

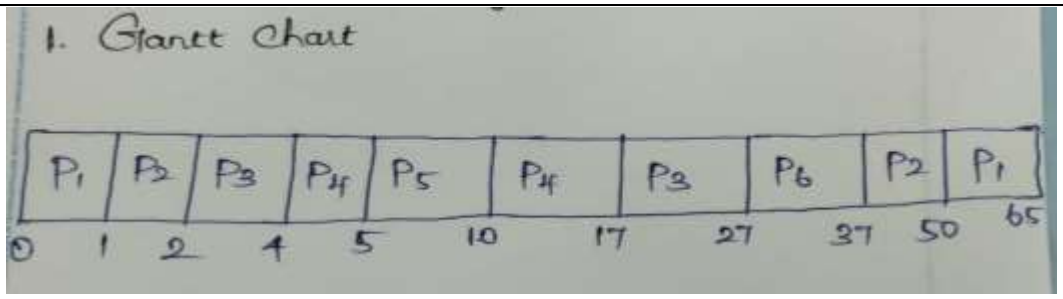
- **CPU time = 80%**
- **I/O time = 20%**

Process	CPU Time (ms)	I/O Time (ms)
P1	16	4
P2	14.4 ≈ 14	4
P3	12	3
P4	8	2
P5	5	1
P6	10	2

Note: Only CPU time is considered for scheduling

1. Gantt Chart - Shortest Remaining Time First (SRTF)

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
CONTINUOUS ASSESSMENT TEST - I
WINTER SEMESTER 2025-2026



SRTF Completion Times

Process	Arrival	CPU Burst Time	Completion Time (CT)
P1	0	16	65
P2	1	14	50
P3	2	12	27
P4	4	8	17
P5	5	5	10
P6	6	10	37

SRTF Time Calculations
Formulas

- Turnaround Time (TAT) = Completion Time − Arrival Time
- Waiting Time (WT) = Turnaround Time − CPU Burst Time
- Response Time (RT) = First CPU Time − Arrival Time

SRTF Results Table

Process	TAT	WT	RT
P1	65	49	0
P2	49	35	0
P3	25	13	0
P4	13	5	0
P5	5	0	0
P6	31	21	21

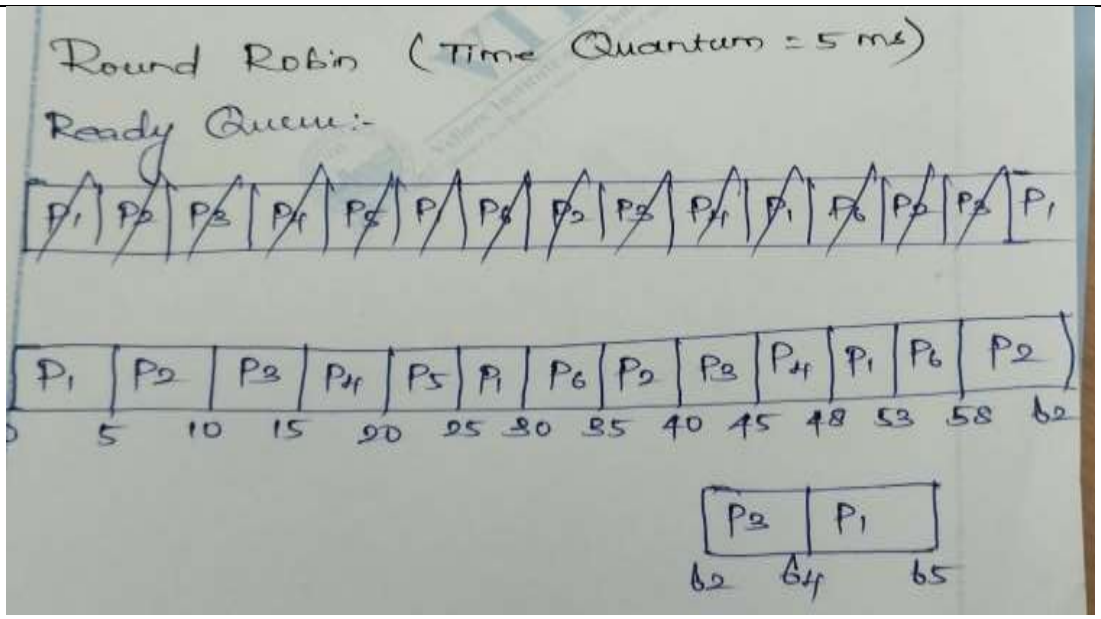
SRTF Averages
Average Turnaround Time = (65+49+25+13+5+31)/6 = 31.33 ms
Average Waiting Time = (49+35+13+5+0+21)/6 = 20.5 ms
Average Response Time = (0+0+0+0+0+21)/6 = 3.5 ms

Round Robin scheduling with a time quantum of 5 milliseconds

1. Gantt Chart

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
CONTINUOUS ASSESSMENT TEST - I
WINTER SEMESTER 2025-2026



RR Completion Times

Process	Arrival	CPU Burst Time	Completion Time (CT)
P1	0	16	65
P2	1	14	62
P3	2	12	64
P4	4	8	48
P5	5	5	25
P6	6	10	58

RR Time Calculations

Formulas

- Turnaround Time (TAT) = Completion Time − Arrival Time
- Waiting Time (WT) = Turnaround Time − CPU Burst Time
- Response Time (RT) = First CPU Time − Arrival Time

RR Results Table

Process	TAT	WT	RT
P1	65	49	0
P2	61	48	4
P3	62	52	8
P4	44	40	11
P5	20	20	15
P6	52	48	24

RR Averages

Average Turnaround Time = (65+61+62+44+20+52)/6 = 50.67 ms
Average Waiting Time = (49+48+52+10+20+48)/6 = 37.83 ms
Average Response Time = (0+4+8+11+15+24)/6 = 10.33 ms

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
CONTINUOUS ASSESSMENT TEST - I
WINTER SEMESTER 2025-2026

---

5. A computer system uses Multilevel Queue CPU Scheduling with three priority queues. The queues are arranged in descending order of priority, where a lower queue number indicates higher priority. Each queue follows a different CPU scheduling algorithm. Processes are permanently assigned to a queue and cannot move between queues. The scheduler always selects a process from the highest priority non-empty queue, and scheduling is preemptive between queues.

The scheduling policies for the queues are as follows:

- Queue Q1 uses Preemptive Priority Scheduling
- Queue Q2 uses First Come First Served (FCFS)
- Queue Q3 uses Round Robin Scheduling with a time quantum of 3 milliseconds

The details of the processes are given in the table below.

Process ID	Arrival Time (ms)	Burst Time (ms)	Priority	Queue
P1	0	10	2	Q2
P2	1	6	1	Q1
P3	2	8	3	Q3
P4	3	4	1	Q1
P5	4	5	2	Q2
P6	4	9	2	Q1
P7	5	13	4	Q3
P8	5	4	1	Q2
P9	7	1	5	Q3

a) Draw the Gantt chart for the given set of processes using Multilevel Queue Scheduling. (4 Marks)

b) Calculate the Waiting Time, Turnaround Time, and Response Time for each process. (3 Marks)

c) Determine the number of context switches that occur during execution and explain their causes. (3 Marks)

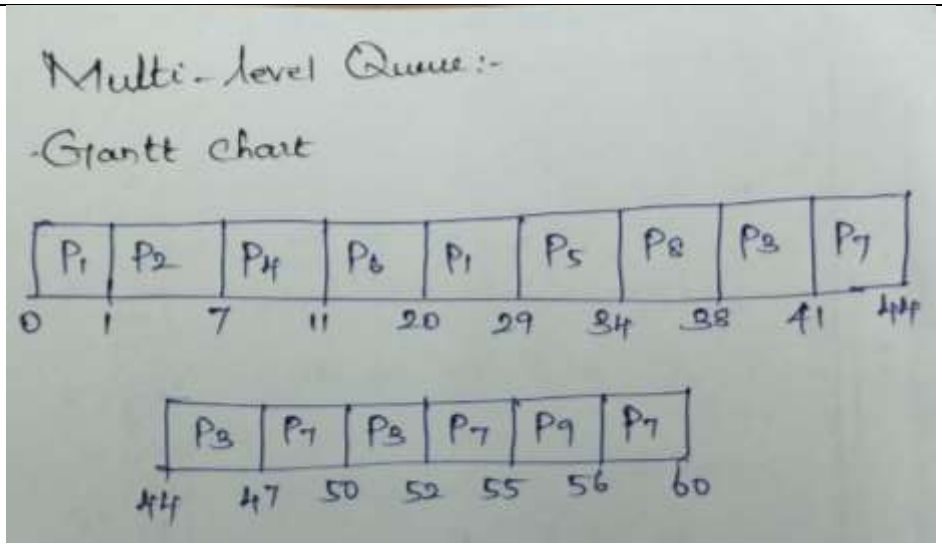
Answer:

Queue Rules

- Q1: Preemptive Priority Scheduling
  - Lower priority number → higher priority
  - If priorities are equal → FCFS, no preemption
- Q2: FCFS
- Q3: Round Robin, Time Quantum = 3 ms
- Preemptive between queues

a) Draw the Gantt chart for the given set of processes using Multilevel Queue Scheduling. (4 Marks)

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
CONTINUOUS ASSESSMENT TEST - I
WINTER SEMESTER 2025-2026



b) Calculate the average Waiting Time, Turnaround Time, and Response Time for each process.

Process	AT	BT	First CPU	CT	TAT	WT	RT
P1	0	10	0	29	29	19	0
P2	1	6	1	7	6	0	0
P3	2	8	38	52	50	42	36
P4	3	4	7	11	8	4	4
P5	4	5	29	34	30	25	25
P6	4	9	11	20	16	7	7
P7	5	13	41	60	55	42	36
P8	5	4	34	38	33	29	29
P9	7	1	55	56	49	48	48

c) Determine the number of context switches that occur during execution and explain their causes. (3 Marks)
Total Context Switches = 13
Reasons

- Arrival of higher-priority Q1 process
- Completion of a process
- Queue switching
- Time quantum expiration in Q3