How to check if a particular software solution to the critical section problem satisfies below requirements :-

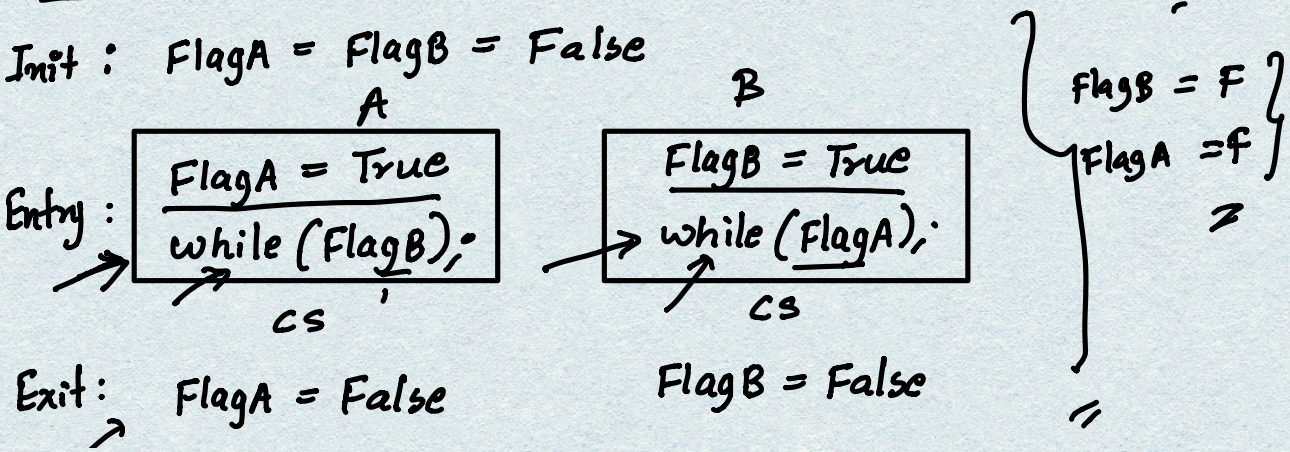
1) Mutual Exclusion ✓
2) Progress - with and without deadlock ✓
3) Bounded wait ✓

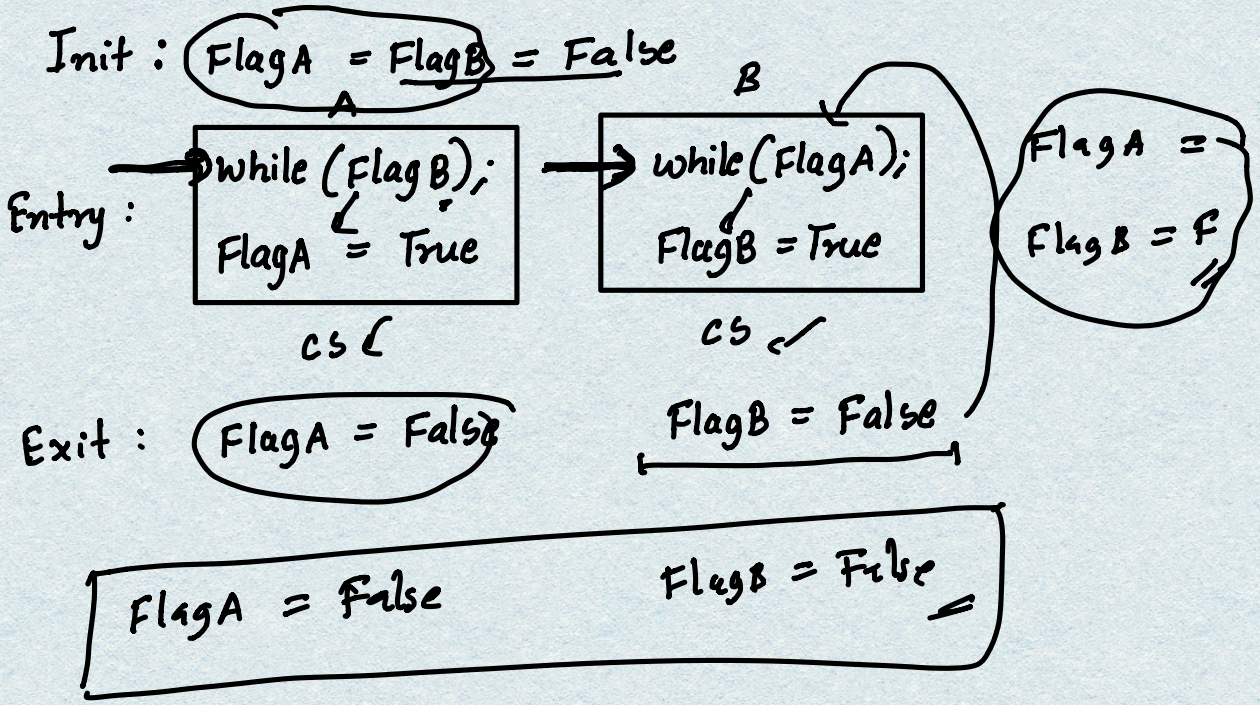
In this video, we look at some methods/tricks to easily check the above requirements.

Mutual Exclusion ✓

Try making while statements (Entry Section) False for two processes at the same time. If you can achieve this, the solution voilates mutual exclusion.

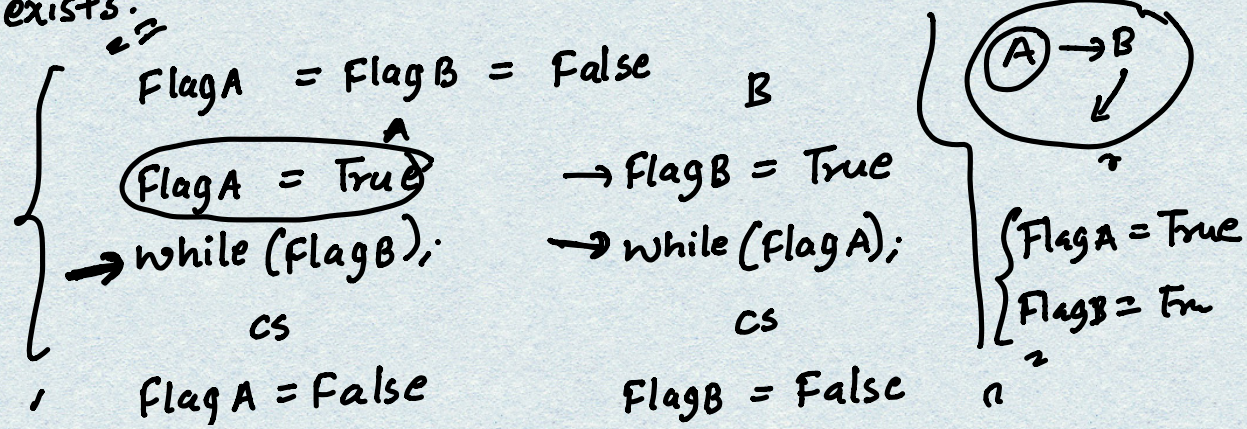
Init : FlagA = FlagB = False

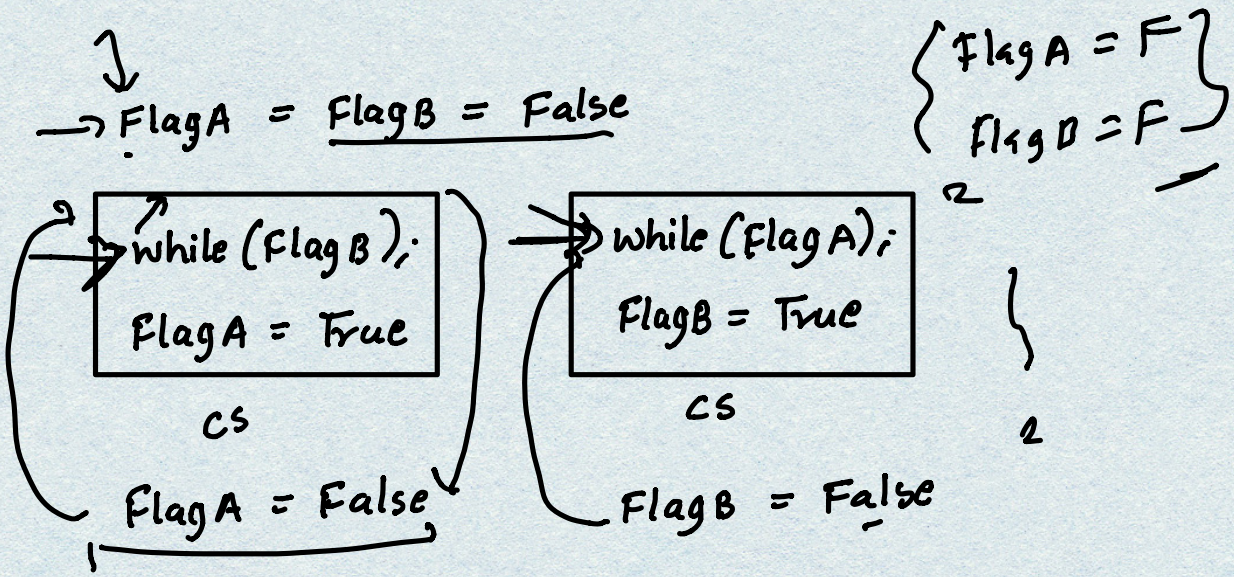




Progress - Deadlock

Try making while statements True for both the process at the same time. If you can achieve this, deadlock exists.





Progress without deadlock

Let the process A first enter the critical section and come out. Let A sleep now.

Give two opportunities to process B for entering critical section. If it fails in any one of these, progress is voilated.

turn = B / A

→ turn = A // B

→ while (turn != A);

CS

(A) turn = B

→ (Sleep)

(B) ←

while (turn != B);

→ CS

turn = A

Flag A, Flag B are randomly assigned True/False

{ while (FlagA == FlagB);

CS

FlagA = FlagB }

(B) →

{ while (FlagA != FlagB);

→ CS

FlagB = not(FlagA)

(A) Sleep

Progress

(FlagA = FlagB = T)

{ FlagB = False

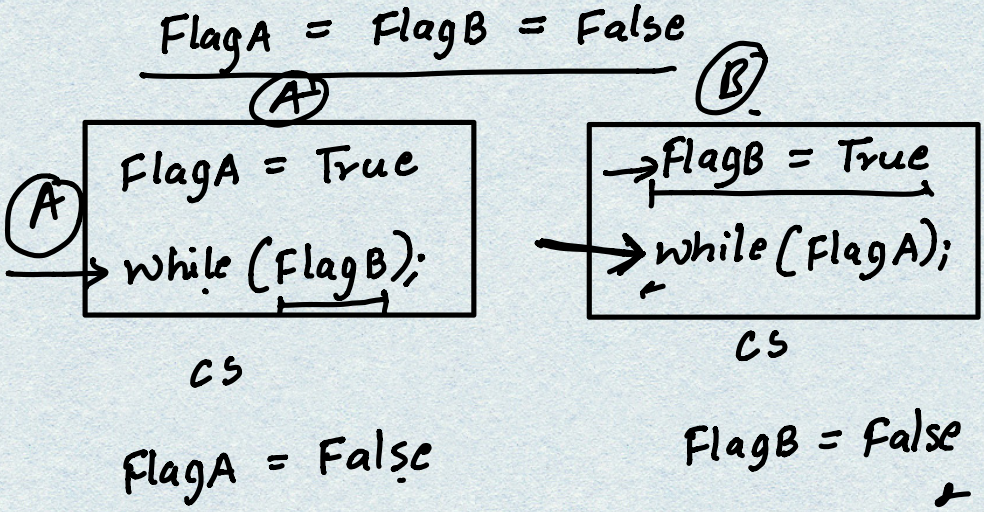
FlagA = True

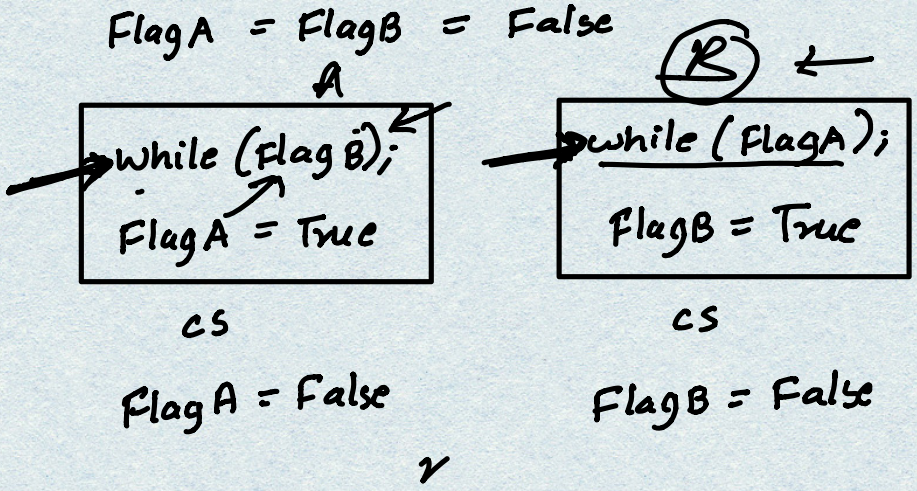
Bounded wait

Let process A enter the critical Section first and come out. Let A again try to enter the critical Section.

Assume that, when A is trying to execute the while loop 2nd time, B is also executing it.

If A can again enter the critical section, there is a high chance it voilates bounded wait.




FlagA = FlagB = False
A
B
while (FlagB);
FlagA = True
while (FlagA);
FlagB = True
CS
CS
FlagA = False
FlagB = False