##### PROJECT-REPORT

We have implemented project-1 using Link list. For synchronization (4 counting semaphore for concurrent execution of the threads) and mutual exclusion (For mutual exclusion 3 binary semaphores) Semaphore are used.

Function starts execution from the main (). It prompt for the size of free list to be given as an user input. Then semaphores are initialized and all the three lists are created.

Then n (input from the user) numbers of nodes are appended to the free list. Three thread th1,th2 and th3 are created.

Semaphore implementation approach: -

\*\*Detailed Pseudo code can be find in the attached Semaphore.txt file

1. We are using count i.e. initialized to n-1. Where n is the number of nodes in the free list. count semaphore is used to make sure that Thread 2 will get at least one element from the free list while Thread 1 is executing.
2. Execution of Thread 2 is checked by list\_1 semaphore i.e. Thread -2 cannot start until there is some element linked to the list\_1 list.
3. Similar to Thread-2, Thread-3 will only be executed when we have some element in the list\_2.
4. Mutual exclusion is provided by the f\_b, l1\_b, l2\_b binary semaphores.

Our project is executable in CSE machine. It is under the name mutex.c. Process will keep executing until is stopped by external interference. Hence our implementation meets Both the objective of the project-1 i.e. Deadlock free and Concurrent.