# Operating Systems Lab

**Semaphores**

**Task 1:**

You have to simulate the opening bridge that usually build upon river/sea. Take input from user: number of cars and number of boats Also input time (in seconds) needed by a boat to pass through bridge and time (in seconds) needed by car to pass through bridge You have only one bridge (Single Lane) Only one boat or car can pass through bridge at a particular time.

Note: Use Binary Semaphore in this task

**Task 2:**

Edit the Lab Task 1 according to the following constraints. Constraints: At a time, at max. TWO (2) boats can pass under the bridge or at max. THREE (3) cars can pass over the bridge at a particular time. No more than 2 boats or 3 cars can pass at a time. Check your code by taking 10 cars and 10 boats for example.

Hint: You can use counting Semaphore for this task

**Task 3:**

Implement Readers-Writers Problem.

Readers-Writers Problem

Consider a situation where we have a file shared between many people.

If one of the person tries editing the file, no other person should be reading or writing at the same

time, otherwise changes will not be visible to him/her.

However, if some person is reading the file, then others may read it at the same time.

Precisely in OS, we call this situation the readers-writer, problem

Case Process 1 Process 2 Allowed/Not Allowed

Case 1 Writing Writing Not Allowed

Case 2 Writing Reading Not Allowed

Case 3 Reading Writing Not Allowed

Case 4 Reading Reading Allowed

Writer process:

do {

// writer requests for critical section

wait(wrt);

// performs the write

// leaves the critical section

signal(wrt);

} while(true);

Reader process:

do {

// Reader wants to enter the critical section

wait(mutex);

// The number of readers has now increased by 1

readcnt++;

// there is atleast one reader in the critical section

// this ensure no writer can enter if there is even one reader

// thus we give preference to readers here

if (readcnt==1)

wait(wrt);

// other readers can enter while this current reader is inside

// the critical section

signal(mutex);

// current reader performs reading here

wait(mutex);   // a reader wants to leave

readcnt--;

// that is, no reader is left in the critical section,

if (readcnt == 0)

signal(wrt);         // writers can enter

signal(mutex); // reader leaves

} while(true);