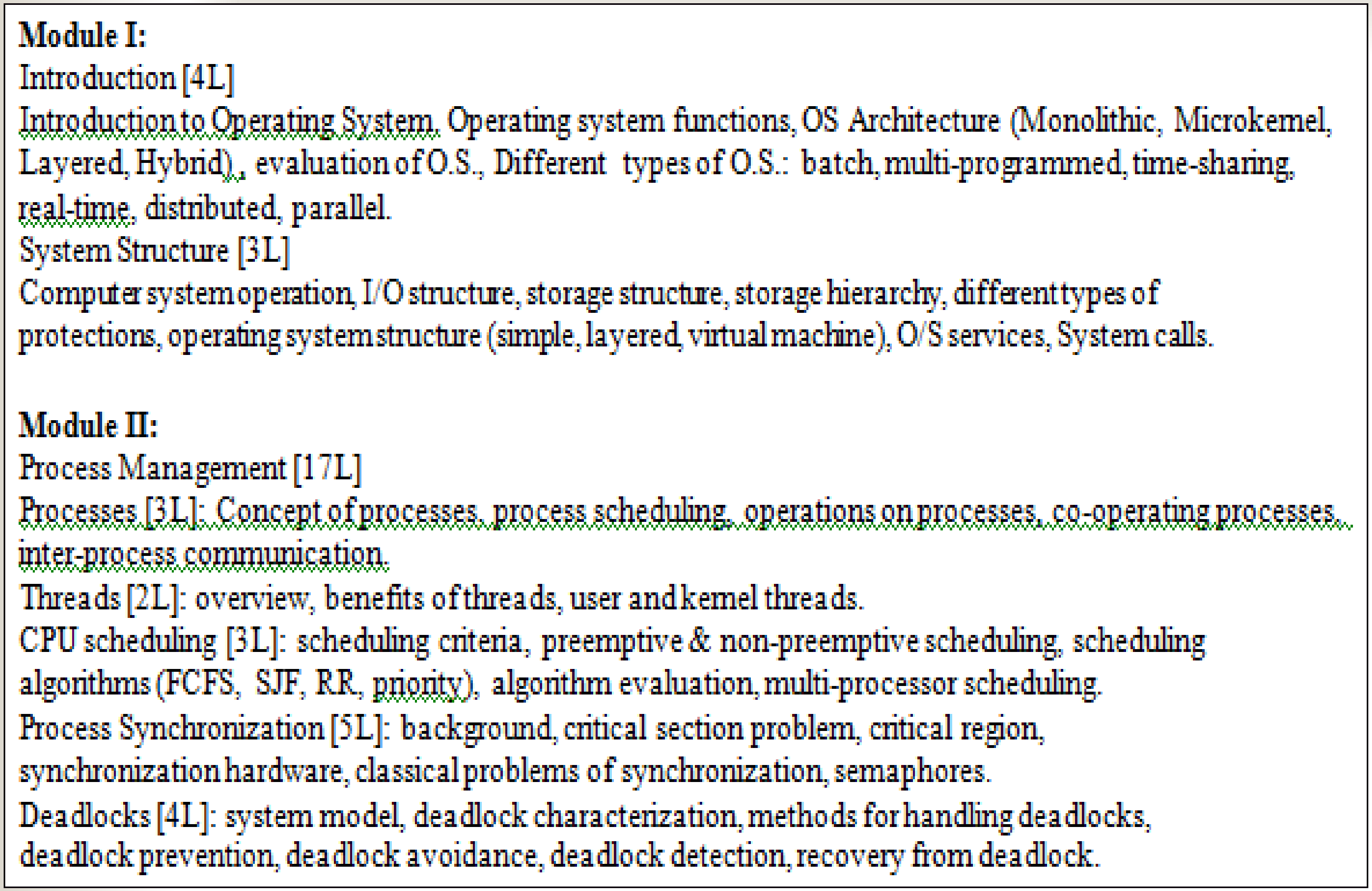
Internal-I : Guidelines

CSEN3103-Sec A, Prepared by Nilina Bera, CSE, HIT

Internal-I syllabus



Module I:

Introduction [4L]

Introduction to Operating System, Operating system functions, OS Architecture (Monolithic, Microkernel, Layered, Hybrid), evaluation of O.S., Different types of O.S.: batch, multi-programmed, time-sharing, real-time, distributed, parallel.

System Structure [3L]

Computer system operation, I/O structure, storage structure, storage hierarchy, different types of protections, operating system structure (simple, layered, virtual machine), O/S services, System calls.

Module II:

Process Management [17L]

Processes [3L]: Concept of processes, process scheduling, operations on processes, co-operating processes, inter-process communication.

Threads [2L]: overview, benefits of threads, user and kernel threads.

CPU scheduling [3L]: scheduling criteria, preemptive & non-preemptive scheduling, scheduling algorithms (FCFS, SJF, RR, priority), algorithm evaluation, multi-processor scheduling.

Process Synchronization [5L]: background, critical section problem, critical region, synchronization hardware, classical problems of synchronization, semaphores.

Deadlocks [4L]: system model, deadlock characterization, methods for handling deadlocks, deadlock prevention, deadlock avoidance, deadlock detection, recovery from deadlock.

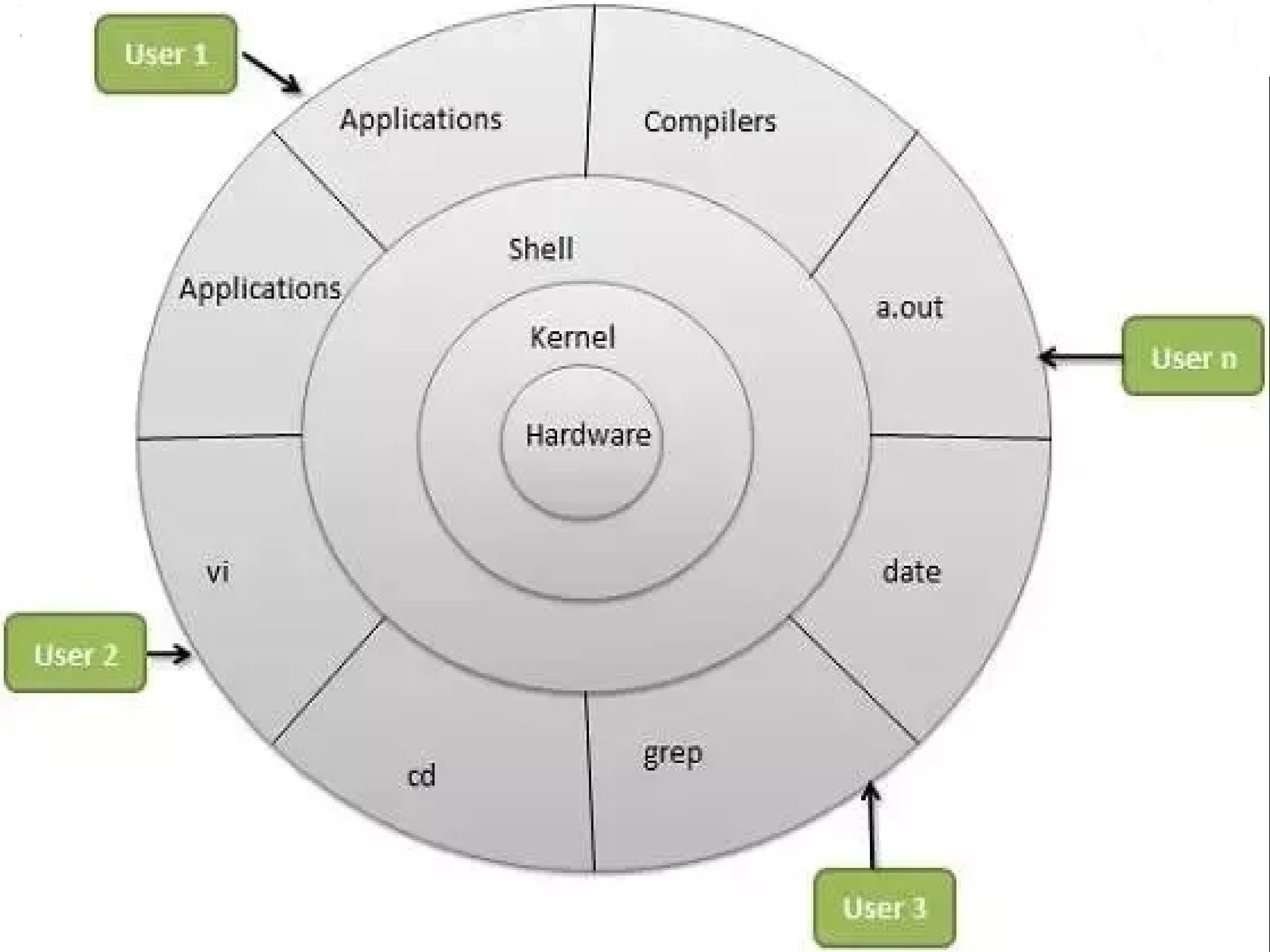
Guidelines: couple of questions & answers are discussed covering Module I and II. Still you need to study text book and reference copies to get a detail idea of OS

1. How is the kernel different from the OS? Why do we say that an operating system is more than a kernel?

- The kernel is the core of the operating system. The operating system as a whole consists of the kernel, supporting hardware management software (device drivers), various system libraries, and specialized utilities. Thus, OS is much more than a Kernel.

2. Why do we say utilities are a part of the OS, but not a part of the kernel? How does the kernel differ from utilities? Can we include all utilities in the kernel? Justify your answer.

- Utilities come with the operating system distribution. They collectively define the user interface to the computer system. Utilities provide a friendly and convenient environment for computer users. Kernel is the core of the operating system and it permanently resides in the main memory at runtime. Utilities are executed by users, and they can go on demand. We cannot include all utilities in the kernel because of memory space limitation and security.


User 1
Applications
Compilers
Shell
Applications
a.out
Kernel
User n
Hardware
vi
date
User 2
cd
grep
User 3

3. In what way are utilities similar to user applications? In what way are they different?

- They (utilities and user applications) all run in the user space and take operating system services to accomplish their tasks. Utilities are system dependent and they normally perform specialized system tasks. For example, a compiler translates a program, that is written in a higher level language, into machine code that can be executed by the system. User application: MS-Office, Code-block

4. What are the two operating modes of a processor and what is the advantage of having multiple operating modes?

- User and kernel modes. The operating modes help promoting protection in the OS – protecting the privileged resources from application processes.

5. Define a process. What is it used for? What is the difference between user and kernel processes?

Ans: A process is an active entity in operating systems' context and it is used by the operating system as a handle to manage one program execution. It is created for the single execution of a given application and it has a unique identifier in the system for its identification among other processes in the system. User processes, also called application processes, execute primarily applications and utilities. Whenever they need services from the operating system, they execute operating system programs. Kernel processes execute only operating system programs.

6. Why do processes interact with one another? What are the two classes of process interactions?

Ans: Some processes work together to perform application-specific tasks. From time to time they interact among themselves to know what they have collectively achieved. Processes also compete for shared resources and they need to coordinate their use of resources. The two classes of interactions are i) inter-process communication and ii) synchronization.

7. What is preemption? Why is processor preemption needed in an OS?

Ans: Preemption is the mechanism of forcefully taking away some allocated resource from a process. Processor preemption is used to multiplex the processor among processes that are ready to execute their program.

8. What is the difference between (processor) preemption and interruption? In what way are they similar?

Ans: They both break the current program execution and start a new execution flow. Interrupt breaks the current program execution and starts a new program execution in the same process context. Preemption also breaks the current program execution, but starts a new program execution in a different process context.

9. Compare and contrast microkernel and monolithic kernel.

Ans: In monolithic kernel, all operating system programs reside in the kernel space. In contrast, in microkernel, only a small fraction of the operating system programs resides in the kernel space; the rest resides in the user space.

10. If we execute kernel programs in the user mode and applications in the kernel mode, what are the consequences?

Ans: The kernel will malfunction because it cannot execute privileged instructions to control hardware resources. Applications can do anything in the system and may throw the system in a state of chaos.

11. In UNIX systems, the superuser is very special and has the most privileges. When she runs an application, can it execute privileged instructions in the user mode? Justify your answer.

Ans: No, no one can execute privileged instruction in the user operating mode.

12. Why do we say that the operating system is a resource manager?

Ans: Because it manages all (hardware and software) resources in the computer system.

13. Why do we say that the operating system is a reactive program?

Ans: The OS is not an active program in a generic sense that it starts with some input, crunches the input, produces some output, and finally exits. Instead, it is driven by three kinds of events: system call, exception and interrupt. It reacts to those events.

Guidelines: Process, threads, process synchronization, deadlocks

14. Which process can be affected by other processes executing in the system?

a) cooperating process b) child process

c) parent process d) init process

Ans: **a)**

15. A semaphore is a shared integer variable

a) that can not drop below zero b) that can not be more than zero

c) that can not drop below one d) that can not be more than one

Ans: a)

16. In UNIX, does exec create a new process? Justify your answer.

Ans: No. It reuses the caller's pid along with the process descriptor.

In Unix whenever we want to create a new process, we fork the current process, creating a new child process which is exactly the same as the parent process; then we do an exec system call to replace all the data from the parent process with that for the new process.

Linux **Exec System Call**. The **exec system call** is used to execute a file which is residing in an active process. ... The user data segment which executes the **exec() system call** is replaced with the data file whose name is provided in the argument while **calling exec**(). The new program is loaded into the same process space.

Guidelines: 1. CPU scheduling problems as discussed in class should be studied (already shared slides).
2. Semaphore should be studied
3. Deadlock: Banker's algorithm problems are discussed in class. ***Important for internals.***

17. Explain why UNIX systems use the zombie state? Is this an execution state of a thread or a process?
Ans: Zombie is a process state. When a process (in any thread) executes the exit statement, the operating system cleans up the process, but keeps the process descriptor in the zombie state. The process remains in the state until its parent obtains its status information.

18. In the context of CPU scheduling, what is starvation? Can starvation occur under the following scheduling schemes?
FCFS; ii) SJF; iii) SRTN; iv) Preemtive priority based; iv) RR
Explain your answer in each case. In the case of possible starvation, briefly describe a method by which it can be prevented?
Ans: Starvation is the situation in which some processes perpetually wait in the ready queue while the CPU is allocated to other newly requesting processes. That is, starvation is a situation where a ready process never gets the CPU. Starvation can happen in SJF, SRTN, and priority based scheduling, and not in FCFS and RR. By definition, we can not change the priority of processes in SJF and SRTN, and we cannot avoid starvation. We can avoid starvation is priority based scheduling by introducing the concept of aging – which dynamically increases priority of low-priority processes.

19. When do we say two processes are concurrent? Explain with example.

Ans: Two processes are said to be concurrent if the first operation execution of each process starts before the last operation execution of the other process is complete.

20. What are interacting processes? When do we say that two processes do not interact.

Ans: Processes, during their life time, access various data item. If some data items are common between two processes and they access the data in a conflicting way, then they are interacting processes; otherwise, they are not.

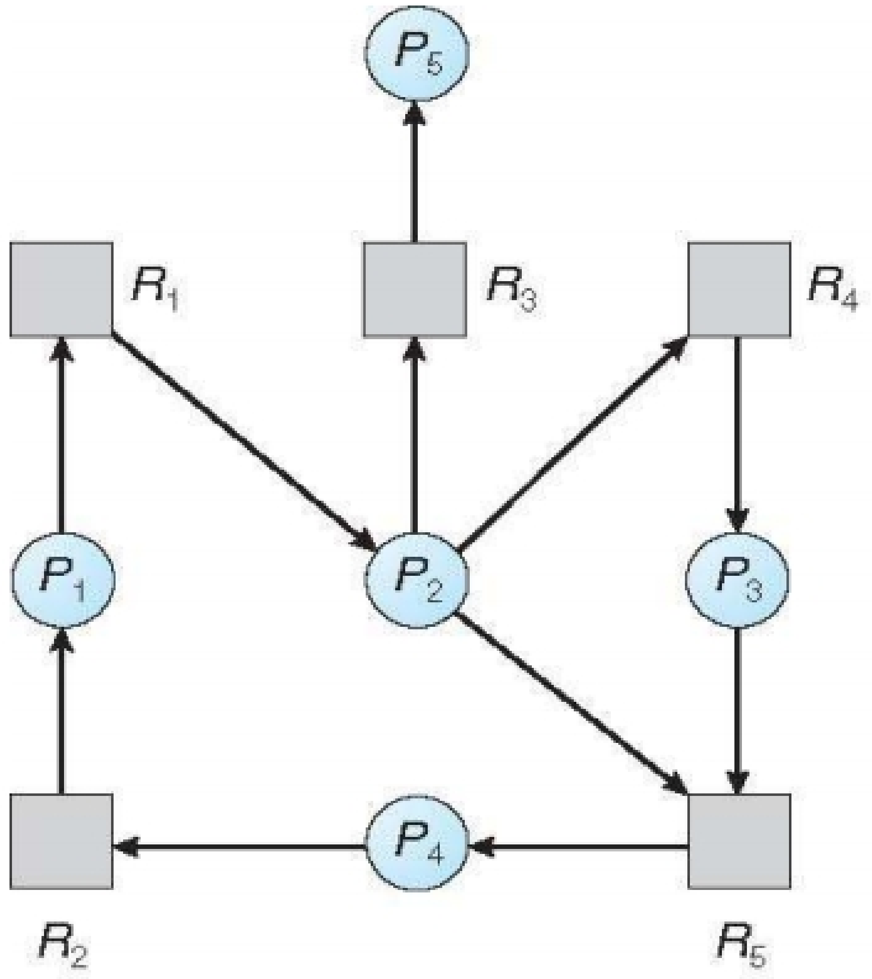
21. Can two non-concurrent processes interact with one another?

Ans: No, because one is complete before the other one starts. The earlier one, of course, can leave some information for the latter one.

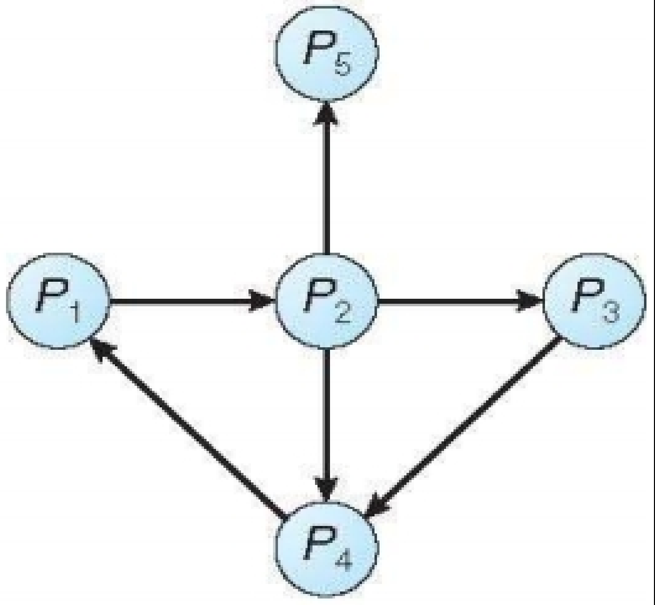
22. What is synchronization?

Ans: It is a way of coordinating a group of processes in their activities to ensure that no process accidentally does something on a data that another process is working on.

Resource-Allocation Graph and Wait-for Graph



(a)



(b)

23. What is the difference between interprocess communication and synchronization? How they are similar?

Ans: They both exchange information among processes. In interprocess communications, one process knows that the other process is active. In synchronization, the other process may not be active.

24. How is starvation different from self-lock?

Ans: A self-lock is caused by the process itself, but a starvation is caused by others.

25. An un-interruptible unit is known as :

A. single **B.** atomic **C.** static **D.** None of these

Ans: B

26. The TestAndSet instruction is executed :

A. after a particular process **B.** periodically

C. atomically **D.** None of these

Ans. C

27. Semaphore is a/an ________ to solve the critical section problem.

A. hardware for a system **B.** special program for a system

C. integer variable **D.** None of these

Ans:C.

28. The code that changes the value of the semaphore is :

A. remainder section code **B.** non – critical section code

C. critical section code **D.** None of these

Ans. C

29. What is safe state?

Ans: State is safe if the system can allocate resources to each process (up to its maximum) in some order and still avoid a deadlock. More formally, a system is in a safe state only if there exists a safe sequence.

30. What are the mechanisms to recover from deadlock?

Ans. 1) Process termination ; 2) Resource pre-emption ; 3) Check point / roll back mechanism

Process termination

❑ Abort all deadlocked process

❑ Successively abort each deadlocked process until the deadlock no longer exists.

Resource pre-emption

Roll back : A process that has a resource pre-empted from it must be roll back to the point to its acquiring of that resource. Total roll back – Abort the process and restart it.

Check point

•Keep checkpointing periodically

•When a deadlock is detected , see which resource is needed

•Take away the resource from the process currently having it

•Restart the process from the checkpointed state.