(1) How can “race condition” happen? Show “how” using an example.

Race conditions happen when machine codes (or processor instructions) for updating the contents of a shared resource, shared by more than one process, are interrupted (or context-switched) as shown below:

(2) What is “critical section”?

A critical section is a portion of (or a set of instructions in) a program/a process that can cause a race condition.

OR

A critical section is a portion of (or a set of instructions in) a program/a process that must be executed by at most one process at a time (to prevent a race condition).
(3) What is “mutual exclusion”?

Mutual exclusion is a solution for a race condition, which makes sure only one process (can execute instructions) in a critical section at a time.

(4) Explain how a semaphore can prevent race condition.

A (binary or mutex) semaphore prevents a race condition by guarantees that at most one process can be active in a race condition at a time.

(5) What “wait” system call to a semaphore exactly performs?

The “wait” system call on a semaphore performs the following activities (as an atomic operation):

- If a semaphore is ‘1’, the OS decreases the content of the semaphore to ‘0’. Then, the OS lets the process (the process that calls “wait”) to proceed into the critical section.

- If a semaphore is ‘0’, the OS brings the calling process to wait (i.e., to the “blocked” state in the short-term scheduler).

Note: both of the two descriptions (one for S = ‘1’ and the other for S = ‘0’) are necessary for full credit.

Wait

If S > 0, do S = S - 1 then proceed
If S = 0, wait on this semaphore