Nirbhay Sharma (B19CSE114)
Lab - 6

Part - 1

Without_deadlock solution

It will allow the philosopher to pick both the fork simultaneously and otherwise let them wait.

How to run

gcc no_deadlock.c -pthread ;./a.out

With deadlock + resource allocation graph solution

If deadlock occurs it will allow the parent process to preempt the philosopher and let them leave the fork they are currently occupying.

How to run

gcc with_deadlock.c -pthread;./a.out

Part - 2

Starvation free solution

1) If we set the limit for the number of readers that at the same time only K readers can read the data and till then writer will wait but after the threshold of K we can see that if there is a reader present or not and if present then reader can unlock and writer can write but if no writer is currently there then no need to stop the readers, in that case the number of readers may also reach above K and then we can check if the number of readers are greater than K and writer is waiting then reader will unlock and writer will proceed we can implement this using semaphores on number of readers currently there
2) Another solution can be to assign severity of writing to writer process each time when writer process will come first we see the severity or urgency of writing, if it is greater than a particular threshold then reader can unlock and let writer enter but if it is less than the threshold then we will let the readers continue, but this solution still can face issues of starvation so we propose another solution to this problem
3) We can try to blend the ideas of both the above solutions and based on severity / urgency of writing and also the threshold of number of readers we can release and apply the lock accordingly so in that case if a particular writer is assigned less severity then instead of waiting infinitely it will have to wait till the reader process threshold has reached and then automatically it will have the chance of writing.