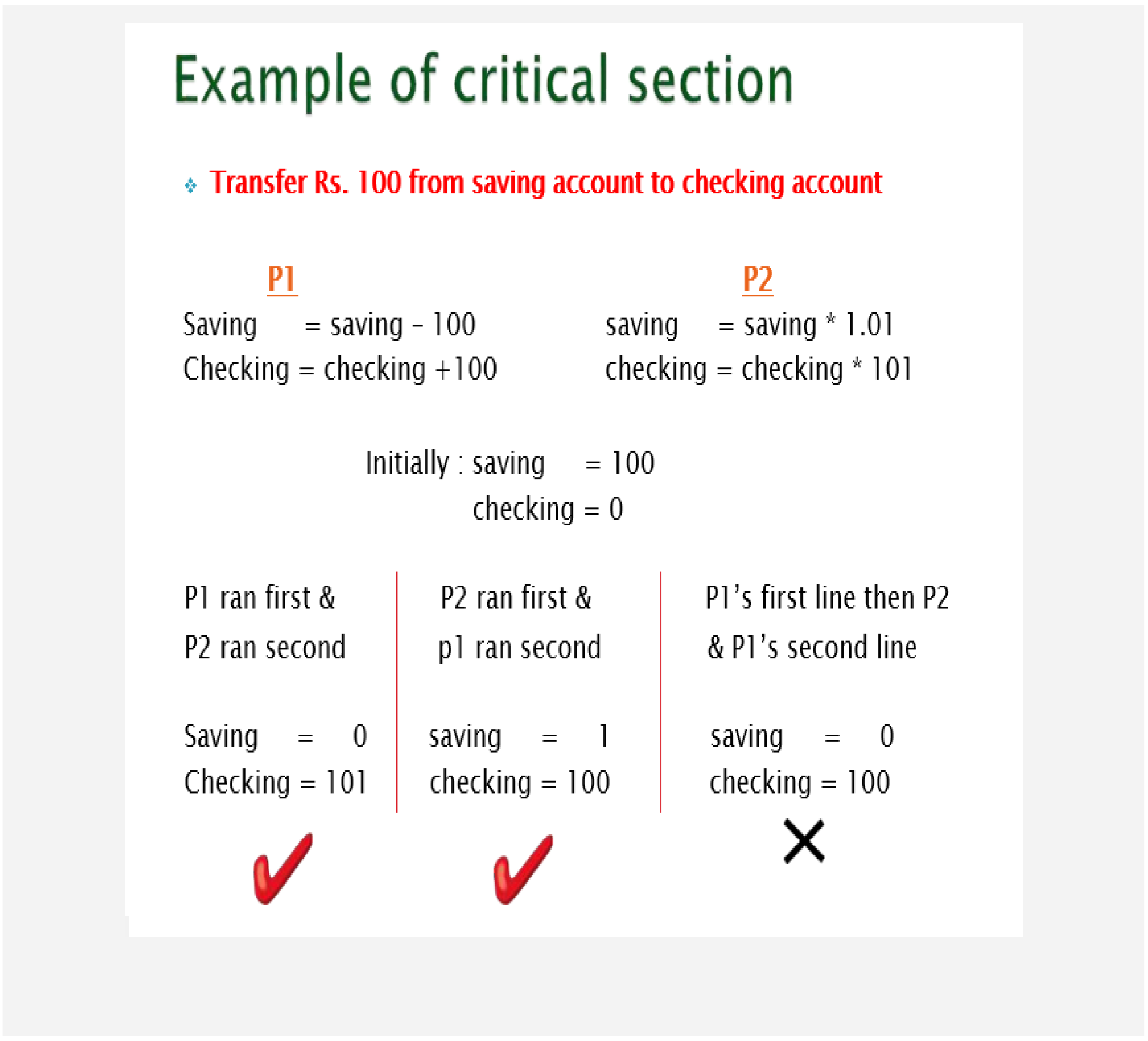
Critical Section Problem

When a process is accessing shared modifiable data or a resource that can only operate on behalf of one process at a time , the process is said to be in a critical section. When one process is in a critical section , all other processes (at least those that access the shared modifiable data and/or resource) are excluded from their critical section.

The Critical Section Problem

- n processes all competing to use some shared data
- Each process has a code segment, called the critical section, in which the shared data is accessed.
- Problem – ensure that when one process is executing in its critical section, no other process is allowed to execute in its critical section.



Example of critical section

- Transfer Rs. 100 from saving account to checking account

P1	P2
Saving = saving - 100	saving = saving * 1.01
Checking = checking +100	checking = checking * 101

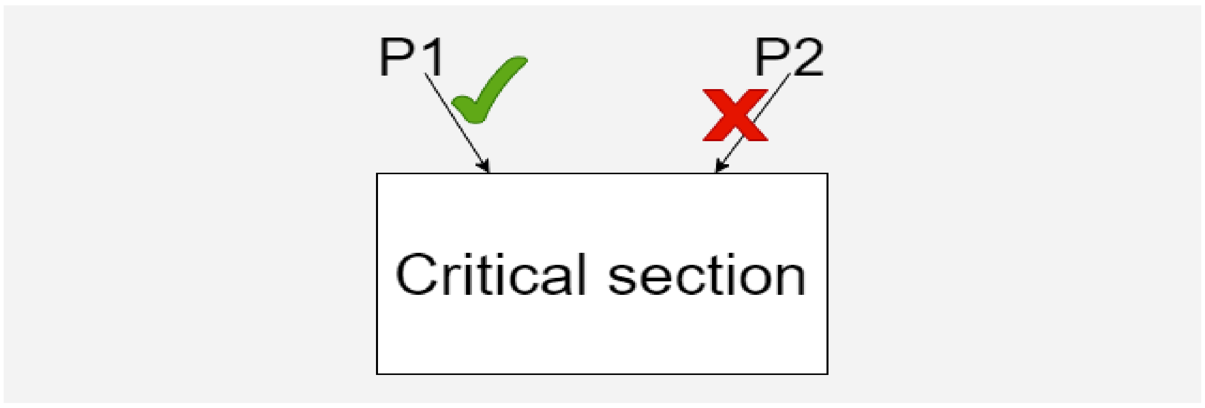
Initially : saving = 100
checking = 0

P1 ran first & P2 ran second	P2 ran first & p1 ran second	P1's first line then P2 & P1's second line
Saving = 0	saving = 1	saving = 0
Checking = 101	checking = 100	checking = 100
✓	✓	✗

Solution To Critical Section Problem

1.Mutual Exclusion: If process P_i is executing in its critical section, then no other processes can be executing in their critical sections.

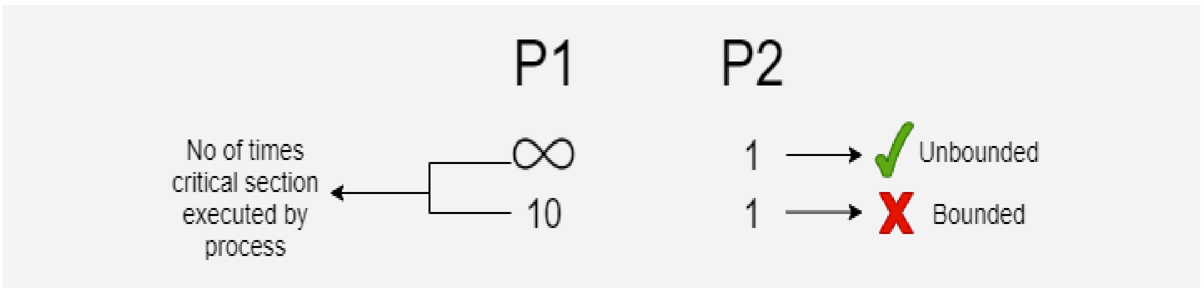
Suppose if P1 wants to execute a critical section and start executing it, and then at the same time (while P1 was executing critical section) P2 wants to enter the critical section in that case P2 will be blocked.
And by that we will achieve mutual exclusion.



2.Progress: If no process is executing in its critical section and there exist some processes that wish to enter their critical section, then the selection of the processes that will enter the critical section next cannot be postponed indefinitely.
Example:
Suppose there is some common code written in the critical section and at present the critical section is empty (means that there is no process in it).
So, now P1 wants to enter the critical section and P2 is blocking it. Maybe because of some code written in the enter section of P2.
Therefore, in this case P2 is not going in the critical section and also blocking P1 to enter the critical section.
This means that progress is not there and this should not happen.

3.Bounded Waiting: A bound must exist on the number of times that other processes are allowed to enter their critical sections after a process has made a request to enter its critical section and before that request is granted.

- Assume that each process executes at a nonzero speed
- No assumption concerning relative speed of the n processes.


P1
P2
∞
10
1
1
Unbounded
Bounded
No of times critical section executed by process