THEORY ASSIGNMENT-1

Name:V Sri Charan Reddy
Roll No:CS16BTECH11044

2)
The following presented solution alleviates the simultaneous use of two mutexes for a Reader.
Initialisation code:
in = Semaphore(1)
out = Semaphore(1)
wrt = Semaphore(0)
ctrin = Integer(0)
ctrout = Integer(0)
wait = Boolean(0)
Reader code:
- Wait in
- ctrin++
- Signal in [Critical section]
- Wait out
- ctrout++
- if (wait==1 && ctrin==ctrout)
 then Signal wrt
- Signal out
Writer code:
- Wait in
- Wait out
- if (ctrin==ctrout)
then Signal out else
- wait=1
- Signal out
- Wait wrt
- wait=0 [Critical section]

- Signal in

The main idea here is that a Writer indicates to Readers its necessity to access the working area. At the same time no new Readers can start working. Every Reader leaving the working area checks if there is a Writer waiting and the last leaving Reader signals Writer that it is safe to proceed now. Upon completing access to the working area Writer signals waiting Readers that it finished allowing them to access the working area again. It also works with no starvation with semaphores where waiting threads selected randomly by the operating system, because the probability that thread does not get a chance to execute decreases with time exponentially.

1)

Assuming there are n producer's and n consumer's

a)

(i) The minimum for empty is -n. Reason: If only producers are running and no consumer's are running then after at instant when buffer is filled then empty is 0 and still n producers try to produce and get blocked and make empty to -n and which is minimum.

(ii) The minimum for full is -n.Reason: If only consumers are running and no producer's are running then after at instant when buffer is empty then full is 0 and still n consumer's try to consume and get blocked and make full to -n and which is minimum.

b)

(i) The maximum of empty is n. Which is the initial case, this is because when a produces a process then empty decreases. And if consumer's try to consume at initial case then value of full changes and their would be no effect on empty. So, maximum of empty is n.

(ii) The maximum of full is n. Reason: If only producers are running and no consumer's are running then after at instant when buffer is filled then full is 0 and still if any producers try to produce and get blocked and doesn't make any difference to full and if consumer consumes then value of full decreases.So maximum full is n.

c)

(i) Maximum of sum of full + empty is n.This at the initial instant.

(ii) Minimum of sum of full + empty is 0. Reason: When only producers or only consumers are running then they reduce empty or full (based on producer or consumer) and get blocked So at an instant either full will be decreasing when consumers are running, empty is n. That provides full to reduce to -n and when producers are running with full n and empty n which decreases empty to -n and not changing full . So minimum of full + empty is 0.