OS

1) Semaphore :- It is an integer variable, used to ~~concurrency~~ control access to common resource by multiple processes in system.

→ It is an integer variable that, apart from initialization, is accessed only through two standard atomic operations, wait & signal.

2A) Monitor is ~~chara~~ it is an abstract data type whose construct allow one process to get activate at one time
→ It is an abstract data type used for process synchronization

3A) The semaphores which are restricted to 0 & 1 values are called binary semaphors

4A) Semaphores which allow an arbitrary resource count of non-negetive integer values - counting semaphore

5A) wait(), signal()

6A) While a process is in critical section, another process that tries to enter critical section must loop continuously. this looping is a problem in system, where only 1 CPU is shared among process, this is called as busy waiting

7A) Busy waiting wastes CPU cycles that someother process might be able to use. This type of semaphore is spinlock.

8A) It has boolean variable 'available' whose value indicates lock is available or not
If call to acquire succeeds the lock is unavailable & process attempts to acquire "unavailable", lock is blocked till lock is released.

9A) The high priority process waiting for a lower-priority one to finish is known as priority inversion.

10A) mutex, wrt, readcount

11A) mutex, full, empty

12A) Chopstick[].

13A) monitor monitor-name
{
procedure body P1()
{
}
;
{ initialization
}
}

14A) x.wait() ; x.signal()

wait(s) { while(S≤0) S--; }

15M) wait() → P (for wait; from Dutch Proberen, to test)

16M) signal() → V (for signal; from verhogen, to increment)

```
signal(s)
{ S++;
}
```

17M) Semaphore is a structure variable which consists of an integer value & list of processes available.

18M) It is a situation where processes wait indefinitely within the semaphore.

IV

1M) Fixed ——— equal — Internal Fragmentation
unequal → ~~Individual queues using~~ Using different ques
Dynamic. ↘ Using single queue.
/
External Fragmentation
(Can be solved using compaction)

2M) The memory allocated to proces may be slightly larger than requested. The difference b/w is internal fragmentation. memory that is internal to ~~partion~~ partition but is not being used.

3M) When enough total memory space exists to ~~specify~~ satisfy a request, but not contiguous, storage is fragmented into a large no. of small holes. This is external fragmentation.

4M) It is process of moving allocated objects together & creating empty space together.