**NAME: PURUSHOTTAM KUMAR**

**REG: 11807502**

**B-TECH CSE (2ND YR)**

**OPERATING SYSTEM ASSIGNMENT**

**GitHub link:** [**https://github.com/puru531/OS-Assignment**](https://github.com/puru531/OS-Assignment)

**Problem** For this part of the assignment, you are expected to implement a solution to a synchronization problem called the “cats and mice” problem. The synchronization primitives that you may use in your solution are semaphores, locks, and condition variables. You are free to use whichever of these synchronization primitives you choose, however you like. However, you must not directly use any “lower-level” methods of synchronization, such as wait channels or spinlocks.

**Description:**

Your first task is to implement locks. The interface for the lock structure is defined in You can use the implementation of semaphores as a model, but do not build your lock implementation on top of semaphores or you will be penalized. In other words, your lock implementation should not use sem create(), P(), V() or any of the other functions from the semaphore interface. Locks are used throughout the kernel. You will need properly functioning locks for this and future assignments to ensure that the kernel’s threads are properly synchronized. Because of this, implementing locks correctly - though not difficult - is the most important part of this assignment. Make sure that you get locks working before moving on to the other parts of the assignment.

**Algorithm:**

implements a simple test case for locks, and another for condition variables. You can run the lock test from the kernel menu by issuing the sy2 command Similarly, the sy3 command will run the condition variable test. If the lock test reports “Lock test done” without reporting any failure messages, it has succeeded. The output from the condition variable test should be self-explanatory. Testing synchronization primitives like locks and condition variables is difficult. Both sy2 and sy3 are subject to false positives. In other words, an incorrect lock or condition variable implementation may 1 pass these tests. However, if your implementation fails a test, there is definitely a problem. Since the synchronization tests are not perfect, we will use code inspection - in addition to testing - to evaluate your lock and condition variable implementations. Although you are free to implement locks however you want, you should not modify any of the kernel’s test programs, i.e., do not modify any of the files in kern/test. Furthermore, you should not make any changes to the way that the tests are invoked, e.g., do not change “sy2” to “sy2a”

**Github Revision:**

* I have made 3 revision of code on Github.

**Github link** [**https://github.com/puru531/OS-Assignment**](https://github.com/puru531/OS-Assignment)